Estimating the vertical direction in a photogrammetric 3D model, with application to visualization

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

The reference of this paper is: Maxime Lhuillier, Estimating the vertical direction in a photogrammetric 3D model with application to visualization, Computer Vision and Image Understanding, 236:1-13, 2023.

This is the accepted manuscript version that is available at the webpage of the author. The published version (DOI: 10.1016/j.cviu.2023.103814) is available at Elsevier via `https://doi.org/10.1016/j.cviu.2023.103814`

Highlights

- Start from a triangulated surface computed by photogrammetry
- Estimate the vertical direction by a 3D PCA and a 2D Hough transform
- Experiment on both man-made and natural environments reconstructed from 360 videos
- Build a VR viewer from a surface and its vertical direction thanks to a game engine
- Visualize and move in large environments thanks to VR viewers (available online)

Abstract

The process of building a Virtual Reality (VR) environment from images involves several steps: 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 postprocessing technique for the photogrammetry step, mostly for outdoor environments that cannot be reconstructed using an unmanned aerial vehicle. As visualization applications (including VR) need a 3D model with a known vertical direction, a method is introduced to compute it. The method is based on 3D principal component analysis and a 2D Hough transform. In the experiments, we first reconstruct both man-made and natural immersive environments using a helmet-held 360 camera, then we import the 3D models in good coordinate systems (i.e. with a vertical axis and a plausible scale) into Unity, and finally we use VR headsets to explore the scenes like a pedestrian. We also experiment on scanner data and show that our method is competitive with previous work.

Keywords: Vertical Direction, Photogrammetry, Virtual Reality, Hough Transform, Principal Component Analysis, 360 camera.

1. Introduction

Knowledge of the vertical direction (or gravity direction) is very useful for visualization of a reconstructed environment: it needs to be known for drawing as vertical lines, in a screen or a head-mounted display, vertical lines in 3D when the user looks the reconstruction with usual poses of the head. Otherwise vertical scene components such as walls and facades appear to be oblique, which makes the result unrealistic. The orientation of the vertical (upward or downward) is also needed to visualize the reconstruction without switching its top and its bottom.

Visualization applications like Google Earth and Sketchfab enforce a zero-roll and vary the yaw-pitch of the camera that draws a 3D model. They need a reference upward vertical direction to define the roll-pitch-yaw angles. The game engines used to build VR applications need a 3D model in a good coordinate system: one of the

three coordinate axes has to be upward vertical and the scale has to be plausible. If the vertical direction is inaccurate, the risk of VR sickness (e.g. nausea) increases since there is a sensor conflict between the vertical seen by the eyes and the vertical detected by the vestibular system of the VR user ([Tian20]). A plausible scale is also useful for stereoscopic rendering and for moving in the VR scene at a realistic speed.

Unfortunately, the Structure-from-Motion step typically included in photogrammetry software to reconstruct a 3D model does not provide a vertical direction and does not provide a good coordinate system. Indeed this step can only reconstruct a scene up to a similarity transform of the 3D space, if the acquisition camera has only one center like a pinhole camera ([Hartley00]). If the acquisition camera has several centers, like most 360 cameras composed of several monocular cameras, and if the baseline between the component cameras is known, the scene can only be reconstructed up to an Euclidean transform.

In practice, the vertical direction is estimated manually using a graphical user interface, e.g. that of game engines like Unity. But it is difficult to obtain an accurate vertical for large environments by this way. Alternatively, the vertical can be computed up to orientation by assuming that the camera motion is on a horizontal plane during the acquisition of the images. The orientation then becomes easy to define either manually or by an additional assumption (e.g. the ground is projected in the bottom of the images) by multiplying the vertical vector by -1 if needed. This often gives a satisfactory render, but there are scenes where the vertical estimation thus estimated is too inaccurate.

The paper mainly focuses on a simple new method to compute the unoriented vertical direction in the 3D model (the photogrammetric output) thanks to three assumptions

- **A0**: the vertical direction does not vary in the 3D model
- **A1**: the height of the camera (with respect to the ground) during acquisition is roughly constant
- **A2**: the density of the surface normal is higher at horizontal directions and near the vertical direction than elsewhere.

We need A0 to define the oriented vertical by a single 3D vector. A0 is common in the Computer Vision literature involving the vertical direction. It is met if both rotation drift (due to Structure-from-Motion) and diameter of the input model (relative to the whole Earth) are small enough. A1 is met in terrestrial acquisition settings, e.g. a helmet-held or car-mounted camera. We justify A2, the anisotropy assumption, as follows. The ground has a large area and its slope is moderated. Other scene components, such as facades in man-made scenes, have substantial vertical areas. So the most probable normals of the input surface are therefore near the vertical direction and at horizontal directions, respectively. A2 looks like the Atlanta world assumption for normals ([Joo18]) but is different: unlike the Atlanta world assumption, we do not assume that the ground surface is horizontal.

The paper also summarizes the build of VR viewers based on photogrammetric models of environments reconstructed by a helmet-held 360 camera. This is useful for applications such as virtual tourism ([Poux20]), games ([Dhanda19]) and Metaverse ([Wikipedia22]). It may also interest researchers who want to use common VR headsets to display photogrammetric reconstructions but don't know how to do this. Surprisingly, this use of VR remains rare, despite the fact that VR headsets now deliver good performances at relatively affordable price. The advantage of a 360 camera is an acquisition with a minimal effort to reconstruct complete environments, but other camera setups are possible.

2. Previous work

Photogrammetry software provides neither vertical direction nor scale, unless its Structure-from-Motion step integrates measurement input from an additional sensor (Sec. 2.1). Furthermore, previous methods estimate the vertical from a cloud of points or normals (Sec. 2.2). Other methods also estimate directions in 3D by using 2D Hough transforms (Sec. 2.3). Last we briefly summarize the workflow from photogrammetry to VR (Sec. 2.4).

2.1. Additional sensors

Measures from other sensors like GPS and IMU can be integrated with both terrestrial ([Klingner13]) and aerial ([Grayson18]) imagery, if these sensors are rigidly fixed to the camera. Bundle adjustment ([Triggs99]) simultaneously refines parameters of both camera poses and

3D points by minimizing a sum of reprojection errors, which can be augmented by a GPS term ([Lhuillier11]) and an IMU term ([Michot10]). Bundle adjustment needs a parameter initialization, which can also be done thanks to GPS ([Crandall11]) and IMU ([Klingner13]). GPS provides georeferenced reconstruction, which implies that both vertical direction and scale are known. Unfortunately, GPS gives unreliable measures in cluttered-scene terrestrial locations where GPS satellites are occluded. IMU measures both translation acceleration (including gravity) and angular velocity. A heuristic is to compute the vertical by the direction of this acceleration if its modulus is that of gravity ([Lobo03, LaValle20]). [Dai17] first compute a rough vertical direction from the IMU measures of a iPhone/iPad, then refine it for a Kinect-like RGB-D capture system applied to indoor scenes. Note that our method presented here does not require additional sensors.

2.2. Estimating the vertical from a cloud of points or normals

The estimation of the vertical direction is also the first step of methods that align several point clouds acquired by a scanner. These methods assume that the ground is horizontal, as in indoor scenes ([Alnuaimi17]), or that there are vertical walls as in urban environments ([Avidar18]). In the work of [Alnuaimi17], the vertical is initialized by the third principal component of the point cloud. Then it is refined as a median vector of the normals that are close to the initialization. The method of [Avidar18] starts from an initial estimate of the vertical, then detects walls with points whose normals are roughly orthogonal to the initial vertical, and finally applies a RANSAC procedure on pairs of normals of detected walls. In a recent development, [Liu22] compute the vertical using the Atlanta world assumption. They estimate an unit vector that is parallel or orthogonal to a maximum number of normals, up to a given angular tolerance. They introduce several methods, including RANSAC and Branch-and-Bound. These works can be applied to our problem after a conversion of the input (a triangulated surface). Nevertheless, they also have drawbacks: no distinction (e.g. weight) between the normals, strong assumptions ([Alnuaimi17, Avidar18]), or no prior on the vertical direction ([Liu22]). To the best of our knowledge, there is no deep learning method that predicts the vertical direction of a triangulated surface or a cloud of points/normals.

2.3. Estimating directions using Hough transforms

Although the Hough transform classically computes lines from edges detected in images ([Mukhopadhyay14]), it can be adapted to compute directions in the unit sphere of $\mathbb{R}^3$. [Rabbani05] estimate cylinders from a cloud of 3D points obtained by a 3D scanner in an industrial site. The first step is a 2D Hough transform that computes potential directions of the cylinders. Here we also use a 2D Hough transform but with differences: there is only one direction to compute (the vertical), we do not assume a set of circular cylinders in the scene, the input is a triangulated surface, the votes are weighted. Hough transforms are also used by [Lutton94] to estimate the three main orthogonal directions of a man-made scene, from edges detected in an image taken by a calibrated perspective camera. The first step is a 2D Hough transform that computes potential vanishing points. In the work of [Lutton94] and [Rabbani05], the sampled parameter space of the directions (or Hough space) is an approximate uniform sampling of the unit (hemi)sphere obtained from spherical coordinates. This is adequate enough to obtain directions with a similar accuracy and if the potential directions are equiprobable.

In a previous conference version ([Lhuillier21]) of our work, the sampled parameter space is obtained by the backprojection of a perspective camera. The current version adds new content: comparisons with work of [Liu22, Alnuaimi17, Avidar18] and with a parameterization based on the spherical parameterization of [Lutton94], comparisons with other sensors, experiments on a greater photogrammetric dataset and a new scanner dataset, more details on the method (parameter tuning, sampling and voting) and the VR viewers, and accuracy improvement thanks to smoothing.

2.4. Workflow from photogrammetry to VR

Building of a VR application (here, a viewer) from a textured and triangulated surface computed by photogrammetry is not a straightforward task, for several reasons. The surface has to be simplified and optimized with its texture for real-time rendering. A manual clean up is useful in order to remove errors or to select segments of interest in the reconstructed surface. Both

[Obradovic20] and [Valve21] describe this kind of workflow and the tools available. The textured 3D model is then imported in a game engine like Unity ([Mel19]), Unreal ([Obradovic20]) or SteamVR ([Valve21]). The game engine includes a graphic user interface to define the upward vertical direction and the scale. It also serves to build VR applications that have to deal with several things. The VR user needs to have a strong spatial awareness (or wayfinding) and a little risk of VR sickness while moving in the VR environment ([LaValle20]). The former implies the ability to find its way and the latter includes nausea and headaches. Motion must be possible even if the VR environment is greater than the tracking area. The VR applications also need to be compatible with several headsets and controllers.

3. Reset the coordinate system

Let $\mathbf{v} \in \mathbb{R}^3$ be the upward vertical direction in the photogrammetric coordinate system. First, Sec. 3.1 defines a search space S for $\mathbf{v}$. Second, Sec. 3.2 estimates $\mathbf{v}$ up to its orientation, i.e. it estimates a $\epsilon\mathbf{v} \in S$ where $\epsilon \in \{-1, +1\}$. Finally, Sec. 3.3 finds the good orientation of $\mathbf{v}$ and Sec. 3.4 updates the coordinate system of the 3D model.

3.1. Search space for unoriented vertical direction

Thanks to the Structure-from-Motion step, the locations $\mathbf{l}_i \in \mathbb{R}^3$ of the camera during the image acquisition process are known in the photogrammetric coordinate system. 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. The search space S of $\epsilon\mathbf{v}$ is a subset of the unit sphere of $\mathbb{R}^3$: the unit vectors forming an angle with $\mathbf{v}_0$ that is less than a threshold α, i.e.

$$S = \{\mathbf{u} \in \mathbb{R}^3, \|\mathbf{u}\| = 1, \mathbf{v}_0^\top \mathbf{u} \geq \cos(\alpha)\}. \quad (1)$$

Since we would like to deal with ground surfaces with moderate slope angles, we need a large enough α. For example, if the ground is planar with a slope angle γ and if the camera is at a constant height above the ground (A1), then the camera motion is also planar with the same slope angle and we need $\alpha > \gamma$ to have $\epsilon\mathbf{v} \in S$. (See the left of Fig. 1.) Furthermore, $\alpha \leq \pi/2$ since a hemisphere contains all unoriented directions.

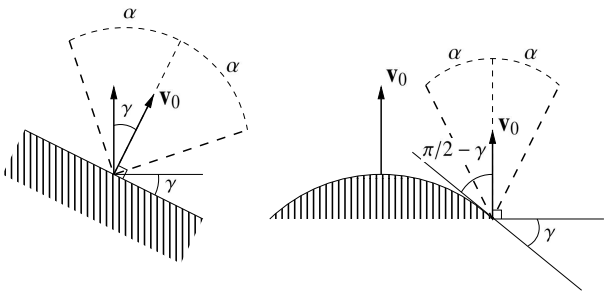


Figure 1: Ground surfaces and search spaces $S(\alpha, \mathbf{v}_0)$ for the unoriented vertical direction. Left: planar ground with slope angle γ. Right: non-planar ground with $\mathbf{v}_0 \approx \mathbf{v}$ and slope angle less than a threshold γ (everywhere on the ground). In both cases, the vertical in 3D is vertical in the figure.

3.2. Unoriented vertical direction

The vertical direction is computed by a 2D Hough transform. First, S is sampled in a finite set of vertical candidates. The sampling process is detailed latter (in Sec. 4) for purposes of clarity and readability. Then, every triangle of the surface votes for all vertical candidates that are almost parallel to the triangle. In other words, the triangle normal $\mathbf{n}$ and the vertical candidate are orthogonal (up to sampling). The vertical candidates of a triangle thus approximate a segment of a great circle $C_\mathbf{n}$ in the unit sphere, such that

$$C_\mathbf{n} = \{\mathbf{u} \in \mathbb{R}^3, \|\mathbf{u}\| = 1, \mathbf{n}^T\mathbf{u} = 0\}. \quad (2)$$

At first glance, the votes of each triangle are weighted by the triangle area such that the vote result is invariant by mesh subdivision. Nevertheless, this is not robust to the few bad triangles that have a very large area due to photogrammetry errors. We reduce this problem by damping the vote weight: a triangle with area A votes with weight $\min\{A, \beta A_\infty\}$ where $0 < \beta \leq 1$ and A_∞ is the largest area of the triangles. Finally, $\epsilon\mathbf{v}$ is the vertical candidate that maximizes the votes.

Here we explain how this method provides the expected result thanks to the anisotropy assumption (A2). On one hand, an important area of the surface is formed by vertical triangles. Each vertical triangle votes for a segment of

a great circle, and all these great circles have exactly two common points: the vertical $\mathbf{v}$ up to its orientation. (Since the i-th great circle is in a vertical plane $\{\mathbf{z} \in \mathbb{R}^3, \mathbf{n}_i^T \mathbf{z} = 0\}$ where $\mathbf{n}_i$ is horizontal and $\mathbf{v}$ is $\pm \mathbf{n}_1 \wedge \mathbf{n}_2$). Thus the votes near $\epsilon\mathbf{v}$ are high. On the other hand, the votes near the horizontal directions can also be high, as the ground has a large area too. If the ground is planar with a slope angle γ and normal $\mathbf{n}_\gamma$, every triangle in the ground votes for all directions that are orthogonal to $\mathbf{n}_\gamma$. There is therefore a risk of getting a bad $\epsilon\mathbf{v}$ if the sum of votes for a ground direction is greater than the sum of votes for the vertical. Summed votes for the vertical and a (horizontal) ground are also shown in Fig. 2. However, the angle between $\mathbf{v}_0$

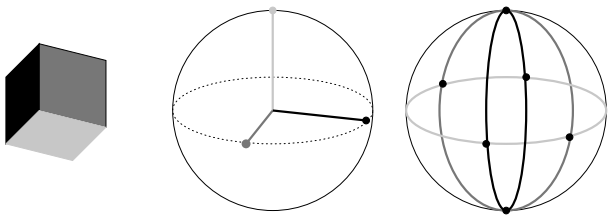


Figure 2: Horizontal and vertical votes in great circles. From left to right: a scene with one horizontal and two vertical rectangles (without plotting a triangle subdivision), their vertical or horizontal normals in the unit sphere (with dotted equator), their horizontal or vertical (voted) great circles in the unit sphere. Horizontal directions can have the largest votes.

and every ground direction is greater than the angle between $\mathbf{v}_0$ and the vertical, since $\mathbf{v}_0$ is approximately vertical and the ground is approximately horizontal. Thus there are values of α between these two angles, such that $S(\alpha, \mathbf{v}_0)$ is small enough to ignore the ground votes but also large enough to retain the vertical votes.

Fig. 1 details two base examples for setting α. If the ground is planar with slope angle γ, $\pi/2 > \alpha > \gamma$. If the ground is not planar, but $\mathbf{v}_0 \approx \mathbf{v}$ and the slope angle is less than a threshold γ everywhere on the ground, $\pi/2 - \gamma > \alpha > 0$. Thus

$$\pi/2 - \gamma > \alpha > \gamma. \tag{3}$$

We choose $\alpha = \pi/4$ to deal with plausible values $\gamma < \pi/4$.

3.3. Oriented vertical direction

Here we introduce additional assumptions to compute the oriented vertical from the unoriented vertical if a 360 camera is used

- **A3**: the surface reconstructed by photogrammetry is complete and closed: its triangles cover both ceiling (in a broad sense, i.e. foliage or tree canopy, sky, and so on) and ground.
- **A4**: we know whether the ground-to-camera distance, i.e. the height, is lower than the ceiling-to-camera distance (in most outdoor scenes, it will be).

A3 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 ground level with a 360 camera, 3D points are reconstructed all around the camera, and the camera is in the convex hull of the points. In this case, the surface reconstructed by standard methods based on 3D Delaunay triangulation and visibility constraints (as [Vu12, Lhuillier18]) is closed. This also implies that A4 is true for most outdoor scenes thanks to high-scene components such as buildings and trees.

Now $\epsilon\mathbf{v}$ is known but both ϵ and $\mathbf{v}$ are unknown. Thus we know the 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 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 of the distance between $\mathbf{l}_i$ and the intersection(s) between hl_i and every triangle in L_i (for all i). Thanks to A3, m_{-1} and m_{+1} are camera-to-ground or camera-to-ceiling distances, but we don't know which is which. According to A4, the smallest distance is camera-to-ground. We obtain $\mathbf{v} = \mathbf{v}_\epsilon$ such that ϵ meets $m_\epsilon < m_{-\epsilon}$.

There are alternatives to A3 and A4, e.g. if sky removal by the photogrammetry software cannot be disabled. We can assume that a main part of the ground is reconstructed and a main part of the sky is not, and then redefine m_ϵ by the sum (for all i) of the numbers of intersection(s) between hl_i and every triangle in L_i, to finally obtain $\mathbf{v} = \mathbf{v}_\epsilon$ such that ϵ meets $m_\epsilon > m_{-\epsilon}$.

3.4. Scale and change of the coordinate system

We assume that

- **A1'**: mean of the height of the camera is known

to obtain the scale. A1' is easy to meet for terrestrial setup: add an offset to the height of a person/vehicle taking the images.

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, which we get from A1'.

4. Sample the search space and vote

Sec. 4.1 provides details on the sampling induced by parameterizing S with a perspective camera. Sec. 4.2 adapts to S the sampling of [Lutton94] based on spherical coordinates.

4.1. Sampling using a perspective camera

We sample S by a parameterization that is defined by the backprojection of a perspective camera. This camera has an image I with $s \times s$ pixels. Its principal direction is $\mathbf{v}_0$, its field-of-view is 2α, its center is $\mathbf{0}$. Thus it projects $\mathbf{u} \in S$ to $p(\mathbf{u}) = \pi(\mathtt{K}\mathtt{R}^T\mathbf{u})$ where $\mathtt{R}$ is a rotation matrix whose third column is $\mathbf{v}_0$ and

$$\mathtt{K} = \begin{pmatrix} f & 0 & s/2 \\ 0 & f & s/2 \\ 0 & 0 & 1 \end{pmatrix}, \quad f = \frac{s}{2\tan\alpha}, \quad \pi(\begin{pmatrix} c_1 \\ c_2 \\ c_3 \end{pmatrix}) = \begin{pmatrix} c_1/c_3 \\ c_2/c_3 \end{pmatrix}. \quad (4)$$

Furthermore, $p(S)$ is the disc inscribed in I and $p(C_\mathbf{n})$ is the projection of the plane $\{\mathbf{z} \in \mathbb{R}^3, \mathbf{n}^T\mathbf{z} = 0\}$, i.e. the line

$$P^0_\mathbf{n} = \{\begin{pmatrix} x \\ y \end{pmatrix} \in I, \begin{pmatrix} x & y & 1 \end{pmatrix}\mathbf{l} = 0\} \text{ where } \mathbf{l} = \mathtt{K}^{-T}\mathtt{R}^T\mathbf{n}. \quad (5)$$

The triangle with normal $\mathbf{n}$ adds its vote to the accumulator of every pixel that discretizes the line $P^0_\mathbf{n}$. Voting for a pixel (x, y) is voting for a vertical candidate $\mathbf{u}$ obtained by backprojection:

$$\mathbf{u} = n(\mathtt{R}\mathtt{K}^{-1}\begin{pmatrix} x \\ y \\ 1 \end{pmatrix}) \text{ where } n(\begin{pmatrix} u \\ v \\ w \end{pmatrix}) = \frac{1}{\sqrt{u^2+v^2+w^2}}\begin{pmatrix} u \\ v \\ w \end{pmatrix}. \quad (6)$$

After all votes have been cast but before computing the maximizer, we reduce the noise by smoothing I using the 3×3 Gaussian kernel, i.e. $(1/16)\begin{pmatrix}1 & 2 & 1\end{pmatrix} \otimes \begin{pmatrix}1 & 2 & 1\end{pmatrix}$.

Now we examine consequences in S of the voting in I. A pixel in I is a 1×1 square that backprojects to a surface element in S whose area is $\|\frac{\partial\mathbf{u}}{\partial x} \wedge \frac{\partial\mathbf{u}}{\partial y}\|$. There is only one vertical candidate $\mathbf{u}$ for each surface element. Furthermore, its area depends on x and y (with integer values in $[0, s]$ for pixels). Appendix A shows that

$$\|\frac{\partial\mathbf{u}}{\partial x} \wedge \frac{\partial\mathbf{u}}{\partial y}\| = \frac{f}{((x - s/2)^2 + (y - s/2)^2 + f^2)^{3/2}} \quad (7)$$

and we see that the pixels near I's borders have smaller surface elements than the pixels near I's center. This implies that the sampling distribution of the vertical candidates is not uniform in S. The expected accuracy of the estimated vertical thus depends on the location of its pixel. Nevertheless, the area range of the surface elements (and thus the accuracy range) is moderated in our application for usual values $\alpha \leq \pi/4$. Indeed, for all pixels in $p(S)$,

$$\frac{1}{f^2} \geq \|\frac{\partial\mathbf{u}}{\partial x} \wedge \frac{\partial\mathbf{u}}{\partial y}\| \geq \frac{1}{f^2}(1 + \tan^2\alpha)^{-3/2} \geq \frac{1}{2\sqrt{2}f^2}. \quad (8)$$

4.2. Sampling using spherical coordinates

Here we parameterize S by using spherical coordinates:

$$\mathbf{u} = \mathtt{R}\begin{pmatrix}\sin\phi\cos\theta & \sin\phi\sin\theta & \cos\phi\end{pmatrix}^T \quad (9)$$

where $\theta \in [0, 2\pi]$, $\phi \in [0, \alpha]$ and $\mathtt{R}$ is defined in Sec. 4.1. We also use the sampling developed by [Lutton94]: using a regular quantization in ϕ and an irregular quantization in θ such that a parameterization cell $\Delta\phi \times \Delta\theta$ has a surface element in S with constant area $\Delta S = \Delta\theta\Delta\phi\sin\phi$.

First, we go onto detail. Let $N \in \mathbb{N}^*$ be the number of ϕ quantizations. We have $\Delta\phi = \alpha/N$ and $\phi_k = k\Delta\phi$ where $k \in \{1, 2, \cdots N\}$. Furthermore, the θ quantization depends on k since $\sin\phi_k\Delta\theta_k$ is constant. The number n_k of cells in the k-th layer is

$$n_k = \frac{2\pi}{\Delta\theta_k} = integer(\frac{2\pi\sin(k\alpha/N)}{\Delta\theta_N\sin\alpha}. \quad (10)$$

For each k, the θ quantization is obtained by $\theta^k_0 = 0$ and $\theta^k_{j+1} = \theta^k_j + \Delta\theta_k$ where $j \in \{0, 1, \cdots, n_k - 1\}$.

Then we set N and n_k to obtain a segmentation of S into cells that is "similar" to that in Sec. 4.1. This is useful in order to obtain fair comparisons in the experiment. We would like a similar number of cells in the border of S in both cases: n_N is also the circumference (measured in pixels) of the disc $p(S)$ with diameter s, i.e.

$$n_N = integer(\pi s). \quad (11)$$

Thanks to Eqs. 10 and 11 and by removing the integer truncation, we obtain $\Delta\theta_N \approx 2/s$. We also would like a similar number of cells in S in both cases:

$$\frac{\pi s^2}{4} \approx \sum_{k=1}^{N} n_k \approx \sum_{k=1}^{N} \frac{2\pi \sin(k\alpha/N)}{(2/s)\sin\alpha}. \quad (12)$$

Now we multiply Eq. 12 by $(\alpha \sin\alpha)/(N\pi s)$:

$$\frac{\alpha}{N}\frac{s\sin\alpha}{4} \approx \frac{\alpha}{N}\sum_{k=1}^{N} \sin(k\frac{\alpha}{N}) \approx \int_0^{\alpha} \sin(x)dx. \quad (13)$$

We obtain

$$N = integer(\frac{s\alpha\sin\alpha}{4(1-\cos\alpha)}). \quad (14)$$

Thus we set N and n_N from s and α using Eqs. 14 and 11, then obtain n_k using Eq. 10 (with $\Delta\theta_N = 2\pi/n_N$) if $k < N$.

In a final substep, we detail the voting of triangle with normal $\mathbf{n}$. Let $(\phi_\mathbf{n}, \theta_\mathbf{n})$ be the spherical coordinates of $\mathbf{n}$ using Eq. 9. We assume that $\phi_\mathbf{n} \leq \pi/2$ by multiplying $\mathbf{n}$ by -1 if needed. (This does not change $C_\mathbf{n}$.) The spherical coordinates (ϕ, θ) of $\mathbf{u} \in C_\mathbf{n}$ (using Eq. 9) meet

$$0 = \mathbf{n}^T\mathbf{u} = \cos\phi\cos\phi_\mathbf{n} + \sin\phi\sin\phi_\mathbf{n}\cos(\theta - \theta_\mathbf{n}). \quad (15)$$

We obtain a parameterization of $C_\mathbf{n}$:

$$\theta(\phi) = \theta_\mathbf{n} \pm \text{acos}(-\frac{\cos\phi}{\sin\phi_\mathbf{n}}\frac{\cos\phi_\mathbf{n}}{\sin\phi}), \frac{\pi}{2} - \phi_\mathbf{n} \leq \phi < \frac{\pi}{2}. \quad (16)$$

The triangle with normal $\mathbf{n}$ votes for every cell $[\phi_{k-1}, \phi_k] \times [\theta^k_j, \theta^k_{j+1}]$ that includes a point $(\phi, \theta(\phi)+$ modulo $2\pi)$. After all votes have been cast but before computing the maximizer, we reduce the noise using a smoothing process that is an adaptation of the 3×3 Gaussian kernel: first apply the smoothing $(1/4)\begin{pmatrix}1 & 2 & 1\end{pmatrix}$ to the votes with same ϕ, then to the votes with similar θ.

5. Experiments

5.1. Notations for experimented methods

There are four variants of the Hough transform:

- H1: Sec. 4.1 without smoothing ([Lhuillier21])
- H1': Sec. 4.1 with smoothing
- H2: Sec. 4.2 (i.e. like [Lutton94]) without smoothing
- H2': Sec. 4.2 with smoothing.

We use $\alpha = \pi/4$ for the search space S (Eq. 1) and $s = 100$ for the resolution of S as in [Lhuillier21]. A pixel of image I (Sec. 4.1) thus has a backprojected cone of about $0.9°$ by $0.9°$. This is enough for the VR application, since the human vestibular system detects the vertical direction with an accuracy of about $1.5°$ ([Lobo03]). We also use $\beta = 0.1$ (Sec. 3.2).

Other methods estimate a vertical direction from normals:

- R: the RANSAC method developed by [Liu22] with an additional and rough approximation $\mathbf{v}_0$ of the vertical
- BnB: the Branch-and-Bound method developed by [Liu22] (with the exponential mapping)
- Urban: the gravity vector estimation of [Avidar18]
- PCA2: a robust version of a method developed by [Alnuaimi17].

The methods of [Liu22, Alnuaimi17, Avidar18] are briefly summarized in Sec. 2.2. The angular tolerance of R and BnB is $\tau = 1°$ like [Liu22]. BnB is the only method that does not use $\mathbf{v}_0$. Here we improve R by using $\mathbf{v}_0$ to avoid spurious verticals. (For each random sample normals $\mathbf{n}_1$ and $\mathbf{n}_2$, the vertical candidate that counts inliers is the vector among $\{\mathbf{n}_1, \mathbf{n}_2, (\mathbf{n}_1 \wedge \mathbf{n}_2)/\|\mathbf{n}_1 \wedge \mathbf{n}_2\|\}$ whose angle with $\pm\mathbf{v}_0$ is the smallest.) PCA2 first selects normals that are parallel to $\mathbf{v}_0$ up to a tolerance error ($15°$), then computes the vertical as the first principal component of the selected normals.

5.2. Experiments using a synthetic dataset

We first compare accuracies of unoriented verticals estimated by eight methods (Sec. 5.1) for a synthetic environment.

5.2.1. Dataset

The synthetic environment is implicitly defined by

$$\{\begin{pmatrix}x & y & z\end{pmatrix}^T \in \mathbb{R}^3, x^{2p} + y^{2p} + (z - x\tan\gamma)^{2p} = 1\}, \quad (17)$$

with $p \in \mathbb{N}^*$ and $\gamma \in [0, \pi/2]$. It approximates the border of the cube $[-1, +1]^3$ for $\gamma = 0$ and a large enough p. If $\gamma \neq 0$, we obtain a smoothed version of cube whose up-face and down-face (near planes $z = \pm 1$) are slanted: their slope angle is about γ with (unoriented) normal

$$\mathbf{n}_\gamma = \begin{pmatrix} -\sin\gamma & 0 & \cos\gamma \end{pmatrix}^T. \tag{18}$$

One of these two faces plays the role of a slanted ground and four others are approximately vertical and orthogonal walls. The search space S (Eq. 1) meets $\mathbf{v}_0 = \mathbf{n}_\gamma$, as if the rough estimate $\mathbf{v}_0$ of the vertical is computed by PCA of a camera motion whose height is constant (A1). The ground truth of the vertical is $(0\ 0\ 1)^T$. The triangulated surface is defined by a regular sampling of the spherical coordinates (60 longitudes, 30 latitudes plus two poles). Furthermore, we disrupt this surface using a noise δ: add to every coordinate of every vertex an uniformly distributed random value in $[-\delta, +\delta]$.

5.2.2. Experiments #1: comparisons between Hough variants

The top of Tab. 1 shows accuracies of the Hough variants (Sec. 5.1) for the synthetic environment using $p = 3$, $\gamma = 5°$ and several values of δ. As expected, the lower the noise δ, the better the accuracy. Note that the smoothing in H1' and H2' greatly improves the accuracy of H1 and H2, respectively. We also see that the methods based on spherical coordinates (H2-H2') are less accurate than the methods based on perspective camera (H1-H1'). The parameterization of S by a perspective camera introduces a bias in favor of the vertical candidates close to $\mathbf{v}_0$, i.e. that correspond to pixels near the center of image I. Since these pixels have greater surface elements in S than the pixels near the border of I (Eq. 7), the pixels near the center are more likely to be voted by an uniform distribution of triangle normals than the pixels near the border. In short, the bias toward $\mathbf{v}_0$ is beneficial to estimate the vertical direction since $\mathbf{v}_0$ is a rough value of the vertical.

Fig. 3 shows the votes in the Hough space for all variants and for two noises. There are two dark strips in all cases. Each strip corresponds to the votes of the great circles of a pair of vertical and parallel faces of the synthetic environment. Strip thickness increases by noise, and noise perturbs the vote maximizer in the intersection of the strips.

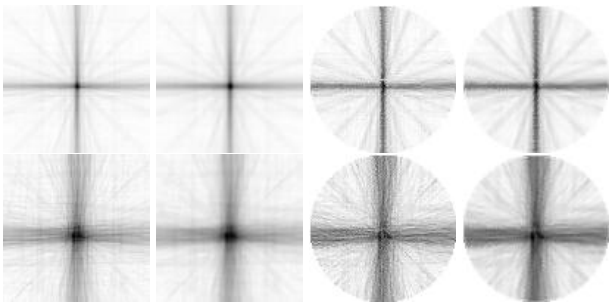


Figure 3: Votes in the Hough space for the synthetic experiments. From left to right: Hough variants H1, H1', H2, and H2'. There are two noises: $\delta = 0.003$ (top) and $\delta = 0.01$ (bottom). In the perspective cases H1-H1', votes for image I are shown in greyscale. The darker the gray level, the greater the vote. In the spherical cases H2-H2', the votes are mapped to I for visual comparison only.

5.2.3. Experiments #2: comparisons with other methods

We also estimate the vertical direction using other methods (bottom of Tab. 1). BnB has the worst results due to a spurious solution: all normals are inlier of $(0\ 1\ 0)^T$ if we neglect noise. Indeed, the spurious solution $(0\ 1\ 0)^T$ is parallel to the slanted ground (with normal $\mathbf{n}_\gamma$) and parallel or orthogonal to the four vertical faces (with normals $(1\ 0\ 0)^T$ or $(0\ 1\ 0)^T$) of the synthetic environment. R does not have this problem thanks to $\mathbf{v}_0$. Urban is more accurate than R, but is less accurate than the Hough transforms. PCA2 has nearly a constant error $\gamma = 5°$. This is not surprising since it computes the vertical as a normal of the ground.

5.3. Experiments using a photogrammetric dataset

Here we provide detailed experiments on two 3D models reconstructed from videos taken by helmet-held 360 cameras: a tourist-attraction city build on an inclined ground and a basalt canyon with a strongly non-planar ground. The experiments include comparisons of vertical directions computed by our method against other vertical directions computed by previous methods or different sensors.

5.3.1. Dataset (Basalt and City)

The 3D model "Basalt" is reconstructed from a 360 video taken using a helmet-mounted GoPro Max camera while walking for 27 minutes around a geological-interest site. (See Fig. 4.) The two sides of the path are

δ	0.002	0.004	0.006	0.008	0.010
n_δ	1.15	2.31	3.48	4.67	5.84
H1	0.44/0.06/0.71	0.63/0.29/1.56	1.00/0.52/2.92	1.40/0.73/4.37	1.70/0.89/4.82
H1'	0.42/0.00/0.42	**0.48**/0.13/1.34	**0.69**/0.36/2.16	**1.04**/0.56/2.94	**1.43**/0.78/4.02
H2	0.70/0.44/2.24	1.32/0.63/3.61	1.93/0.89/6.02	2.60/1.06/6.40	3.07/1.25/7.39
H2'	**0.41**/0.23/1.28	0.84/0.44/2.36	1.48/0.66/3.62	2.13/0.89/6.40	2.65/1.06/7.05
R	2.93/14.9/88.9	3.74/16.1/88.4	7.30/22.3/89.3	8.08/23.0/89.8	12.3/28.1/89.9
BnB	89.9/0.11/89.9	89.3/1.04/89.9	88.2/1.79/89.9	87.4/5.83/90	87.2/4.31/89.9
Urban	0.46/0.19/1.26	2.87/11.6/90.0	5.76/16.3/90.0	7.60/19.0/90.0	10.3/22.0/90.0
PCA2	5.20/0.02/5.26	5.20/0.06/5.37	5.19/0.10/5.50	5.17/0.14/5.59	5.15/0.17/5.69

Table 1: Accuracy of the (unoriented) vertical for the synthetic environment. For each method and noise, we have a tuple of values: mean, standard deviation and maximum of angles between estimated and ground-truth verticals (computed from 500 estimates). Also see n_δ: the noise of triangle normals (the mean of angles between triangle normals with noise δ and triangle normals without noise). All values, except δ, are angles in degrees.

Figure 4: A Gopro Max 360 camera and a cubemap image.

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 2: Characteristics of Basalt and City: number of triangles, number of camera locations $\mathbf{l}_i$ (selected by Structure-from-Motion), camera-to-ground distance (i.e. camera height) h, trajectory length, slope-length (e.g. at least 10% slope along 240m of the acquisition trajectory).

composed of basalt prisms that range from 5m to 20m in height and that have been eroded and degraded by vegetation and high humidity. The texture is favorable for photogrammetry, except at a few locations with low light due to narrow canyon (less than 1m) or where the vegetation is very close to the camera. 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 sliding on the buttocks. This path was thoughtfully selected before the acquisition as it offers a complicated test (a kind of labyrinth with obstacles and dangerous sections) and as the trajectory has to include closed loops for the Structure-from-Motion step to reduce the drift. (A small drift is useful for A0.)

The 3D model "City" is reconstructed from 360 videos taken using two GoPro Max cameras (only one fisheye per GoPro is useful) rigidly mounted on either side of the head (at ear level) while walking for 21 minutes around the streets of a local city (Collonges-la-Rouge). This model is interesting for experimenting our method on a more standard ground surface with a moderated slope. It includes small castles, a church, trees and medieval porches. The path was chosen before acquisition using a map to ensure that it includes several closed loops. Both day and hour of acquisition were purpose-chosen to reduce the number of tourists in the streets. As in the first model, the weather was chosen to ensure there was enough light and to ensure that surrounding vegetation stayed still.

Tab. 2 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 and feature millions of triangles and non-negligible slope angles. They are reconstructed using the work of [Lhuillier18] and [Lhuillier19]. Fig. 5 illustrates global and top-down views of reconstructed points.

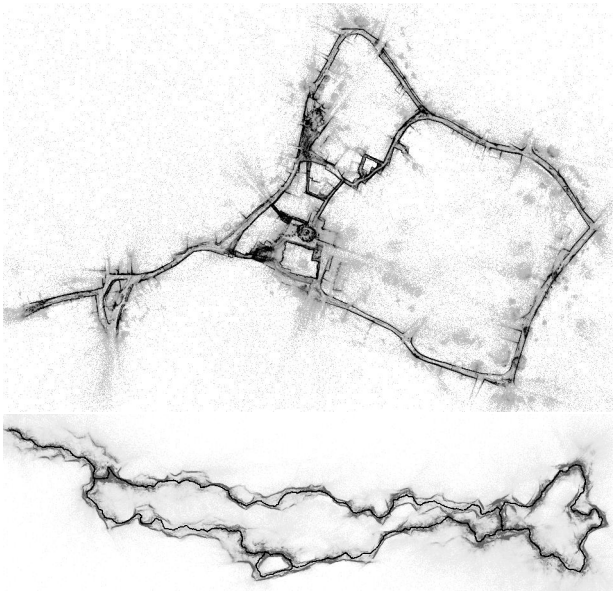
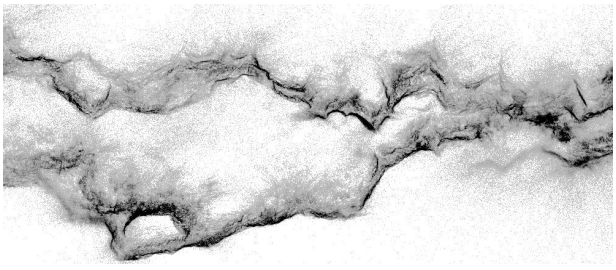
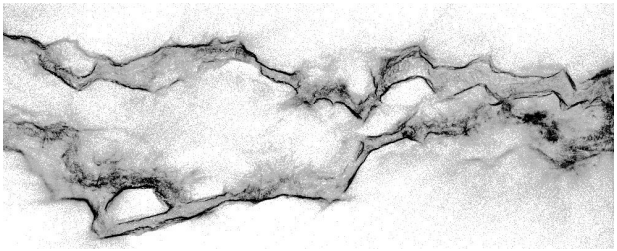


Figure 5: Global and top-down views of the reconstructed points of City (top) and Basalt (bottom). The darker the gray level, the greater the point density. The trajectory of the acquisition camera is also shown in black for Basalt.



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



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

Figure 6: Top-down views of reconstructed points of Basalt. The direction estimated by the Hough transform is more parallel to the canyon sides than the direction estimated by PCA.

5.3.2. Experiments #1: is the Hough step really useful ?

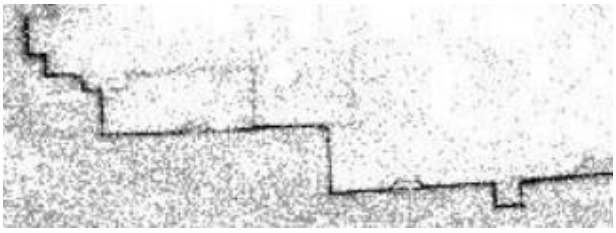
Here we compare methods that compute the vertical direction up to its orientation, i.e. the PCA method in Sec. 3.1 and the Hough method in Sec. 3.2 (H1'). We remind that the former initializes the latter, and the former is enough for an acquisition trajectory that is almost horizontal. Their estimates are $\mathbf{v}_0$ and $\epsilon\mathbf{v}$, respectively.

First 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 have to be projected into lines or curves. Fig. 6 shows the projections for Basalt, which is composed of corridors (or canyons) whose sides are mostly vertical. The direction estimated by the Hough transform is more parallel to the corridor sides than the direction estimated by PCA, since new dark curves appear with $\epsilon\mathbf{v}$. The comparison is less easy for City because we must to zoom in to see a difference between PCA and Hough results (Fig. 7). We see that 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 lower the thickness, the more accurate the vertical. We thus arrive at the same conclusion for both 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 14° for Basalt and 4° for City. This explains why this comparison is easier to make for Basalt than for City.

Second we run a further comparison to show that $\epsilon\mathbf{v}$ is better than $\mathbf{v}_0$. In Fig. 8, each image of the City's 3D model is drawn by a pinhole camera ([Hartley00]) that has zero pitch and zero roll angles (and square pixels) with respect to a vertical direction of reference among $\mathbf{v}_0$ and $\epsilon\mathbf{v}$. Then all vertical lines in 3D are projected to vertical lines in 2D, if the vertical direction of reference is an accurate estimation of the true vertical. They are if the vertical direction of reference is $\epsilon\mathbf{v}$, but they are not if this direction is $\mathbf{v}_0$.

Third we discuss the votes in the Hough space of Basalt and City thanks to Fig. 9. The dark spots suggest that the computation of $\epsilon\mathbf{v}$ is more reliable in City than in Basalt. This confirms the observation that the natural (Basalt)

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



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

Figure 7: Top-down views of reconstructed points of City. The direction estimated by the Hough transform is more parallel to walls and facades than the direction estimated by PCA.

Figure 8: Views of City 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, if the reference vertical direction is $\mathbf{v}_0$ (top).

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 3: Mean and standard deviations of ground-to-camera and ceiling-to-camera distances in the photogrammetric coordinate system.

walls are less vertical than the man-made (City) walls.

5.3.3. Experiments #2: orientation and scale

Now we use Tab. 3 to detail the estimations of the orientation and scale in Sec. 3.3. Again, note that we distinguish camera-to-ground and camera-to-ceiling distances in $\{m_{-1}, m_{+1}\}$ by comparing m_{-1} and m_{+1}. Since $m_{+1} < m_{-1}$ for both City and Basalt, m_{+1} is the camera-to-ground distance and the 3D models will be rescaled to make m_{+1} equal to h. The table also provides the standard deviation σ_ϵ associated to m_ϵ. We see that the assumption of constant camera height (A1) is better for City than for Basalt, which is confirmed by observation: the person 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).

5.3.4. Experiments #3: other methods

We compare H1' with other methods. Let $a(M)$ be the angle in degrees between the vertical estimated by H1' and the vertical estimated by another method M. For City, $a(R) = 0.723$, $a(BnB) = 0.509$, $a(Urban) = 0.413$ and $a(PCA2) = 4.547$. For Basalt, $a(R) = 3.84$, $a(BnB) = 5.49$, $a(Urban) = 7.52$, and $a(PCA2) = 15.9$. H1' takes less than 3 seconds with one core of a Laptop. Its time complexity is $O(sN + s^2)$ where s is the Hough resolution and N is the number of input triangles. BnB is the slowest method: 25 minutes for City and 102 minutes for Basalt. Its runtime not only depends on N but also on the shape complexity of its score function. Since all methods (except PCA2 which computes a normal of the ground) provide similar results for City, we are more confident in these results than in the results for Basalt.

5.3.5. Experiments #4: toward limits of our method

Here we compare the methods based on transformations of the dataset to make it more difficult to estimate the

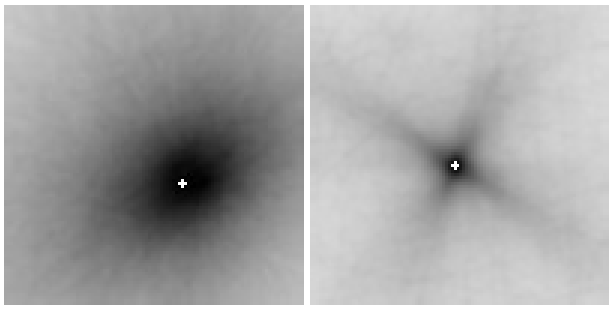


Figure 9: Hough space of Basalt (left) and City (right) using H1' with $\alpha = \pi/4$ and $s = 100$. The white cross shows $\epsilon\mathbf{v}$. The center of the space is $\mathbf{v}_0$.

unoriented vertical direction than in Sec. 5.3.2. Let CityR be a set of 100 sub-models of City with R a real number. A sub-model is the part of City (both triangles and trajectory) inside a sphere whose radius is R and whose center is a camera location $\mathbf{l}_i$. (i is evenly spread in its range.) We define several datasets, as follows: City20, City50, City100, Basalt50, Basalt100 and Basalt200.

According to Tab. 4, for each scene and each method, the rule is: the smaller the radius of a dataset, the greater the degradation of the estimated verticals. Degradation is measured by the angle between a vertical estimated for a sub-model and the vertical estimated by PCA+H1' for a complete model. For our PCA+H1' experiment, the rule can be explained as follows. If PCA is applied on a camera trajectory that is a line segment, the rough estimate $\mathbf{v}_0$ of the vertical can be every vector that is orthogonal to the segment. Thus the search space can exclude the target vertical and H1' fails. Other failures are due to scenes whose vertical components have a single unoriented normal, e.g. a straight road with only walls parallel to the road. In this case, the maximizer of the votes is not well-defined, and the vertical estimation can be every vector that is orthogonal to this normal. The probability of such failures increases as sphere radius decreases.

For the Hough methods, Tab. 4 shows that the degradation is reduced by smoothing or if the sampling of the search space is based on a perspective camera. Furthermore, the degradation is greater for the natural environment (Basalt) than for the man-made environment (City). H1' is the best for City. BnB has the worst results for City but also has the most best results for Basalt. The results for City have similarities with the synthetic experiments (City can be locally approximated by a slanted ground plane and vertical-orthogonal walls). The results for Basalt are due to the fact that Basalt has a different structure to City, i.e. the ground is strongly nonplanar and the canyon sides are not limited to two orthogonal vertical walls. PCA+R gives some of the best results for both Basalt (since it has the same score function as BnB) and City (thanks to our $\mathbf{v}_0$). Nevertheless, vertical estimates of small 3D models can be too inaccurate for a visualization application. Here a degradation of 4° is not acceptable for City according to Fig. 8.

5.3.6. Experiments #5: comparisons with other sensors

Basalt and City are reconstructed from images taken by a 360 camera, whose firmware also estimates a vertical direction for each image. These estimates are produced using measurements coming from built-in sensors detailed by [Kirschenbaum19]: accelerometer, gyroscope, GPS, magnetometer. They are used by applications like horizon leveling and image stabilization. Here we aim to find out the discrepancy between verticals estimated by our method versus the firmware.

Nevertheless, there are two problems. First, the estimations of the vertical direction are not in a same coordinate system. Since an estimation done by the firmware is in a camera coordinate system F_m chosen by the manufacturer, we need to compute the change of coordinate system toward the coordinate system F_g of the 3D model. This is the composition of two rotations: between F_m and the camera coordinate system F_p of the photogrammetry software, and between F_p and F_g. The latter depends on the image (It is computed by Structure-from-Motion.) and the former is constant. Second, the vertical estimation by the firmware is not unique after the coordinate change for several reasons ([Lobo03]) including: drift error due to time integration of inertial measurements, a heuristic to separate gravity and linear acceleration, incremental computations. Therefore, here we do not report a single angle discrepancy a between two estimated verticals, but the mean $\bar{a}$ and standard deviation σ_a and maximum $\max a$ of angle discrepancy.

The angle discrepancies, in degrees, between the firmware verticals and our vertical (using H1') are as follows. For City, the first Gopro has $\bar{a} = 2.24$, $\sigma_a = 0.85$, $\max a = 4.39$ and the second Gopro has $\bar{a} = 1.29$,

Dataset	City20	City50	City100	Basalt50	Basalt100	Basalt200
PCA+H1	6.20/14.1/81.1	1.34/4.33/43.8	0.86/0.28/1.86	14.3/22.0/104.	8.97/14.0/104.	2.00/2.96/8.29
PCA+H1'	**3.77**/11.5/81.1	**0.75**/0.44/2.49	**0.71**/0.24/1.36	13.4/20.3/104.	8.90/16.3/120.	**1.63**/2.74/8.68
PCA+H2	12.9/14.9/71.3	6.66/13.1/50.3	3.60/7.99/39.5	14.7/20.7/119.	10.4/7.90/42.3	13.3/5.87/37.7
PCA+H2'	9.22/12.2/51.6	3.20/7.82/43.4	1.65/2.49/19.0	13.2/18.3/109.	11.3/7.49/32.7	8.83/2.09/18.7
PCA+R	6.15/19.7/92.9	4.38/17.3/91.3	1.56/8.97/90.7	6.31/10.6/77.7	4.71/2.90/22.3	4.61/1.34/7.79
BnB	71.7/35.6/94.8	46.1/45.7/97.2	27.8/41.9/94.9	**4.31**/2.70/9.97	**4.27**/1.45/6.35	5.52/0.41/6.35
PCA+Urban	21.9/20.6/87.6	14.8/18.2/77.2	7.45/11.1/56	32.0/26.7/111.	25.0/30.5/123.	11.0/8.44/47.6
PCA+PCA2	6.02/4.85/26.7	4.55/2.66/17.7	4.06/1.45/10.5	26.4/32.0/136.	20.4/29.8/137.	17.6/4.76/33.8

Table 4: Degradation of the vertical direction as a function of radius of sub-models in City and Basalt. For each model (City or Basalt) and radius, we have a tuple of values, i.e. the mean, standard deviation and maximum of angles (in degrees) between estimated and reference verticals.

$\sigma_a = 0.60$, $\max a = 3.54$ (remember that two Gopro Max are used for City.) For Basalt, $\bar{a} = 3.94$, $\sigma_a = 2.16$, $\max a = 11.4$. We also examine how far $\bar{a}$ can be minimized by updating the rotation between F_m and F_p: with rotation changes of about 2 degrees, we obtain $\bar{a} = 1.45$, $\bar{a} = 1.16$, $\bar{a} = 3.64$, respectively, for the three above cases.

5.3.7. Discussion on parameter tuning

The estimation of the vertical direction depends on α, β and s, which are described in Sections 3.1, 3.2 and 5.1. Note, again, that the value of $\beta \in]0, 1]$ is a trade-off: it must be large enough to get invariant votes by subdivision of the input triangulation, but also small enough to get robustness with respect to the large spurious triangles. In our experiments on photogrammetric 3D models, $\beta = 0.1$. If photogrammetry software removes spurious triangles, the invariance can be improved by increasing β.

Furthermore, α is an upper bound for the angle between the rough-initialized vertical (using PCA) and the refined vertical (using Hough transform). It is also an upper bound of the slope angle of the ground surface. In our experiments, $\alpha = \pi/4$ to deal with a lot of ground surfaces including the disrupted Basalt ground surface. If we know a tight upper bound of the slope angle, e.g. 10 degrees for usual environments, then we can set α with this value. Then the Hough space is reduced, which in turn implies a decreased risk of a bad maximiser.

Finally, note again that s is the image size (i.e. the resolution of the Hough space) that needs to be large enough to provide an accurate vertical, but not be too large, otherwise the votes for the true vertical would get spread across several pixels (instead of a single one in the ideal case) due to noise, which in turn would corrupt the vote maximizer and thus the computation of the vertical. Here $s = 100$ for an accurate-enough vertical for our VR application. The noise is that of triangle's normals, which are computed from the reconstructed 3D points. The lower the reconstruction noise, the greater the possible values of s.

5.4. Experiments using a scanner dataset

Although scanner input is not within the main scope of the paper, it is interesting to experiment with it and compare our method against previous ones that were introduced purposely for this input.

5.4.1. Dataset (Bremen and Kapelle)

Two point clouds ([Osnabruck]), Bremen and Kapelle, are taken by a scanner (Riegl VZ400) and shown in Fig. 10. Since they have high numbers of points (about 200M) without normals, we first downsample them and then compute normals (by PCA of 6 nearest points) in a similar way to [Liu22]. Bremen is an urban environment that meets the Atlanta world assumption. It has 1.8M normals and a vertical of about $\mathbf{g}^T = \begin{pmatrix} 0 & 0 & 1 \end{pmatrix}$. Kapelle is a mixture of natural and man-made environments: a small chapel surrounded by trees, cultures and a town in the background. It has 1.6M normals and a vertical of about $\mathbf{g}^T = \begin{pmatrix} 0 & 1 & 0 \end{pmatrix}$.

In this context, the H1' method is limited to the computations in Sec. 3.2 and it takes these normals that share a same vote weight directly as input, as in [Liu22].

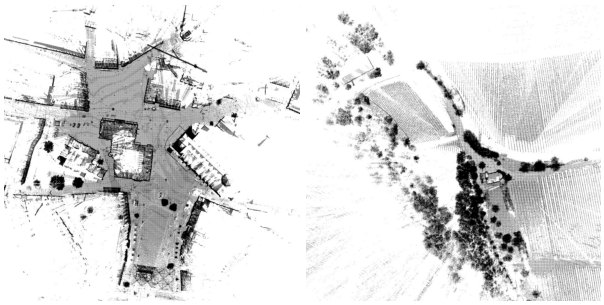


Figure 10: Top-down views of the scanner dataset: Bremen center (left) and Maria-Schmerz-Kapelle Randersacker (right).

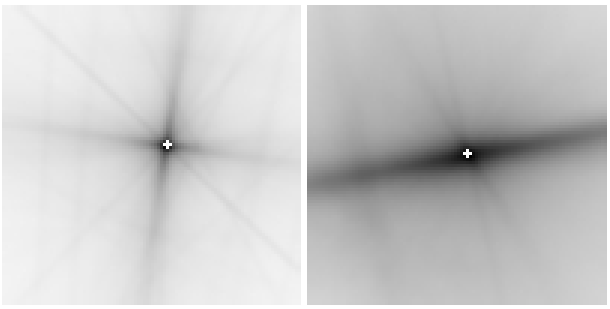


Figure 11: Hough space of Bremen30 (left) and Kapelle5 (right) using H1'.

Dataset	Bremen30	Kapelle30	Kapelle5
H1'	**0.62**/0.26/1.10	**2.82**/0.32/3.43	2.90/0.34/3.50
R	0.86/0/0.86	5.40/10.9/81.8	3.84/0/3.84
BnB	0.74	80.7	80.7
Urban	0.69/0.25/1.47	4.36/9.58/75.5	2.90/0.32/3.74
PCA2	2.92/2.65/12.4	9.02/3.16/17.0	**2.04**/0.91/3.61

Table 5: Discrepancies in vertical direction on the scanner dataset using noisy $\mathbf{v}_0$. For each model (Bremen or Kapelle) and noise, we have a value or a tuple of mean/standard deviation/maximum of angles (in degrees) between estimated and reference verticals.

Name	Acquisition	360 camera	#tri
Snowy town	walk 670m	Garmin Virb	697k
Forest	walk 800m	Garmin Virb	2.1M
Semi-medieval town	walk 1km	Gopro Max	2.3M
Campus	bike 5.2km	Garmin Virb	2.8M
Verdon canyon	kayak 2.5km	Gopro Max	1.8M
Ruin in forest	walk 400m	Gopro Max	1.7M

Table 6: Characteristics of six other input 3D models, including acquisition information (camera name and approximate trajectory length) and number of surface triangles.

5.4.2. Experiments

Tab. 5 compares the vertical directions estimated by 100 runs of each method applied on the scanner dataset with noisy initialization $\mathbf{v}_0$. The Bremen30 notation (and likewise Kapelle30 and Kapelle5) means that we experiment on Bremen with a noise such that the angle between $\mathbf{v}_0$ and $\mathbf{g}$ is less than 30°. For Bremen30, all verticals (except that of PCA2) are similar to $\mathbf{g}$ with discrepancies below 1° in most cases. H1' gives some of the best results among methods R-BnB-Urban methods that are purpose-designed for this kind of dataset. For Kapelle, a small-enough noise of $\mathbf{v}_0$ is needed to obtain discrepancies of about $2° - 3°$ (except BnB which fails). Fig. 11 shows the votes of H1' for Bremen30 and Kapelle5 using $\alpha = \pi/4$ and $s = 100$ (β is unused).

5.5. VR for the extended photogrammetric dataset

Sec. 5.5.1 summarizes the extended photogrammetric dataset, which includes Basalt and City, and its VR applications (viewers). Sec. 5.5.2 focuses on a 3D model of this dataset that does not meet the assumption A2. The other sections provide informations describing the VR applications, which are useful for people who want to use VR to display photogrammetric reconstructions: common VR headsets (Sec. 5.5.3), compromise between rendering quality and framerate (Sec. 5.5.4), Locomotion (Sec. 5.5.5).

5.5.1. Overview of the input and the results

VR applications are available for eight 3D models: Basalt and City in Sec. 5.3 and six others that are summarized in Tab. 6. The 3D models are reconstructed from videos taken by helmet-held 360 cameras by [Lhuillier18] and [Lhuillier19] followed by an incremental mesh simplification based on edge collapses ([Botsch10]). A VR application is a viewer for a single 3D model (ranging in size between 169Mo and 800Mo). The VR user can choose the rendering (textures or normals) and the motion (2D like a pedestrian, 3D like a bird).

Figs. 12 and 13 show screenshots taken by an Oculus Quest running the VR applications of Basalt and City,

after our reset of their coordinate systems (Sec. 3). The outward-pointing normal direction is color-encoded: white for upward vertical, black for downward vertical, red-green-blue for horizontal. Fig. 14 shows two screenshots, the Hough space (H1') and a top view of reconstructed points for the six other 3D models. Note that there is no manual cleanup, since a secondary objective here is to show the quality of the reconstruction.

5.5.2. Special case: the Verdon canyon

The Verdon canyon stands out a special case among the eight 3D models for several reasons. First the second step (Hough) of our method fails, since the assumption A2 is not met. The two sides (walls) of this canyon are too far from vertical, and so the unoriented vertical direction is provided by the first step (PCA). Second the reconstruction of the ground is inaccurate, since the ground is water, which in turns means that ground-to-camera distance is inaccurate too. Since the ceiling-to-camera distance is two magnitude orders greater than the ground-to-camera distance, the oriented vertical direction is good. But the final scale obtained for VR is inaccurate in comparison to those of the other 3D models. Third, the water is segmented to improve its texturing based on its mean color.

5.5.3. Common VR headsets

The VR applications are built for common consumer-grade VR headsets. For each 3D model, there are five VR applications summarized in Tab. 7. An application can be PC VR or standalone. "PC VR" means that a PC renders the scene and the VR headset only displays the output result via a link cable. "Standalone" means that the VR headset renders the scene without a PC and cable. The Oculus Quest headset (2) supports both PC VR and standalone modes. The rendering is usually better (higher framerate and texture quality) for PC VR than for standalone mode. The VR applications are experimented on (but not restricted to) an Oculus Quest, an Oculus Go, an Oculus Quest 2, a Lenovo Explorer and a HP Reverb G2, using Windows 10 for PC VR. They are built in Unity 2019.4, except in the case of the SteamVR Workshop which has its own game engine. The VR applications (which are not required for understanding this paper) are freely downloadable at

`https://maximelhuillier.fr`
`https://sidequestvr.com/user/330664`
`https://steamcommunity.com/profiles/76561198051374313/myworkshopfiles/`

5.5.4. Compromise between rendering quality and framerate

The PC VR applications render the models in Tabs. 2 and 6 with the relevant game engine's default setting. In the standalone mode, which has less computational resources, more things are needed to obtain a decent framerate. A decent framerate reduces the risk of VR sickness, and is also required for the Oculus Quest to store a VR application in a database (at least 60 frames per second for the App Lab, as summarized in the third line of Tab. 7).

Here we explain how to increase the framerate for a given 3D model. First we enable monoscopic rendering: the two eyes see the same image. This halves the computations needed in comparison to the default stereoscopic rendering and doubles the framerate. If the framerate is still too low, we discard triangles in the far background during the rendering. This is done carefully so as not to lose too much rendering quality: most discarded triangles should be occluded by undiscarded triangles. A simple method is to reduce the depth of the far clipping plane. If the framerate is still too low, another method is used. We split the 3D model (here, of Basalt and City) into local models of horizontal size $50m \times 50m$, then import all the local models in Unity (they share the same texture blocks as the global model), and finally we render only the nine local models in the 8-neighborhood of the VR user. A more sophisticated method is to combine local models and level-of-details like [Finsterwalder20].

5.5.5. Locomotion

We would like the VR user to be able to freely move on the ground of the environment like a pedestrian, with strong spatial awareness and a low risk of VR-sickness (Sec. 2.4).

Since the ground of the environment is greater than that of the tracking area, the VR user controls their motion in the environment using hand-held controllers. (The user can remain seated.) If the VR user chooses to move like a pedestrian (i.e. 2D motion), their location is constrained to be in a "motion surface" that approximates the ground. We define the motion surface using assumption A1': it extrapolates the trajectory of the acquisition camera up to a known vertical offset (see Appendix B for

VR headsets-compatibility	file	mode	installation	rendering
Oculus Go/Quest/Quest 2	APK	standalone	enable developer mode (sideloading)	monoscopic, all triangles
Oculus Quest/Quest 2	APK	standalone	easy (App Lab)	monoscopic, foreground, FPS>60
Windows Mixed Reality	ZIP	PC VR	easy	stereoscopic, all triangles
OpenVR	ZIP	PC VR	install Steam and SteamVR	stereoscopic, all triangles
SteamVR Workshop	VPK	PC VR	install Steam and SteamVR	stereoscopic, all triangles

Table 7: Five VR applications for each 3D model. Abbreviations: APK=Android package, ZIP=compressed directory including a Windows 10 executable and data, VPK=Valve package, FPS=frames per seconds.

details). Inspired by VR games, we implement two locomotion modes. The first has continuous translations on the motion surface with the left joystick and discrete yaw rotations with the right joystick. The second replaces the continuous translations by a "virtual laser pointer" that enables the user to instantaneously move to a location on the surface by using a point-and-click controller. The first locomotion mode provides the best spatial awareness and can be better for experienced users. The second locomotion mode has the lowest risk of VR-sickness and is better for inexperienced users.

6. Conclusion

The paper provides a new method for estimating the vertical direction of a triangulated surface reconstructed by photogrammetry from terrestrial imagery. The method is simple: first compute the unoriented vertical by principal component analysis of the acquisition trajectory followed by a Hough transform with votes of the triangle normals, then obtain the oriented vertical direction and the scale by projecting the trajectory onto the surface with respect to the two possible opposite vertical directions. We report experiments using the method in both man-made and natural environments reconstructed from consumer grade 360 videos with trajectory lengths of several hundred meters. Variants of the Hough transform and previous methods are compared on both synthetic and real datasets. We detail both our method assumptions and limitations. The method is used to build VR applications for immersive visualization of eight outdoor environments by common consumer-grade VR headsets. They are freely available on the internet, which is also a novelty in the context of photogrammetric scans of complex environments.

Declaration of competing interest

The author declares that he has no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Appendix A. Proof of Eq. 7

We use properties of the comatrix $Com(\mathtt{A})$ of a 3×3 matrix $\mathtt{A} = \begin{pmatrix}\mathbf{a} & \mathbf{b} & \mathbf{c}\end{pmatrix}$, i.e.

$$Com(\mathtt{A}) = \begin{pmatrix}\mathbf{b} \wedge \mathbf{c} & \mathbf{c} \wedge \mathbf{a} & \mathbf{a} \wedge \mathbf{b}\end{pmatrix} \tag{A.1}$$

and

$$(\mathtt{A}\mathbf{x}) \wedge (\mathtt{A}\mathbf{y}) = Com(\mathtt{A})(\mathbf{x} \wedge \mathbf{y}) \tag{A.2}$$

Thanks to Eq. 6 and the Chain rule,

$$\frac{\partial \mathbf{u}}{\partial x} = \frac{\partial n}{\partial (u,v,w)} \mathtt{R} \begin{pmatrix} 1/f \\ 0 \\ 0 \end{pmatrix}, \frac{\partial \mathbf{u}}{\partial y} = \frac{\partial n}{\partial (u,v,w)} \mathtt{R} \begin{pmatrix} 0 \\ 1/f \\ 0 \end{pmatrix}. \tag{A.3}$$

Thus

$$\begin{aligned} \frac{\partial \mathbf{u}}{\partial x} \wedge \frac{\partial \mathbf{u}}{\partial y} &= Com\left(\frac{\partial n}{\partial (u,v,w)}\right)\left(\mathtt{R} \begin{pmatrix} 1/f \\ 0 \\ 0 \end{pmatrix} \wedge \mathtt{R} \begin{pmatrix} 0 \\ 1/f \\ 0 \end{pmatrix}\right) \\ &= \frac{1}{f^2} Com\left(\frac{\partial n}{\partial (u,v,w)}\right) \mathtt{R} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}. \end{aligned} \tag{A.4}$$

Let $N = u^2 + v^2 + w^2$. Eq. 6 also implies

$$\frac{\partial n}{\partial(u,v,w)} = N^{-3/2}\begin{pmatrix} N-uu & 0-vu & 0-wu \\ 0-uv & N-vv & 0-wv \\ 0-uw & 0-vw & N-ww \end{pmatrix}. \quad \text{(A.5)}$$

Let $\begin{pmatrix} \mathbf{a} & \mathbf{b} & \mathbf{c} \end{pmatrix} = N^{3/2}\frac{\partial n}{\partial(u,v,w)}$. We have

$$\begin{aligned} \mathbf{a} \wedge \mathbf{b} &= (N\begin{pmatrix}1\\0\\0\end{pmatrix} - u\begin{pmatrix}u\\v\\w\end{pmatrix}) \wedge (N\begin{pmatrix}0\\1\\0\end{pmatrix} - v\begin{pmatrix}u\\v\\w\end{pmatrix}) \\ &= \begin{pmatrix}0\\0\\N^2\end{pmatrix} + Nv\begin{pmatrix}0\\w\\-v\end{pmatrix} + Nu\begin{pmatrix}w\\0\\-u\end{pmatrix} = Nw\begin{pmatrix}u\\v\\w\end{pmatrix} \quad \text{(A.6)} \end{aligned}$$

Using similar relations for $\mathbf{b} \wedge \mathbf{c}$ and $\mathbf{c} \wedge \mathbf{a}$, we obtain

$$Com\begin{pmatrix} \mathbf{a} & \mathbf{b} & \mathbf{c} \end{pmatrix} = N\begin{pmatrix} u & v & w \end{pmatrix}^T\begin{pmatrix} u & v & w \end{pmatrix}. \quad \text{(A.7)}$$

Thanks to Eq. A.4 and Eq. A.7,

$$\begin{aligned} \|\frac{\partial \mathbf{u}}{\partial x} \wedge \frac{\partial \mathbf{u}}{\partial y}\| &= \|\frac{1}{f^2}\frac{Com\begin{pmatrix} \mathbf{a} & \mathbf{b} & \mathbf{c} \end{pmatrix}}{N^3}\mathrm{R}\begin{pmatrix}0\\0\\1\end{pmatrix}\| \\ &= \frac{N\sqrt{N}}{f^2N^3}|\begin{pmatrix} u & v & w \end{pmatrix}\mathrm{R}\begin{pmatrix}0 & 0 & 1\end{pmatrix}^T| \quad \text{(A.8)} \end{aligned}$$

Since $\begin{pmatrix} u & v & w \end{pmatrix} = \begin{pmatrix} x & y & 1 \end{pmatrix}\mathrm{K}^{-T}\mathrm{R}^T$ (thanks to Eq. 6),

$$\begin{aligned} \|\frac{\partial \mathbf{u}}{\partial x} \wedge \frac{\partial \mathbf{u}}{\partial y}\| &= \frac{N^{-3/2}}{f^2}|1| = \frac{1}{f^2}\|\mathrm{K}^{-1}\begin{pmatrix}x\\y\\1\end{pmatrix}\|^{-3} \\ &= \frac{1}{f^2}\|\begin{pmatrix}(x-s/2)/f\\(y-s/2)/f\\1\end{pmatrix}\|^{-3} \quad \text{(A.9)} \end{aligned}$$

and we obtain Eq. 7.

Appendix B. Motion Surface

The motion surface extrapolates the trajectory of the acquisition camera computed by Structure-from-Motion 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, 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 VR camera on the motion surface thanks to f. Let

$$f(x,z) = a + \frac{\sum_j w_{cj}y_{cj}}{\sum_j w_{cj}} \text{ where } w_i = e^{-\frac{\sqrt{(x-x_i)^2+(z-z_i)^2}}{b}}, \quad \text{(B.1)}$$

a is a vertical offset of the VR 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 (as staircase effects can generate VR sickness).

We finish by a note about the coordinate systems. Ours is right-handed (as usual) with a vertical z-axis toward sky, whereas Unity's coordinate system 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# script needed by Unity that computes f: there is a $-\pi/2$ rotation around the x-axis and a sign change of the x-coordinate in order to switch from right-handed to left-handed. We therefore convert (x_{ci}, y_{ci}, z_{ci}) to $(-x_{ci}, z_{ci}, -y_{ci})$. Second, we use Unity to rotate by $-\pi/2$ around the x-axis when importing the 3D models (the sign change of x is implicitly added by Unity here). The acquisition locations and 3D models are then consistent in Unity.

References

[Alnuaimi17] Al-Nuaimi, A., 2017. Methods of point cloud alignment with applications to 3D indoor mapping and localization. Ph.D. thesis. Technische University Munich. Munich, Deutchland.

[Avidar18] Avidar, A., Malah, D., Barzohar, M., 2018. Point cloud registration refinement in an urban environment using 2d edge-maps, in: ICSEE international conference on the science of electrical engineering.

[Osnabruck] Borrmann, D., Elseberg, J., Nuchter, A., Lauterbach, H., . Robotics 3d scan repository.

Http://kos.informatik.uni-osnabrueck.de/3Dscans , last accessed on 2023/06/02.

[Botsch10] Botsch, M., Kobbelt, L., Pauly, M., Alliez, P., Levy, B., 2010. Polygonal mesh processing. AK Peters.

[Crandall11] Crandall, D., Owens, A., Snavely, N., Huttenlocher, D., 2011. Discrete-continuous optimization for large-scale structure from motion, in: Conference on Computer Vision and Pattern Recognition, IEEE.

[Dai17] Dai, A., Chang, A., Savva, M., Halber, M., Funkhouser, T., Niessmer, M., 2017. ScanNet: richly annotated 3d reconstructions of indoor scenes, in: Conference on Computer Vision and Pattern Recognition, IEEE.

[Dhanda19] Dhanda, A., Reina Ortiz, M., Weigert, A., Paladini, A., Min, A., Guyi, M., Su, S., Fai, S., Santana Quintero, M., 2019. Recreating cultural heritage environments for VR using photogrammetry, in: the 8th intl. workshop 3D-ARCH virtual reconstruction and visualization of complex architectures.

[Finsterwalder20] Finsterwalder, D., 2020. How we made the Audi AI:me's flying VR experience - part 2, optimizing photogrammetry data for VR using the realities.io pipeline editor. Https://medium.com/realities-io/how-we-made-the-audi-ai-mes-flying-vr-experience-part-2-41df10b1e63 , last accessed on 2023/06/02.

[Grayson18] Grayson, B., Penna, N., Mills, J., Grant, D., 2018. GPS precise point positioning for UAV photogrammetry. The Photogrammetric Record 33, 427–447.

[Hartley00] Hartley, R., Zisserman, A., 2000. Multiple View Geometry in Computer Vision (chapter: 3D reconstruction of cameras and structure). Cambridge University Press.

[Joo18] Joo, K., Oh, T., Kweon, I., Bazin, J., 2018. Globally optimal inlier set maximization for Atlanta frame estimation, in: Conference on Computer Vision and Pattern Recognition, IEEE. pp. 5726–5734.

[Kirschenbaum19] Kirschenbaum, M., 2019. Gopro max teardown. Https://gethypoxic.com/blogs/technical/gopro-max-teardown , last accessed on 2023/06/02.

[Klingner13] Klingner, B., Martin, D., Roseborough, J., 2013. Street view motion-from-structure-from-motion, in: International Conference on Computer Vision, Computer Vision Foundation.

[LaValle20] LaValle, S., 2020. Virtual Reality. Cambridge University Press.

[Lhuillier11] Lhuillier, M., 2011. Fusion of GPS and structure-from-motion using constrained bundle adjustments, in: Conference on Computer Vision and Pattern Recognition, IEEE.

[Lhuillier18] Lhuillier, M., 2018. Surface reconstruction from a sparse point cloud by enforcing visibility consistency and topology constraints. Computer Vision and Image Understanding 175, 52–71.

[Lhuillier19] Lhuillier, M., 2019. Local-convexity reinforcement for scene reconstruction from sparse point clouds, in: International Conference on 3D immersion.

[Lhuillier21] Lhuillier, M., 2021. From photogrammetric reconstruction to immersive VR environment, in: International Conference on 3D immersion.

[Liu22] Liu, Y., Chen, G., Knoll, A., 2022. Globally optimal vertical direction in Atlanta world. IEEE Transaction on Pattern Analysis and Intelligence 44, 1949–1962.

[Lobo03] Lobo, J., Dias, J., 2003. Vision and inertial sensor cooperation using gravity as a vertical reference. IEEE Transaction on Pattern Analysis and Intelligence 25, 1597–1608.

[Lutton94] Lutton, E., Maitre, H., Lopez-Krahe, J., 1994. Contribution to the determination of vanishing points using hough transform. IEEE Transactions on Pattern Analysis and Intelligence 16, 430–438.

[Mel19] Mel, K., Luca, B., Fabio, V., Varvara, A., Ugo, B., Elena, R., Fabio, M., Luca, F., Paraskevi, N.,

Martin, K., Eva, S., Whitworth, M., 2019. Workflows for virtual reality visualisation and navigation scenarios in earth sciences, in: the 5th international conference on geographical information systems theory, applications and management (GISTAM).

[Michot10] Michot, J., Bartoli, A., Gaspard, F., 2010. Bi-objective bundle adjustment with application to multi-sensor SLAM, in: International Symposium on 3D Data Processing, Visualization, and Transmission.

[Mukhopadhyay14] Mukhopadhyay, P., Chaudhuri, B., 2020. A survey of hough transform. Pattern Recognition 48, 993–1010.

[Obradovic20] Obradovic, M., Vasiljevic, I., Duric, I., Kicanovic, J., Stojakovic, V., Obradovic, R., 2020. Virtual reality models based on photogrammetric surveys - a case study of the iconostasis of the serbian orthodox cathedral church of saint nicholas in sremski karlovci (serbia). Applied Sciences 10, 1–21.

[Poux20] Poux, F., Valembois, Q., Mattes, C., Kobbelt, L., R., B., 2020. Initial user-centered design of a virtual reality heritage system: applications for digital tourism. Remote Sensing 12, 1–32.

[Rabbani05] Rabbani, T., Van Den Heuvel, F., 2005. Efficient hough transform for automatic detection of cylinders in point clouds, in: Laser Scanning, ISPRS. pp. 60–65.

[Tian20] Tian, N., Clement, R., Lopes, P., Boulic, R., 2020. On the effect of the vertical axis alignment on cybersickness and game experience in a supine posture, in: IEEE Conference on Games, IEEE. pp. 359–366.

[Triggs99] Triggs, B., McLauchlan, P., Hartley, R., Fitzgibbon, A., 1999. Bundle adjustment - a modern synthesis, in: Vision Algorithms: Theory and Practice, Springer-Verlag. pp. 298–372.

[Valve21] Valve, 2021. Steamvr/environments/photogrammetry. Https://developer.valvesoftware.com/wiki/SteamVR/Environments/Photogrammetry , last accessed on 2023/06/02.

[Vu12] Vu, H., Labatut, P., Pons, J., Keriven, R., 2012. High accuracy and visibility-consistent dense multiview stereo. IEEE Transactions on Pattern Analysis and Machine Intelligence 34, 889–901.

[Wikipedia22] Wikipedia, 2022. Metaverse. Https://en.wikipedia.org/wiki/Metaverse , last accessed on 2023/06/02.

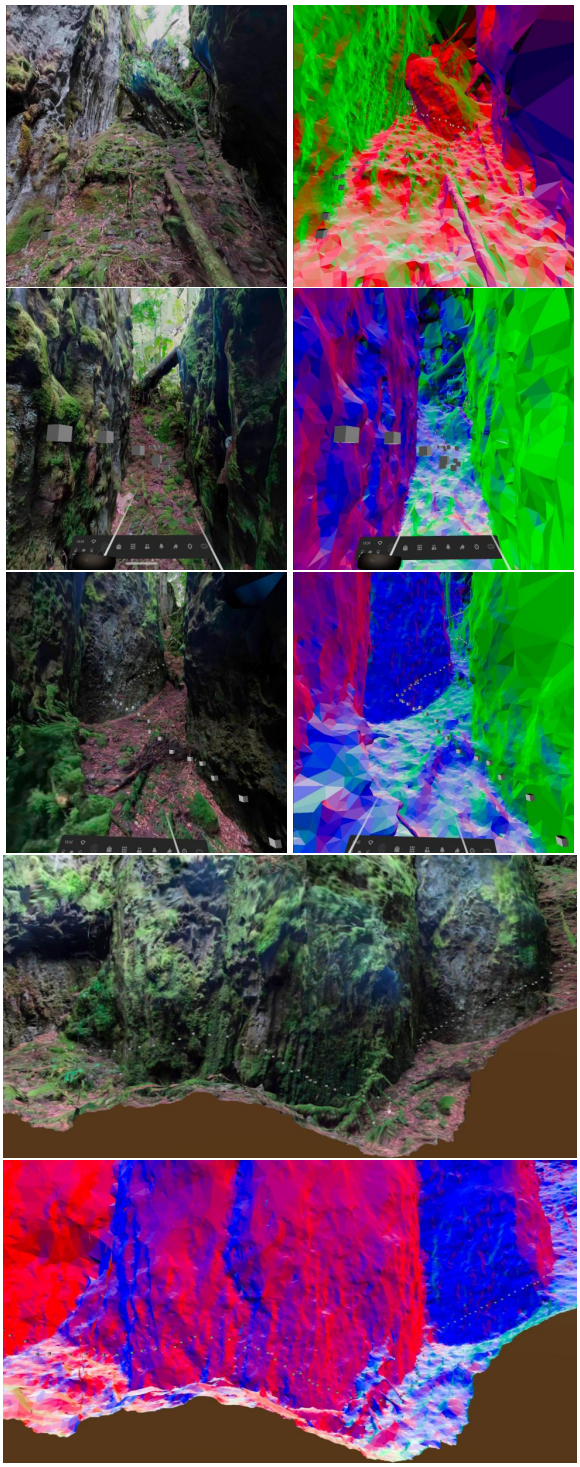


Figure 12: Screenshots of Basalt taken by a VR headset (Oculus Quest).

Figure 13: Screenshots of City taken by a VR headset (Oculus Quest).

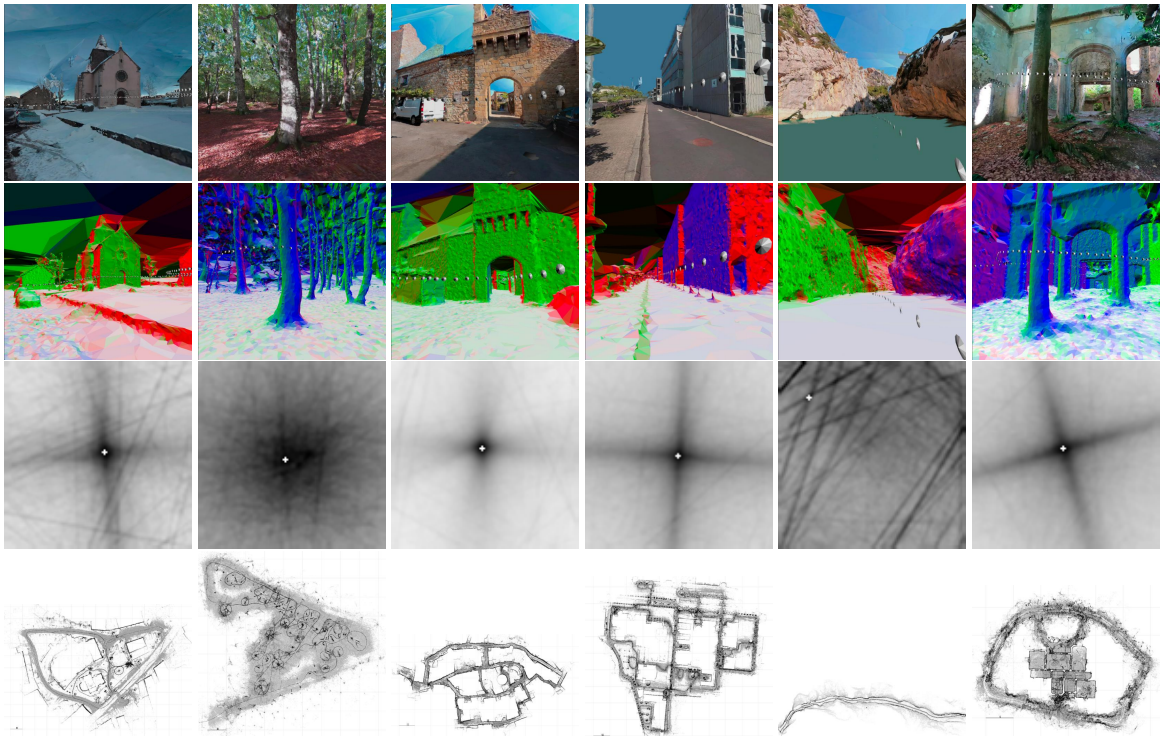


Figure 14: Screenshots by an Oculus Quest, the Hough space (H1') and a top view of the reconstructed points for the 3D models in Tab. 6.