

# Reconstruction of environments

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

This is the initial version of the fourth chapter of a book, referenced as: Maxime Lhuillier, Reconstruction of Environments, chapter in book “Omnidirectional Vision: from Theory to Applications”, ISTE Wiley, editors: Pascal Vasseur and Fabio Morbidi, pages 63–104, 2023.

The published version of the chapter is at <https://doi.org/10.1002/9781394256440.ch3>

The published version of the book is at <https://www.iste.co.uk/book.php?id=2075>

## Summary

The reconstruction of environments from omnidirectional images has two steps. The first one estimates the geometry and is described in other chapters. It includes structure-from-motion (SfM), calibration or self-calibration. It estimates all parameters of the camera(s) including 6 degrees-of-freedom (DoF) poses, intrinsic parameters and radial distortions. SfM also provides a sparse cloud of 3D points reconstructed from features (e.g. points and lines) detected and matched in the images. The second step provides an approximation of the environment that is more complete than the sparse cloud: a dense cloud of points or a triangulated surface in 3D. This chapter surveys the second step in the previous works. In most cases, they assume that the camera is moving in a rigid scene, or similarly, that several cameras take images of a non-rigid scene at a same time.

We start by prerequisites on reconstruction using perspective cameras (Sec. 1) and a discussion on omnidirectional cameras for reconstruction (Sec. 2). Then the previous works are classified in several groups: dense stereo adapted to omnidirectional cameras (Sec. 3), reconstruction from only one central image (Sec. 4), reconstruction of a non-rigid scene by using a stationary non-central camera (Sec. 5), and reconstruction by a moving camera (Sec. 6). Last we conclude in Sec. 7.

## 1 Prerequisites

Sec. 1 mostly summarizes standard methods for reconstruction of a rigid scene from input images taken by a perspective camera. This is helpful to understand the case of an environment reconstructed by using an omnidirectional camera. We start with reminders on image rectification and matching constraints (Sec. 1.1), links between disparity and depth (Sec. 1.2). Then dense stereo methods are summarized: semi-global matching (Sec. 1.3), plane sweeping (Sec. 1.4), minimization of global energy (Sec. 1.5), propagation methods (Sec. 1.6). There are also surface reconstruction methods from point clouds (Sec. 1.7). All these methods are used or adapted to omnidirectional images in the works described in the

remainder of the chapter. Last, Sec. 1.8 shortly mentions that the 3D can be provided by other sensors added to an omnidirectional camera.

## 1.1 Image rectification and matching constraints

If two points in two distinct images are projections of a same 3D point in the scene, they meet the epipolar constraint [19]: there is a pair of corresponding epipolar lines that include the two points. Since this constraint is known by the geometry estimation step, the matching problem of a pixel in one image is reduced to an 1D search of another pixel in the other image. This search is simplified and accelerated by *rectification*: there are 2D homographies that map the original images to auxiliary images, named rectified images, such that every pair of corresponding epipolar lines becomes a same horizontal line. Such a rectification cannot be applied in a neighborhood of the epipoles, if they are in the original images. (Otherwise the rectified images would be infinite.)

After rectification, we can search the match of a point  $(x \ y)$  in the first image as a point  $(x + d \ y)$  in the second image by finding the disparity  $d$  that minimizes a cost: a photo-consistency measure like the absolute difference of gray levels of the two points. ( $x$ ,  $y$  and  $d$  are integers.) The robustness of this cost increases by *spatial aggregation*, e. g. if we replace it by the sum of absolute differences in local windows centered at  $(x \ y)$  and  $(x + d \ y)$ . Other costs are possible like sum of squared differences, one minus zero-mean centered correlation, using luminance or RGB channels, etc.

The strategy that consists to compute the disparity  $d$  of each pixel as a simple search of the minimizer of an aggregated photo-consistency measure is called *winner-take-all (WTA)*. However there are two cases where the minimizer is not well defined and WTA is unreliable. In the former, the pixel has low texture in its neighborhood. In the latter, which is called the aperture problem, the image gradient is orthogonal to the epipolar line in the search area of the disparity computation. These cases occur even if the search space is reduced by enforcing lower and upper bounds on  $d$ . Thus additional constraints are needed to reduce the risk of bad matches, e. g.

- disparity gradient limit: disparities  $d(x)$  and  $d(x + 1)$  of adjacent pixels at  $x$  and  $x + 1$  meets  $|d(x) - d(x + 1)| \leq c$  where  $c$  is a threshold.
- ordering: the matching conserves the ordering, i. e.  $x + d(x) < x' + d(x')$  if  $x < x'$ .
- uniqueness: the function  $x \mapsto x + d(x)$  is injective.

This list of constraints is not exhaustive.

## 1.2 From disparity to depth

Since the goal is reconstruction, Sec. 1.2 reminds relations between disparity  $d$  and depth  $Z$ . Assume for a moment that we have a pair of rectified images (Sec. 1.1) with disparities computed by dense stereo, e. g. a method in the next paragraphs. The projections of a 3D point  $(X \ Y \ Z)$  in the two rectified images are

$$(x \ y) = (fX/Z \ fY/Z) \text{ and } (x + d \ y) = (f(X + T)/Z \ fY/Z) \quad (1)$$

where  $f$  is the focal length expressed in pixels and  $T$  is the baseline. Thus the depth is the inverse of the disparity up to a scale factor:

$$Z = \frac{fT}{d}. \quad (2)$$

Furthermore, we examine how a small error in the disparity propagates to an error of the depth. Assume that  $d$  follows a random variable with mean  $d_0$  and standard deviation  $\sigma_d$ .

By using a first-order approximation of  $Z$  as a function of  $d$ ,  $Z$  follows a random variable with mean  $Z_0 = fT/d_0$  and standard deviation  $\sigma_Z$  that meets  $\sigma_Z^2 = (\frac{\partial Z}{\partial d}(d_0))^2 \sigma_d^2$ . We obtain

$$\sigma_Z = \frac{Z_0^2}{f|T|} \sigma_d. \quad (3)$$

On the one hand, we see that experimental choices provide depths with small errors: large image resolution (i. e. large  $f$ ), large baseline  $|T|$ , and scenes without far background (i. e. small  $Z_0$ ). On the other hand, camera price increases with large resolution, disparity computation is more difficult with large baseline, and reconstruction can be restricted to foreground. Thus a compromise must be done.

### 1.3 Dynamic programming and semi-global matching methods

A lot of methods are possible to enforce matching constraints. Dynamic programming minimizes a of sum of pixel costs for each epipolar line by enforcing several constraints: uniqueness, ordering and bounds. It also uses the epipolar constraint by using two rectified images. First it computes a cost  $C(x_1, x_2)$  in the disparity space image, i. e. for all  $x_1$  in the first image and  $x_2$  in the second one with bounds on  $x_2 - x_1$  where  $x_i \in \mathbb{N}$ . Then it searches the connected path in the disparity space image from  $(0, 0)$  to  $(x_{max}, x_{max})$  that minimizes the sum of  $C$  along the path corrected by terms to deal with occlusions. The path only has moves  $(+1, +1)$ ,  $(+1, 0)$  and  $(0, +1)$  to enforce the constraints and to make possible a recursive computation of the sum. The move  $(+1, +1)$  corresponds to a match between pixel pairs and the two other moves correspond to occlusions.

However dynamic programming does not enforce disparity constraint between adjacent lines and makes errors for this reason. The popular *semi-global matching (SGM)* [21] removes this drawback and also considers minimal paths but in a different way. It first computes a *cost volume*  $C(\mathbf{p}, d)$  where  $\mathbf{p}$  is the pixel coordinate in a reference image and  $d$  is a disparity ( $d$  is bounded). Then for each discrete image direction  $\mathbf{r}$ , e. g.  $\mathbf{r} \in \{-1, 0, +1\}^2 \setminus \{(0, 0)\}$ , it searches the path in the volume with direction  $\mathbf{r}$  from image boundaries to every  $(\mathbf{p}, d)$  that minimizes the sum of  $C$  along the path corrected by terms to deal with occlusion and slanted surfaces. All paths with a given  $\mathbf{r}$  are computed recursively: the cost  $L_{\mathbf{r}}(\mathbf{p}, d)$  of the minimal path that ends at  $(\mathbf{p}, d)$  is a function of  $L_{\mathbf{r}}(\mathbf{p} - \mathbf{r}, \cdot)$ . Last it chooses the  $d$  for each  $\mathbf{p}$  that minimizes  $\sum_{\mathbf{r}} L_{\mathbf{r}}(\mathbf{p}, d)$ . This sum of costs is a spatial aggregation and improves robustness. It is better than that of the dynamic programming, which is restricted to the horizontal direction. The time complexity is the product of the 3 dimensions of the volume.

The SGM can be generalized to more than two images even if they are not rectified. The range of disparity is replaced by a finite set  $H$  of hypothesis depths. A cost in the volume is defined for every pair  $(\mathbf{p}, Z)$  of pixel  $\mathbf{p}$  in a reference image and  $Z \in H$  by a multi-view photo-consistency of the reprojections in all images of the 3D point corresponding to  $(\mathbf{p}, Z)$ . However, the time computation of the cost volume can be long since the time complexity is the product of the volume dimensions and the (squared) number of images.

### 1.4 Plane sweeping methods

The plane sweeping methods are other popular (WTA) stereo methods. They estimate a depth map in a reference image by sweeping the scene using parallel planes. They use the following property for fast computations of the photo-consistency: each plane  $\pi$  induces 2D homographies  $H_i^\pi$  between the input images such that, for every 3D point in  $\pi$ , its projection in the reference image is mapped by  $H_i^\pi$  to its projection in the  $i$ -th image.

During the computation, each pixel  $\mathbf{p}$  of the reference image has a depth  $Z(\mathbf{p})$  and a photo-consistency  $C(\mathbf{p})$  initialized by  $C(\mathbf{p}) = +\infty$ . For each plane  $\pi$ , photo-consistencies of 3D points in  $\pi$  are computed by mapping all images to the reference image using  $\forall i, H_i^\pi$  and

followed by spatial aggregations. Let  $Z_\pi(\mathbf{p})$  be the depth of the 3D point in  $\pi$  that projects to  $\mathbf{p}$ . Let  $C_\pi(\mathbf{p})$  be the photo-consistency of this 3D point. If  $C_\pi(\mathbf{p})$  is better (smaller) than  $C(\mathbf{p})$ , then the method resets  $C(\mathbf{p}) = C_\pi(\mathbf{p})$  and  $Z(\mathbf{p}) = Z_\pi(\mathbf{p})$ . Once this is done for each pixel  $\mathbf{p}$ , the next plane  $\pi$  is considered to improve the depth map  $Z$ .

These computations are ideal for segments of the scene surface that are parallel to the planes. Basic plane sweeping only uses one plane direction: the planes are fronto-parallel. Advanced plane sweeping uses several directions that are the expected ones in the scene [15], e. g. horizontal planes for ground surface and vertical planes for walls.

## 1.5 Minimization of global energy (or cost function)

A standard approach for computing a depth map  $Z(\mathbf{p})$  for the pixels  $\mathbf{p}$  in a reference image is global energy minimization (Sec. 3.1.3 in [14]). It enforces depth smoothness in a more general way than the SGM and the plane sweep methods. It also takes as input a finite set  $H_{\mathbf{p}}$  of hypothesis depths for each pixel  $\mathbf{p}$ , but it does not simply assume that the true depth has the best photo-consistency. Here the depth-per-pixel depends on those in immediate neighborhood. The solution  $Z$  minimizes a cost function

$$E(Z) = \sum_{\mathbf{p}} C(\mathbf{p}, Z(\mathbf{p})) + \lambda \sum_{(\mathbf{p}, \mathbf{q}) \in \mathcal{N}} R(Z(\mathbf{p}), Z(\mathbf{q})) \quad (4)$$

where  $Z(\mathbf{p}) \in H_{\mathbf{p}}$ ,  $C(\mathbf{p}, Z)$  is a photo-consistency measure of the 3D point with depth  $Z$  projected in  $\mathbf{p}$ ,  $R$  is a spatial regularization term,  $\sum_{\mathbf{p}}$  is a sum for all pixels in the reference image,  $\sum_{(\mathbf{p}, \mathbf{q}) \in \mathcal{N}}$  is a sum for all pairs of adjacent pixels in the reference image using a neighborhood system (4-neighborhood or 8-neighborhood),  $\lambda > 0$  is a weighting parameter. The spatial regularization tends to enforce similar depths for adjacent pixels in the solution  $Z$ . (The greater  $\lambda$ , the stronger the regularization.) In practice, better results are obtained by bounding  $C$  and  $R$ , e. g. use

$$R(Z_1, Z_2) = \min(\tau, |Z_1 - Z_2|) \quad (5)$$

with a threshold  $\tau > 0$ . The terms  $\sum_{\mathbf{p}} C$  and  $\sum_{(\mathbf{p}, \mathbf{q}) \in \mathcal{N}} R$  are also called data term and smoothing term, respectively.

The minimization is difficult (NP-hard) in a general context, but, fortunately, there are efficient methods to do it for dense stereo (if  $R$  meets the triangular inequality and  $R(Z, Z) = 0$ ) that are based on graph-cut. We can also use a second-order prior smoothness instead of a first-order prior one, e. g. use  $|Z_1 + Z_3 - 2Z_2|$  instead of  $|Z_1 - Z_2|$  for three adjacent pixels instead of two. This reduces the risk of fronto-parallel surface (i. e. with constant depth) and provides a more piece-wise planar surface, but the minimization method is more complicated.

## 1.6 Propagation methods

The summarized methods above have drawbacks: they assume a finite set of hypothesis depths or disparities for each pixel and require a post-processing to obtain a global 3D model by merging the depth maps of several images.

Patch propagation (Sec. 3.2 in [14]) is a standard approach to avoid these drawbacks, that directly reconstructs 3D points by using all images. A patch  $p$  is a small piece of plane in 3D that locally approximates the scene surface. It is reconstructed by optimizing its photo-consistency, which is a function of the patch center  $\mathbf{c}(p)$  and patch normal  $\mathbf{n}(p)$ , in selected images  $V(p)$ . First, a set  $S$  of seed patches is initialized by matching features in the images and by reconstructing them. (This can be provided by the geometry estimation step.) New patch will be initialized from seed patch in its neighborhood. This neighborhood is defined by adjacency of cells in images where project the patches. Each image cell also

stores a list of patches that are projected in the cell. Then  $S$  evolves as follow: pull a patch  $p$  from  $S$  and try to add new patches  $p'$  in its neighborhood. If there is room for  $p'$  in the neighborhood of  $p$  and if the reconstruction of  $p'$  is successful, then  $p'$  is added to  $S$  and to the image cells where it is projected. In a few words,  $p'$  is initialized using  $\mathbf{n}(p') = \mathbf{n}(p)$  and  $V(p') = V(p)$  and  $\mathbf{c}(p')$  by ray-plane intersection, then  $\mathbf{n}(p')$  and  $\mathbf{c}(p')$  are refined by optimizing the photo-consistency of  $p'$ . Once  $S$  is empty, a filtering removes inconsistent patches (in cells) by examining the visibility and geometry of the patches. The growing and filtering alternate many times, and the patches progressively cover the surface scene.

Match propagation [34] is an ancestor of the patch propagation for two images, which is still used for reconstruction of environments. It also needs neither hypothesis depths nor cost volume definition. The matches are pairs  $(\mathbf{p}, \mathbf{q})$  of pixels that have good photo-consistencies and that should meet disparity constraints: uniqueness and small 2D disparity gradient. (The epipolar constraint is optional.) The 2D disparity of  $(\mathbf{p}, \mathbf{q})$  is  $\mathbf{q} - \mathbf{p}$ . The 2D disparity gradient between  $(\mathbf{p}, \mathbf{q})$  and  $(\mathbf{p}', \mathbf{q}')$  is small if

$$(\mathbf{q}' - \mathbf{p}') - (\mathbf{q} - \mathbf{p}) \in \{-1, 0, +1\}^2. \quad (6)$$

First, a set  $S$  of seed matches is initialized from the sparse matching of interest points done by the geometry estimation step. The list  $L$  of accepted matches is initialized to empty. Then  $S$  evolves as follow: pull a match  $(\mathbf{p}, \mathbf{q})$  from  $S$  and try to add new matches  $(\mathbf{p}', \mathbf{q}')$  in its neighborhood, i. e. if the absolute coordinates of  $\mathbf{p}' - \mathbf{p}$  and  $\mathbf{q}' - \mathbf{q}$  are below a threshold. If the uniqueness constraint is met (i. e.  $\mathbf{p}'$  and  $\mathbf{q}'$  do not appear in  $L$ ) and if the 2D disparity gradient is small, then  $(\mathbf{p}', \mathbf{q}')$  is added to  $S$  and  $L$ . The match  $(\mathbf{p}, \mathbf{q})$  has the best photo-consistency in  $S$ , which is defined by zero-mean normalized cross-correlation in local windows centered on  $\mathbf{p}$  and  $\mathbf{q}$ . This choice provides a best-first match propagation with advantages including robustness with respect to false seed matches and occlusion handling. The time complexity is only  $\mathcal{O}(n \log n)$  where  $n$  is the number of pixels in the images.

## 1.7 Surface reconstruction methods

The previously summarized methods estimate disparity and depth maps in reference images, points and patches in 3D. They do not provide a triangulated surface, although this is the standard representation for a lot of applications. Thus we briefly explain how to estimate a triangulated surface using an input cloud of 3D points reconstructed from images.

We focus on methods that discretize the space by using a 3D Delaunay triangulation whose vertices are points. They label each Delaunay tetrahedron by “inside” or “outside” such that the target surface separates the inside and outside tetrahedra. An alternative of tetrahedra would be voxels in a regular grid, but this is not efficient for large scale environments and uneven distributions of input points. Here we need definitions. A *reconstruction ray* is a line segment  $\mathbf{CP}$  where  $\mathbf{P}$  is a point and  $\mathbf{C}$  is a location (or center) of a camera that reconstructs  $\mathbf{P}$ . The full line that includes  $\mathbf{C}$  and  $\mathbf{P}$  is written  $(\mathbf{CP})$ . Let  $O$  be the set of the outside tetrahedra. The surface  $\partial O$  is the *boundary* of  $O$ , i. e. the set of Delaunay triangles that are faces of exactly one tetrahedron in  $O$ . The methods summarized in Sec. 1.7 enforce visibility constraints: each triangle in  $\partial O$  should not be crossed by a reconstruction ray (unless  $\mathbf{P}$  is a vertex of the triangle). They also regularize  $\partial O$  and deal with noisy input point cloud.

A graph-cut [57] method estimates the tetrahedron labels that minimize a cost function  $E_{vis} + \lambda E_{qual}$  composed of a visibility term  $E_{vis}$  and a surface quality term  $E_{qual}$  with a weighting parameter  $\lambda > 0$ . The term  $E_{vis}$  not only avoids triangle in  $\partial O$  that is crossed by a  $\mathbf{CP}$ , it also enforces tetrahedron to be in  $O$  if it includes a camera location  $\mathbf{C}$ , and it tries to enforce tetrahedron to not be in  $O$  if it is crossed by a  $(\mathbf{CP}) \setminus \mathbf{CP}$  where  $\mathbf{P}$  is one of the four vertices of this tetrahedron (Fig. 1). The term  $E_{qual}$  tries to avoid triangle in  $\partial O$  that is face of “too flat” tetrahedron. The choice of  $\lambda$  is a trade-off: thin structures (e. g. posts) of  $\partial O$  are removed if  $\lambda$  increases,  $\partial O$  is less robust to bad points if  $\lambda$  decreases.

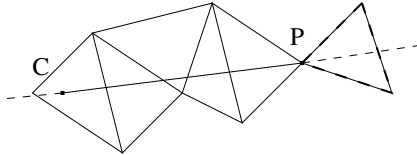


Figure 1: Visibility constraints for surface reconstruction in a 3D Delaunay triangulation using camera center  $\mathbf{C}$  and point  $\mathbf{P}$ . Here tetrahedra and triangles are drawn by triangles and edges, respectively. The tetrahedron that includes  $\mathbf{C}$  must be in  $O$ . The tetrahedron with hatched edges should not be. The four triangles crossed by  $\mathbf{CP}$  should not be in  $\partial O$ .

In contrast to the graph-cut methods, a manifold method [35] enforces the manifold constraint on  $\partial O$ . This means that every point of the surface (i. e. in  $\cup \partial O$ ) has a surface neighborhood that is homeomorphic to a disk. This constraint is useful not only for surface regularization during computation but also for post-processing and applications such as surface smoothing, texturing and rendering. Fig. 2 shows tetrahedra configurations that are avoided by the manifold constraint. Let  $r(\Delta)$  be the number of reconstruction ray(s) that

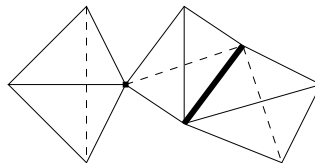


Figure 2: A set of three tetrahedra whose boundary is not manifold. There is a non-manifold vertex at the intersection of the tetrahedra of the left and middle, and a non-manifold edge at the intersection of the tetrahedra of the middle and right. (The non-manifold points have coarser pencil stroke.) All other boundary points are manifold.

cross a Delaunay tetrahedron  $\Delta$ . We say that  $\Delta$  is *free-space* if  $r(\Delta) > 0$ , otherwise  $\Delta$  is *matter*. The goal is to find the set  $O$  of free-space tetrahedra that maximizes a visibility score function

$$r(O) = \sum_{\Delta \in O} r(\Delta) \quad (7)$$

subject to the constraint that  $\partial O$  is manifold. A greedy method approximates the solution of this problem: add progressively free-space tetrahedra in  $O$  as long as  $\partial O$  remains manifold. First try to add  $\Delta$  with the highest  $r(\Delta)$  such that  $O$  grows from the most confident free-space to the less confident free-space. Then try to add several tetrahedra at once to allow topological changes of  $\partial O$  and to escape from local extrema. Efficient manifold tests are locally applied at each tried update of  $O$ .

## 1.8 Estimation of the 3D using other sensors

Since this chapter is limited to the reconstruction using cameras, it does not detail previous methods that use other sensors to obtain the 3D, even if the acquisition hardware includes an omnidirectional camera. Here are two examples of such hardware. A first one is designed for outdoor environments [4]: Google Street View uses both omnidirectional camera and laser scanners mounted on the roof of a car. The scanner is used to estimate the dominant planes of the environments. GPS and IMU are also used for the estimation of the camera parameters. Another one is designed for indoor environments [17]: a RGB-D acquisition system is formed by height Asus Xtion Pro Live sensors (Kinect-like sensors). Each of them

has a structured light depth sensor (It projects a pattern of light that is only seen by an infrared camera.) and a standard camera. They are rigidly fixed around a symmetry axis such that their standard cameras are forming a 360 camera.

## 2 Pros and cons for using omnidirectional cameras

Using an omnidirectional camera to reconstruct an environment seems like a good idea. Its field of view (FoV) is larger than that of a standard camera: only one image is enough to capture the (almost) complete part of the environment that is visible from a single view point. Thus the number of images needed for the environment reconstruction is lower than that of a standard camera. This not only simplifies and accelerates the image acquisition (no need to rotate a camera to see all around) but also can accelerate the computation (e.g. the number of camera poses refined by bundle adjustment is smaller) and simplify sub-problems (e.g. the detection of loop closures). Furthermore, this can also improve the accuracy of a reconstructed point since this point is visible and tracked in a higher number of images. However, these qualitative comparisons only hold if the resolutions (number of image pixels per steradian) of the compared omnidirectional and standard cameras are similar. They are, if the omnidirectional camera is a multi-camera, which is formed by a rigid set of several standard cameras that point to various directions. They are not, if the omnidirectional camera is a catadioptric camera, which is formed by a mirror in front of a standard camera. Here “standard” means monocular and can be fisheye for multi-camera.

### 2.1 Multi-cameras

The multi-camera case (Fig. 3.bcd) is the most favorable one for reconstruction accuracy but has drawbacks: the standard cameras must be synchronized, each of them has its own intrinsic parameters including distortion ones, there are additional parameters of relative poses between them. Furthermore, a lot of reconstruction methods take as input composite images, e.g. spherical images, obtained by stitching the images taken by the standard cameras. These methods implicitly assume that the multi-camera is central and its parameters are accurately known for stitching. However these assumptions can be bad, e.g. the larger the number of standard cameras, the more doubtful the central assumption. The consequences are not only blur/ghosting artifacts in the spherical images, but also inaccurate spherical calibration. (A perfect spherical image is equirectangular: the spherical coordinates of ray direction corresponding to pixel are linear to pixel coordinates.) Bad assumptions obviously degrade the reconstruction accuracy if the spherical images are used instead of the original ones.

### 2.2 Catadioptric cameras

The catadioptric case is less favorable for reconstruction accuracy due to its lower resolution. Furthermore, this resolution is not spatially-uniform and the vertical FoV, assuming that the optical axis is vertical, is limited. These drawbacks can be partially compensated by using still images (instead of video images which have lower resolutions) and by using an equiangular catadioptric camera. Equiangular means that the angle between back-projected ray and optical axis is linear to the distance between the corresponding pixel and the principal point. Equiangular catadioptric cameras also have favorable vertical FoV compared to other catadioptric ones: about 50 degrees above and below the horizontal plane (Fig. 3.a). Such considerations are important in practice, e.g. to reconstruct both ground surface and facades in an urban environment. However the equiangular catadioptric cameras are not central cameras, which can complicate the 3D computations. For example, image rectification for depth computation requires the central assumption. An advantage of the usual



(a) Catadioptric Camera in [61]

(b) Professional rig in [38]



(c) DIY rig #1 in [32]

(d) DIY rig #2 in [43]

Figure 3: A few omnidirectional systems (among many others) used for environment reconstruction. The catadioptric camera is formed by the 0-360 mirror and the Canon Legria HSF10. It is equiangular with a good vertical FoV for both ground and buildings. The professional rig is a PointGrey’s Ladybug multi-camera mounted on a vehicle. It is composed of six global shutter fisheye cameras. (The others in the figure are rolling shutter.) The do-it-yourself (DIY) rigs are formed by four Gopro Hero 3 cameras mounted on a helmet.

(central) catadioptric cameras is the small number of intrinsic parameters to be estimated in comparison to the multi-cameras.

### 2.3 Toward a wide use of the 360 cameras

Recently, 360 cameras become popular for several reasons: decreasing prices and dimensions, increasing resolutions and frequencies, trendy applications such as 360 videos and content generation for virtual reality (VR). Their growing is almost explosive during years 2015-2020 since more than thirty 360 cameras appear in this period [59], which (roughly) begins when Youtube started the support of 360 videos. A lot of them are multi-cameras composed of two fisheyes that see in opposite directions (Fig. 4), with a price less than 800 euros. For these reasons, and thanks to their resolution advantage over catadioptric cameras, the bi-cameras become dominant in the publications about reconstruction using omnidirectional cameras after 2015. Before 2015, most publications use catadioptric cameras or professional (costly) multi-cameras.



Figure 4: Bi-cameras among many others. From left to right: Ricoh Theta S, Samsung Gear 360, Garmin Virb 360, image pair taken by the Virb. The bi-cameras are dominant in the publications about reconstruction using omnidirectional camera during years 2015-2020.

### 3 Adapt dense stereo to omnidirectional cameras

This section surveys methods that adapt dense stereo methods in Sec. 1 to images taken by an omnidirectional camera assuming that the calibration (mapping between ray directions and pixels of the original images) is known. The adaptation is partly in the rectification step (Sec. 1.1), with three possibilities in the omnidirectional context: planar, cylindrical and spherical rectifications. Here the name is that of a coordinate system for ray directions. The pixel color of a rectified image is generically defined as follows: take the 2D coordinates of the pixel, multiply them by scale factors and add offsets to obtain the two coordinates of a ray direction, get the color where projects the ray direction in the original image. The standard rectifications transform two original images to two rectified images such that each pair of corresponding epipolar curves in the former is mapped to a same row (or equivalently, a same column) in the latter. There are also weak versions of rectifications that drop the constraint on the straight epipolar lines and only reduce the distortions between the original images (to make easier the pixel matching), especially in the planar case.

Previous works using spherical, cylindrical and planar rectifications are in Sec. 3.1, 3.2 and 3.3, respectively. Plane sweeping methods are also adapted into sphere sweeping methods and summarized in Sec. 3.4. Last there are other methods that use neither sweeping nor standard rectification in Sec. 3.5.

#### 3.1 Spherical rectifications

The method in [36] rectifies images taken by two arbitrary cameras with centers  $\mathbf{C}$  and  $\mathbf{C}'$  such that the epipolar curves become horizontal lines (Fig. 5.a). It uses spherical coordinates of a 3D point  $\mathbf{P}$  in two well chosen spherical coordinate systems: with origins  $\mathbf{C}$  and  $\mathbf{C}'$ , with same zenith direction  $\mathbf{z} = \mathbf{C}' - \mathbf{C}$  and same azimuth direction  $\mathbf{a}$  (orthogonal to  $\mathbf{z}$ ). Remind that the latitude  $p \in [0, \pi]$  is the angle between  $\mathbf{z}$  and  $\mathbf{P} - \mathbf{C}$ , the longitude  $t \in [0, 2\pi]$  is the angle between  $\mathbf{a}$  and  $\mathbf{P} - \mathbf{C}$  projected on a plane orthogonal to  $\mathbf{z}$ . Thus  $\mathbf{P}$  has a same longitude  $t$  and distinct latitudes  $p$  and  $p'$  in the two coordinate systems. A rectified image is  $L \times 2L$  such that pixel  $(x, y)$  has the color of the point in an original image whose back-projected  $\mathbf{P}$  has coordinates  $p = \pi x/L$  and  $t = \pi y/L$ . Corresponding epipolar curves in the original images (not shown here) are mapped in a same horizontal line. Large image distortions occur near boundaries  $x = 0$  and  $x = L$  where the epipoles are mapped.

The spherical rectification in [36] is applied on a catadioptric camera with parabolic mirror in [5] before a graph-cut method with first-order prior smoothness (Sec. 1.5). However the depths corresponding to the disparities near the epipoles are unreliable, at least because the reconstruction rays corresponding to matched pixels near the epipoles are almost parallel. To improve these depths, a third catadioptric image is taken at another camera location  $\mathbf{C}''$  that is not collinear to the two first ones  $\mathbf{C}$  and  $\mathbf{C}'$ , then a second disparity map is computed between  $\mathbf{C}'$  and  $\mathbf{C}''$  by using the same method, last the inaccurate depths of the pair  $(\mathbf{C}, \mathbf{C}')$  near epipoles are replaced by more accurate depths of the pair  $(\mathbf{C}', \mathbf{C}'')$  in the common reference image (of  $\mathbf{C}'$ ).

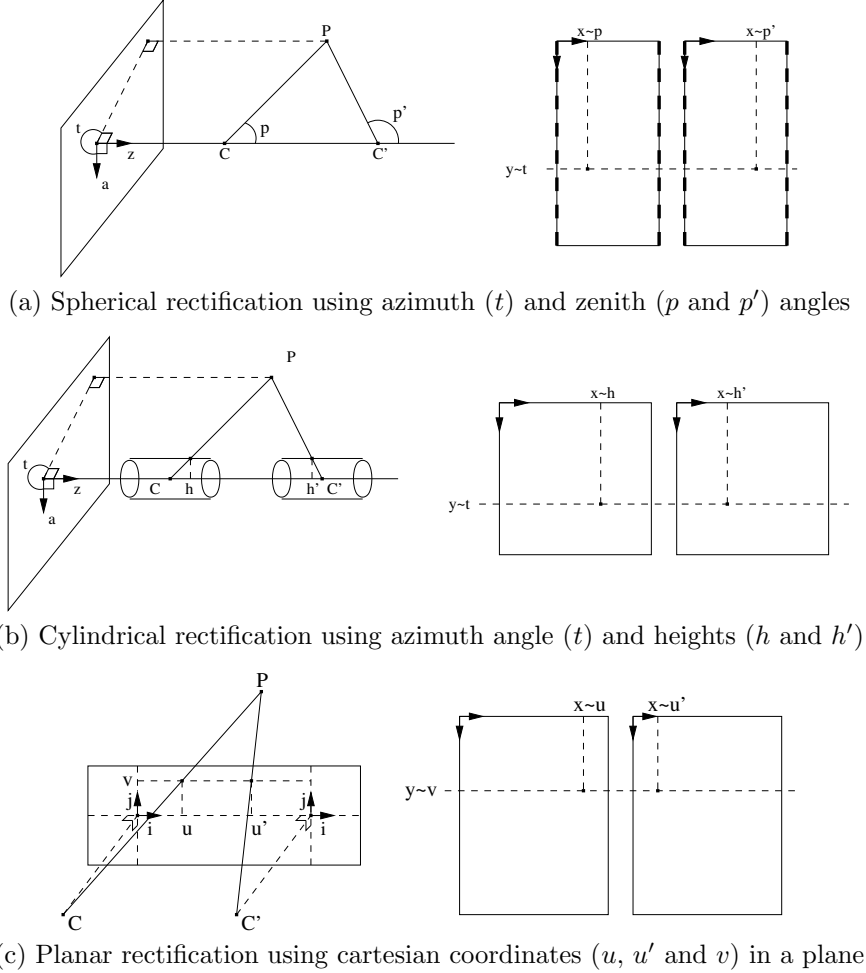


Figure 5: Rectifications such that pairs of corresponding epipolar curves become same rows. Left: parameters of rays  $\mathbf{CP}$  and  $\mathbf{C'P}$ . Right:  $x$  and  $y$  coordinates in the rectified images are linear to the ray parameters. Top: the epipoles are projected on the hatched borders. Middle: the two cylinders have same radius and  $h$  is relative to  $\mathbf{C}$  ( $h'$  is relative to  $\mathbf{C'}$ ). Bottom: the planar rectification is also given for comparison.

Two rectified spherical images [36] with a vertical baseline are taken by using a commercial line-scan camera in [27]. Then a global energy function (Sec. 1.5) is minimized to estimate a depth map thanks to a variational approach that avoids disparity discretization. Both the input images with large resolution  $12k \times 5.6k$  and the estimation of floating-point disparities reduce the quantization effects [8] of the reconstructed surface. Furthermore, a lot of scene details can be reconstructed from only two images thanks to the high resolution and a non-negligible baseline (60cm). The energy functional has a data term (squared difference of gray levels) and a smoothing term. The latter is anisotropic (It prefers depth discontinuities along image edges.) for sharp depth discontinuities and makes possible to solve the minimization problem by a gradient descent. The former is weighted to deal with occluded areas. Last a hierarchical method is used to escape from local extrema and accelerate the calculations for large images. This algorithm is also used in [56] to estimate depth maps in a case where the two images are taken by two spherical cameras (Ricoh Theta S) with a vertical baseline and a same vertical (symmetry) axis.

Urban environments are reconstructed using two catadioptric cameras mounted on a car roof [52]. Four images are considered for reconstruction: those of the left (L) and right (R) cameras taken at two consecutive time steps  $t$  and  $t + 1$ . There are three steps once the four view geometry is estimated. First a depth map is computed using SGM (Sec. 1.3) for each pair of rectified spherical images [36] among pairs L&R at  $t$ , L&R at  $t + 1$ , L at  $t$ & $t + 1$  and R at  $t$ & $t + 1$ . Then the images and depths are fused in a central virtual 360 image. This is useful to increase the reconstructible area around the car and to replace inaccurate depths near epipoles of a pair by more accurate depths in another pair. Last the virtual image is segmented into planes using assumptions about the scene: one horizontal plane for the ground, several vertical planes for walls and facades. This is useful to smooth the reconstruction and remove outliers. A set of hypothesis planes is computed by Hough voting in the virtual image, then a plane is selected for each superpixel of the virtual image by minimizing a discrete minimization problem (which has similarities to that in Sec. 1.5).

### 3.2 Cylindrical rectifications

Both spherical [36] and cylindrical rectifications are discussed in details in [8]. In both cases, the epipolar plane (which includes a point  $\mathbf{P}$  and the two camera centers  $\mathbf{C}$  and  $\mathbf{C}'$ ) is parameterized by an angle  $t \in [0, 2\pi]$ . Latitude  $p \in [0, \pi]$  in the spherical case is replaced by height  $h \in \mathbb{R}$  in the cylindrical case for a given cylinder radius (Fig. 5.b). Since many stereo methods assign integer disparities to pixels in rectified images, it is interesting to explain how this 2D discretization propagates to a 3D discretization that approximates the scene. The disparities are  $p - p'$  in the spherical case and  $h - h'$  in the cylindrical case, up to the sampling scale in the images. In the former, the iso-disparity surfaces are tori centered at the line segment  $\mathbf{CC}'$ . In the latter, these surfaces are cylinders with the same axis ( $\mathbf{CC}'$ ). Their densities in 3D increase with the disparity modulus.

In [25], cylindrical images are computed at several view-points by stitching several images taken by a camera that rotates around a vertical axis (as if a virtual multi-camera takes the original images). An advantage is a high resolution of the cylinders in spite of the low resolution cameras at this time. Once the relative poses of the cylinders are estimated, a basic (WTA) multi-baseline stereo is applied: back-project each pixel of a reference cylinder with hypothesis depths, estimate a photo-consistency in local windows by projecting the obtained 3D point to all cylinders, then select the depth with the best photo-consistency for each pixel. Here the cylinders have vertical axes and the epipolar curves (where project the 3D points) are sinusoids.

A stereo pair formed by two catadioptric cameras is introduced in [16]. Each catadioptric camera is central and composed of a parabolic mirror and an orthographic projection camera. Since they have the same (vertical) symmetry axis, the epipolar curves are simple: radial lines. The radial epipolar lines become parallel after projection of the images in cylindrical images, then a standard window-based correlation (WTA) is applied in the cylinders. This is a first step toward real-time omnidirectional stereo: 7 fps for  $600 \times 60$  pixels with 32 hypothesis depths.

Cylindrical images are computed by [9] at several view-points by cylindrical projections of images taken by a central catadioptric camera (with a hyperbolic mirror) mounted on a mobile robot, after the computation of the camera poses. Then a simple multi-baseline stereo is applied, which is similar to that in [25] (neither disparity gradient limit nor ordering). The axes of the virtual cylinders have a same direction (vertical); this accelerates the mappings between cylinders involved in the photo-consistency computation (sum of squared differences in  $11 \times 11$  windows). The inaccuracy of the reconstruction is partly due to a low resolution: the cylindrical images are  $720 \times 120$  (i.e. 2 pixels per degree in the horizontal plane) and 25 hypothesis depths per pixel of the reference cylinder.

A work [18] deals with a catadioptric camera composed of a perspective camera and two mirrors (parabolic and spherical), where rays are successively reflected. This system is

slightly non-central due to misalignment between mirrors and camera, and is approximated by a virtual central camera. Thanks to this approximation, the rectification of two input images becomes possible with parallel epipolar lines (rows in images). This implies that the axis of the virtual cylinders, where are projected the images, is parallel to the baseline between the two view points. Then a standard (unspecified) stereo method is applied.

The catadioptric stereo system in [20] is mounted on a mobile robot and composed of two hyperbolic mirrors in front of a perspective camera, that are coaxial. The mirror that is closest to the camera is projected into an annulus, it has a hole at its center to project the other mirror into a disc surrounded by the annulus. Thanks to this configuration, a scene point can have two distinct images (only one mirror reflection per ray) and a simple epipolar constraint: same radial epipolar line. The baseline, which is roughly the distance between the two mirrors, is about 20cm. Both disc and annulus parts of the catadioptric image are warped to cylinder images such that each pair of corresponding epipolar lines becomes a same column. Then a graph-cut method with a second-order prior smoothness is applied on the two cylinder images (Sec. 1.5).

### 3.3 Planar rectifications

In an early work [13], several images are taken by a catadioptric camera mounted on a mobile robot, the geometry is estimated (using a laser scanner) and a depth map is computed for each triple of consecutive images. Each pixel of the reference image (of a triple) is matched to 50 candidate points in the previous and next images along epipolar curves to define the photo-consistency volume. These computations are accelerated by local planar rectifications of the search regions. Then the depth map is computed using a graph-cut method (Sec. 1.5) with a first-order prior smoothness and by including an additional cost for each occluded (unmatched) pixel. A post-processing greatly improves the results (sub-disparity refinement, removal of epipolar neighborhood, floor correction, filling of small gaps).

Methods designed for perspective images are applied in [24] to panoramic images of Google Street View after image conversions using planar rectifications. This is not optimal due to the central assumption (Sec. 2.1), nevertheless large scale 3D models are obtained like this. Experimented methods include an improved patch-propagation (Sec. 1.6) and an improved graph-cut method that estimates a surface as a sub-complex of a 3D Delaunay triangulation (Sec. 1.7). The latter improves reconstructed thin structures by favoring inside tetrahedra where the visibility gradient is high.

A work [22] reconstructs a 360 video so that it can be replayed in VR headsets with 6 DoF, i. e. allowing not only rotations but also small translations of the user’s head. First a SfM provides the camera poses and a sparse cloud of 3D points. Then an accelerated version (in [53]) of patch propagation (Sec. 1.6) is adapted for the input spherical images. For each image, a second image in the 360 video is selected to obtain a stereo pair with a “good” baseline, that is not too large to allow matching (e. g. good common coverage) and not too small to allow reconstruction (Sec. 1.2). The selection is based on angles between reconstruction rays of same point in the sparse cloud (not too large and not too small angles). Local planar rectifications of the spherical images are used to compute the correlation score between matched points. A Samsung Gear 360 camera and an Oculus Rift are used in the experiments.

### 3.4 Sphere sweeping

The method in [23] deals with small baseline. It takes as input a short video of a bi-camera (a Ricoh Theta S in the experiments), then estimates the geometry of the image sequence, last generates a depth map in a reference image by adapting the plane sweeping method (Sec. 1.4). Here each input image concatenates two original fisheye images taken in two opposite directions. This is better for accuracy than stitched images. For each of the two

reference sub-images, the sweeping plane is replaced by a sweeping half-sphere centered on a camera center (with a discrete set of radii) to deal with the large field of view. For each point in a half-sphere obtained by back-projecting a pixel in the reference image, a photo-consistency (luminance variance) is computed by projecting the point in all images (Fig. 6). Then the depth map is improved by a method based on non-local aggregation in the cost

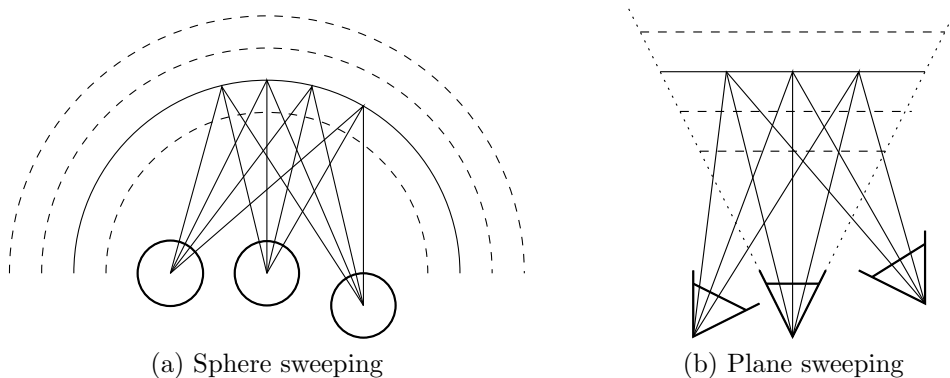


Figure 6: Sphere/plane sweeping. The bold spheres are input images. The virtual (half-)spheres, which include 3D points projected to images for photo-consistency computations, are centered on the reference image sphere. Only the current virtual sphere has a continuous pencil stroke. Plane sweeping for perspective cameras is also shown for comparison.

volume, which is better than standard local aggregation for the low textured image regions.

Another method named SweepNet in [60] deals with large baseline: four fisheye cameras ( $220^\circ$  FoV) near the four corners of the roof of minivan and pointing to four orthogonal directions. The set of sweeping spheres has 192 spheres centered at the middle of the capture system, such that their inverse radii form an arithmetic progression. For each sphere, the four images are back-projected to the sphere and a matching cost is computed by a method based on deep learning. The neural network takes as input two  $600 \times 150$  spherical images (one per back-projected image with same radius, polar regions are ignored) and predicts a cost in range  $[0, 1]$  for each pixel, which is 0 if the pixel depth is equal to the sphere radius. It includes 20 convolution layers, 5 fully connected layers, and a final sigmoid layer. The training is done with a synthetic urban dataset and a binary cross-entropy loss between ground truth Boolean (0=good depth, 1=bad depth) and the predicted cost. Then the cost volume is obtained by averaging the cost for image pairs and concatenating for each radius. Last a depth map is computed using SGM (Sec. 1.3). Thanks to the network, which uses global context of spherical image, the cost volume is cleaner (has less false positives) than those obtained by previous methods, e. g. by using zero-mean normalized cross-correlation.

### 3.5 Neither sweeping nor standard rectification

Both scene surface and relative pose from two images taken by a catadioptric camera are simultaneously estimated in [49]. The surface is defined by a 2D Delaunay triangulation in an image whose vertices are back-projected using depth parameters. Some of the vertices are located at point and edge features for a better (piece-wise planar) scene approximation. Since the camera is mounted on the roof of a car, the camera motion is restricted to be planar with an instantaneous center of rotation. The method estimates the pose parameters and the inverse depths of vertices that minimize the sum over the images of the least squares of differences of luminances at the projections of the surface in the two images. The occluded area are neglected since the two images are successive. The minimization is based on a gradient descent, that is initialized with depth inverse equals to zero (infinite surface) and

no camera motion. In the experiments, the surface is deformed along a 1.4km long trajectory forming a closed loop using 1800  $1032 \times 778$  images. (The drift is 40m.)

The approach in [45] takes as input two spherical images with small baseline and arbitrary relative rotation (taken by a Ricoh Theta S). A first estimate of the geometry is estimated by using only a matching of sparse image features. It is approximate but it allows to correct the relative rotation between the two spherical images, i. e. one of the two images is corrected by rotating its spherical coordinate system. (The color of a pixel in the rotated image is defined as follows: compute its ray direction by back-projection, multiply it by the rotation, get the color where projects the direction.) Then an optical flow method is applied between two spherical images in a more favorable experimental context where their relative motion is a small translation. The optical flow is based on deep learning and trained by using standard (non omnidirectional) images. Last the geometry is refined thanks to a greater number of correspondences and a 3D point cloud is reconstructed from the correspondences.

A method is proposed to compute a depth map between images taken by two general central cameras [50]. First, one image is pre-rotated such that the residual motion between the two images is a pure translation (as in [45]). As a consequence, pairs of corresponding epipolar curves become same curves in the two images. (This is a kind of rectification with smaller distortions near the epipoles and without the constraint that the epipolar curves are lines). Then an 1D disparity can be defined as the signed arc length between two points in such a curve. A pre-processing (curve tangent at every pixel) accelerates the computation of one of the two points from the other point and a disparity value. Last this fast computation is used to estimate the disparities for all pixels in the reference image that minimize a global cost function with a data term and a smoothing term (Sec. 1.5). The former is the absolute difference of luminances. The latter is an anisotropic smoothing that prefers depth discontinuities along image edges. An adequate minimization method is required (primal-dual algorithm) in a hierarchical scheme for large disparities. The approach runs at 10 fps for two  $848 \times 800$  videos taken by a rigid pair of fisheye cameras.

## 4 Reconstruction from only one central image

The methods summarized in Sec. 4 reconstruct an environment by taking as input only one image provided by an omnidirectional camera. We assume that the image calibration is known and that this image is obtained by stitching if a multi-camera is used. Since reconstructing from one view is more difficult than reconstructing from several views, strong prior knowledge or scene constraints are required in addition to the input image. This is a reason why these environments are only, up to now, indoor environments. Indeed the indoor environments have strong priors: horizontal floor and ceiling, vertical walls that are orthogonal most of times, doors and windows with standard heights and widths whose boundaries are horizontal or vertical.

There are two kinds of methods: those that explicitly use such geometric constraints in Sec. 4.1 and those that are based on deep learning in Sec. 4.2. The latter are supervised learning methods since they require datasets for training that collect omnidirectional images and their depth maps. The datasets can be obtained from rendering of synthetic scenes or from real scenes acquired by RGB-D cameras (e. g. [10]).

### 4.1 Explicit use of geometric constraints

In an early work [54], the user manually provides all constraints and selects points in the image to be reconstructed. First parallelism and perpendicularity constraints are used to estimate directions in the 3D coordinates system of the omnidirectional camera. The direction of parallel lines are estimated by SVD once a pair of points are clicked for each line in the image (e. g. vertical lines at the intersection of two walls). A plane normal is

known if the plane is orthogonal to a known line direction or if it is parallel to several lines with known directions. Then points selected in the image can be progressively reconstructed up to a global scale factor defined by the depth of an initial point. If we know a point in a line whose direction (or in a plane whose normal) is known then we know all points in the line (or plane) from their projections, including points that are shared by other lines (or planes) whose directions are known, and so on. Last the user draws in the image the boundaries of the reconstructed planes and obtains a simplified 3D model. Details and experiments are given for a catadioptric camera formed by a parabolic mirror and an orthographic camera.

An automatic method [46] assumes that the scene has vertical walls, horizontal ceiling and floor, and the observer’s height is known. The input is a spherical image whose zenith axis is vertical. (Every image column is back-projected as a vertical half-plane.) First the image is segmented in floor, ceiling and walls by using superpixel segmentation, texture homogeneity and vertical convexity in the image. For each  $x$ , there are  $y_c$  and  $y_f$  such that pixel  $(x, y)$  is in the ceiling or wall or floor if  $y \in [0, y_c]$  or  $y \in [y_c, y_f]$  or  $y \in [y_f, y_{max}]$ , respectively. The pixels  $(x, y_c(x))$  are back-projected on a horizontal plane at height  $h_c$ . The resulting 3D points form an approximate boundary of the ceiling if  $h_c$  is the height of the ceiling. Similarly,  $(x, y_f(x))$  are back-projected at height  $-h_f$  and we obtain an approximation of the floor boundary if  $-h_f$  is the observer height. Then the method estimates the height of the ceiling such that the two boundaries are similar after their orthogonal projections in a horizontal plane. It also estimates a common boundary projection in the horizontal plane by using Canny edge map. Last a 3D simplified room is obtained as a right cylinder whose base is the polygonized boundary and with the vertical direction. Several reconstructed rooms of a level can be registered into a single 3D model thanks to their relative poses estimated by an IMU of a standard mobile device (which also takes one spherical image per room).

## 4.2 Deep learning

The first deep learning approach that predicts a depth map from a spherical image is proposed in [62]. First a dataset composed of 35k 360 images with their depth maps (which are  $512 \times 256$ ) is generated from several previous datasets, that include both synthetic and real indoor scenes acquired by RGB-D cameras. These images are vertically aligned and do not have stitching artifacts. Then two fully-convolutional encoder-decoder architectures are proposed. One of them (RectNet) deals with the high distortions near the poles by increasing the receptive field of neurons using dilated convolutions. The training loss is a sum, for several scales, of the L2 loss between the predicted and ground-truth depth maps, and the squared moduli of depth gradients. The incomplete pixels (images and depths can have holes) are discarded in the sum to help the training. In the experiments, the previous methods (which are trained with perspective images) provide worse results, whether they are applied directly on 360 images or on individual faces of cubes obtained from 360 images.

Another approach [11] improves the previous one by enforcing piece-wise planar constraints to reconstruct indoor environments. The neural network not only predicts a depth map but also a surface normal map and a curvature map: it has an encoder-decoder architecture with two decoder branches, the first for depths and the second for normals and curvatures. It takes as input a spherical image with supplementary maps of longitude and latitude pixel coordinates, which help to distinguish the image areas with different magnitudes of distortion. Then a previous dataset is augmented by the normals and curvatures (computed from depths) and the training loss is a sum of terms to enforce piece-wise planar constraints including: a L2 loss between the predicted and ground-truth depth maps, a L2 loss between the predicted and ground-truth curvature, a L1 norm of curvature (it should be 0 except at plane boundaries) and a loss for consistency between predicted normals and predicted depths. Last a 3D model is obtained by meshing an improved version of the output depth map (with sharpened planar boundaries) thanks to the normals and curvatures.

The input of the BiFuse neural network [58] is two 360 images taken at the same view-point but with complementary projections: equirectangular (e) and cubemap (c). The e-projection mimics the peripheral vision of the human eye thanks to its large FoV but has large distortions. The c-projection mimics the foveal vision without large distortion but has discontinuities at the cube edges. The network is composed of two encoder-decoder branches, a e-branch and a c-branch, each predicts a depth map from a 360 image with the same projection. The two branches are connected at several same levels by fusion blocks for sharing advantages; each fusion block estimates a mask that decides how to combine the features of the level for the next level. The network is ended by a convolution block that combines the e-depths and the c-depths into a final equirectangular depth map. The training has three steps: first train the e-branch and the c-branch independently, then only train the fusion blocks, last train the complete network. Each step uses a L2 loss between predicted and ground truth depth maps. The method compares favorably to previous work, e. g. sharper depths at wall boundaries.

## 5 Reconstruction using stationary non-central camera

A 360 professional VR video capture system and a generic software for depth reconstruction/image rendering are presented in [48]. The capture system is a spherical rig of 16 fisheye cameras with 1m diameter (Fig. 7.a), which has a good overlap for reconstruction. (Most 3D points are visible in 6-7 cameras.) For each frame of each camera, a depth map is

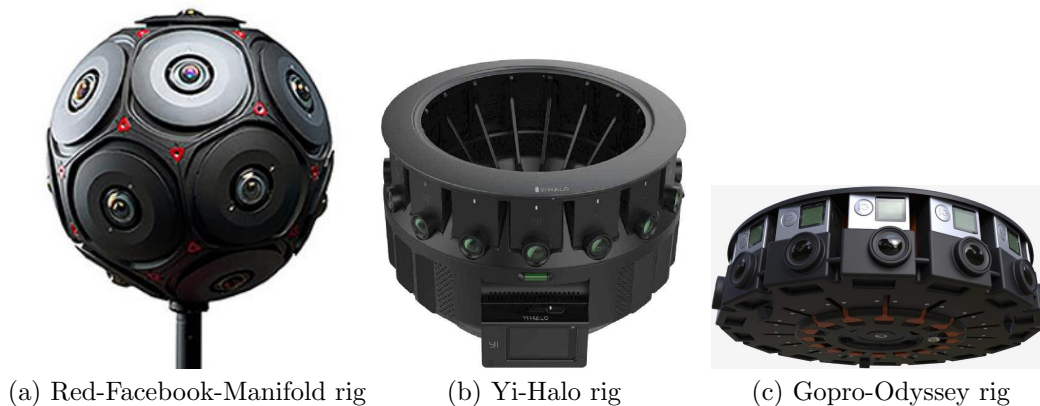


Figure 7: Stationary non-central cameras for reconstruction from videos. Every rig has 16 cameras mounted in a spherical or a cylinder scheme.

reconstructed by using a propagation method, which has similarities with those in Sec. 1.6. It includes several ingredients: image rectifications, search of 1D disparities between the reference image and many others by minimizing zero-mean sum of squared differences over small windows in a hierarchical scheme, pixel-wise propagation in a  $5 \times 5$  neighborhood, spatial filtering to smooth the disparity and preserve the edges, temporal filtering (with previous and next frames) to avoid depth flickering. If the capture system does not move, foreground masks are also used to accelerate the depth computation. They are provided by a standard background subtraction and restrict the depth computations in the detected foreground (i. e. the moving objects). The rendering step is real-time thanks to GPU and combines the original images and their depth maps in 360 images with several formats: cubemap and equirectangular projections, left-right eye renders for VR headsets.

Multi Depth Panoramas (MDP) are introduced in [37] for local view synthesis from images taken by a ring of cameras (a 360 multi-camera like the Yi-Halo in Fig. 7.b, or

Gopro Odyssey in Fig. 7.c.). MDP are several coaxial RGBDA panoramas: each cylindrical image has both depth and alpha channel maps. Many panoramas are better than a single one to render parts of the scene that are disoccluded due to translations of the viewer. They are computed as follow. First a reduced version of MDP is computed for each reference image and four neighboring images in the ring by using a deep learning method designed for standard (non-omnidirectional) images. A reduced version is composed of several fronto-parallel planes (for the reference image) with RGBA images. The neural network for learning is not 2D but 3D and takes as input a plane sweep volume, i. e. a set of images obtained by warping the input images to parallel planes with several hypothesis depths (their inverses form an arithmetic progression). It is trained from photo-realistic images synthesized by 21 large-scale (indoor and outdoor) scenes. Then the reduced versions of MDP are merged to MDP. The 3D space is divided in a few coaxial cylindrical regions (about 5), and in each of them, the parts of the reduced versions are merged to one RGBDA panorama using cylindrical coordinates and alpha compositing. Last a novel image is synthesized from MDP by projecting the panoramas as point clouds in a back-to-front order using alpha compositing and z-buffering.

## 6 Reconstruction by a moving camera

This section surveys works that do not focus on the dense stereo step from a few images (This is done in Sec. 3.) but in other steps of the reconstruction. The methods in Sec. 6.1 mostly merge several local 3D models (e. g. a depth map in a reference image, or a mesh approximation of it) to a global 3D model of the scene. Secs. 6.2 and 6.3 summarize methods that reconstruct a surface from a cloud of 3D points reconstructed from sparse features (points and/or edge curves) detected and matched in the images. Such sparse approaches are interesting for several reasons: to obtain compact models of large scale environments, to avoid computationally expensive dense stereo (e. g. for applications that do not need a high level of details), or to initialize dense stereo in other cases. Last Sec. 6.4 provides a list of commercial and open source software packages. All these works reconstruct a rigid environment from a set of omnidirectional images taken by a moving camera.

### 6.1 From local to global models

3D models of outdoor environments are calculated from hundreds of still images taken by a catadioptric camera (Fig. 8.b) mounted on a monopod in [30]. Once the geometry of the image sequence is computed by SfM including self-calibration [29] (Fig. 8.a), a local 3D model is computed for each image taken as reference (Fig. 8.c), then the local models are combined to a global 3D model (Fig. 8.d). A local model is computed using planar rectifications (to reduce image distortions) and a pair-wise stereo method as follows. Two catadioptric images are projected to two virtual cubes such that the epipolar lines are parallel except for the faces that contain the epipoles (at the face centers). The match propagation (Sec. 1.6) is applied to each of the six pairs of cube faces that are parallel. Once matches are obtained like this for several images pairs that include the reference image, a cloud of 3D points is estimated by ray-intersection of the matches. Furthermore, the reference (catadioptric) image is over-segmented by a 2D mesh that has the following properties: its triangles back-project to similar solid angles and should have low image gradient in their interior, some of its edges approximate the strongest image contours. These triangles are back-projected to approximate the 3D point cloud. This involves several operations including weighted least squares to fit a plane from 3D points, connections between adjacent triangles, and hole filling. Now the representation of the environment by the set of the local models is highly redundant since there is one omnidirectional local model for each input image. Since scene points are reconstructed several times at different accuracies, the triangles with

the worst accuracies are removed. These operations are based on a generic covariance (It does not depend on the camera model.) which is related to the standard covariance of the ray-intersection problem.

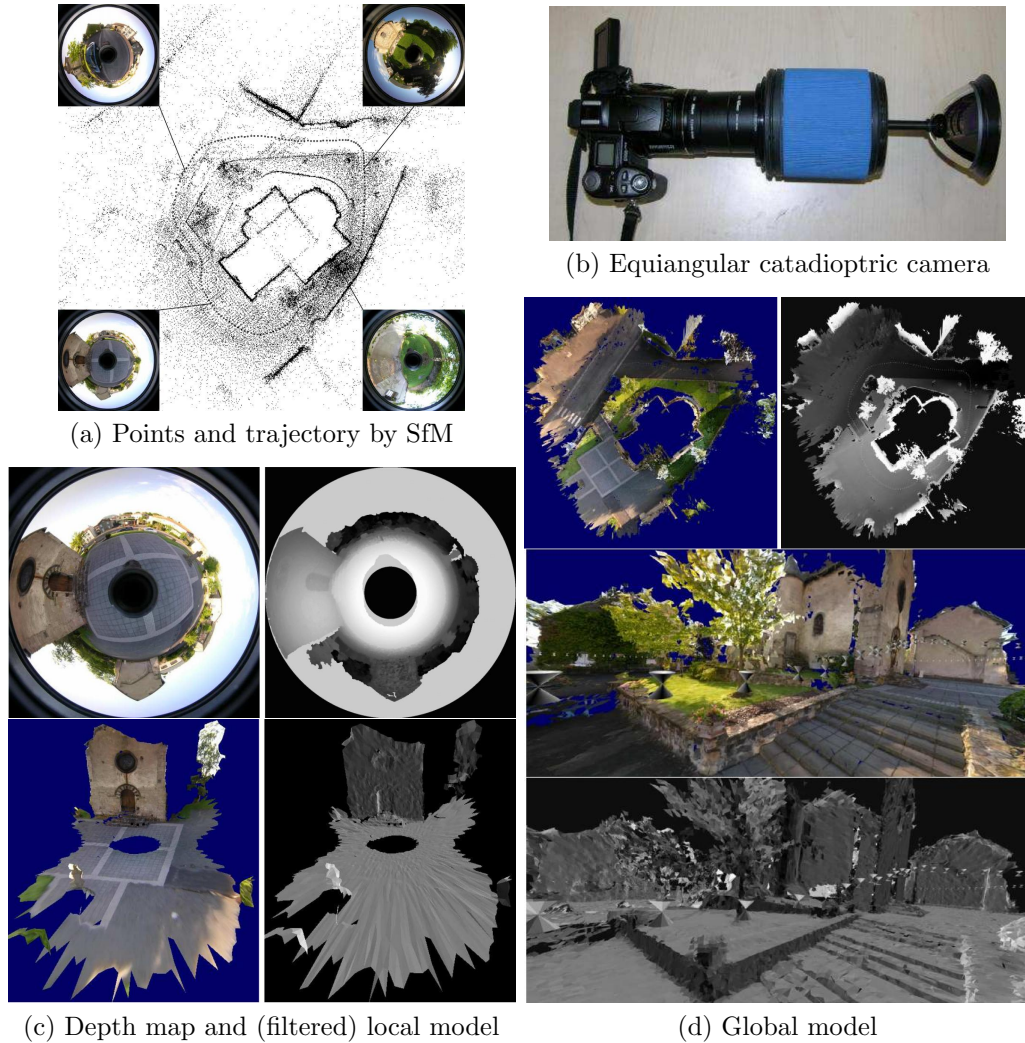


Figure 8: From local to global models [30]. 208  $1632 \times 1224$  images are taken all around a church using a catadioptric camera and reconstructed by SfM. Each local model (one per image) is a list of triangles approximating a depth map. Their union forms a global model.

In [40], a graph of RGBD spherical panoramas (with saliency) is computed from videos taken by a multi-camera system mounted on the roof of a car. Each graph edge has a 6 DoF transform that is the relative pose between the two panoramas at its two vertices. These transforms and the selection of multi-camera poses, where are computed the panoramas, are provided by a variant of SfM (visual odometry). The multi-camera system is composed of two triplets of fisheye cameras with a vertical baseline (3 on the top and 3 on the bottom). Each triple is rectified to a spherical panorama using central approximation such that the epipolar lines are same vertical lines. A dense stereo method like SGM (Sec. 1.3) provides the depth maps. In the experiment, the graph representation is estimated for a 1.5km long trajectory. 310 RGBD panoramas are computed from  $7364 \times 6$  images. Then the graph representation is applied to two tasks: real-time localization and view synthesis. The former

can estimate the pose of every common camera (monocular, stereo, and omnidirectional) from one image. In the latter, the RGBD sphere whose location is the closest to the virtual viewpoint is used to generate the virtual view.

Both urban and indoor scenes are reconstructed and simplified using cuboids and the Manhattan assumption in [28], i. e. by assuming that the scene is piecewise-planar with only three orthogonal directions for the plane normals. First spherical images with depth maps are computed as in [27] (Sec. 3.1). Second each spherical image is projected to a unit cube whose edges have the three orthogonal directions. This implies that the image edges, that correspond to edges between planar pieces of the Manhattan scene, have only three configurations in the cube faces: horizontal direction, vertical direction, or in a line that crosses the center of the cube face. Thus the method estimates the rotation, between spherical and cube poses, that maximizes the image edges in these three configurations. Third the cube faces are segmented into planar regions whose boundaries are preferred in the three configurations. Fourth planar regions with rectangular boundaries are reconstructed from the segmentation and the depth map by least squares and followed by several refinements. Fifth the reconstructions are registered in a same 3D coordinate system. Thanks to the second step, only relative translations between the cube poses need to be estimated. This is done by matching SURF features between images and estimating the translations between the corresponding 3D points. Last the scene is estimated by an union of cuboids, each of them tries to complete a pair of connected and orthogonal rectangles.

A method [47] reconstructs a 3D model of rooms in a same level from several spherical images (taken by a Ricoh Theta S in the experiments). First a simplified room is reconstructed for each image by using a method similar to that in [46] (see Sec. 4.1). Second the simplified rooms are registered in a same 3D coordinate system by SfM. Each of them is a cylinder whose base is a horizontal polygon and axis is the vertical direction. Furthermore, they have the same horizontal plane as floor, and similarly for ceiling. Since room may require more than one view, a third step merges the simplified rooms that share at least 20% of their bases. Last the user can interactively detect and localize room components mounted on the walls/ceiling/floor (e. g. electric outlets, light switches and air-vents) by 2D search using sliding windows in the back-projected texture of the rooms.

## 6.2 Sparse approaches for local models

In [26], a SfM tracks and reconstructs points and lines in a series of catadioptric images taken in a piece-wise planar environment. Lines are useful for low textured scenes, which have a small number of reliable tracked interest points. Then a 2D Delaunay triangulation is computed in a reference image whose vertices are points with 3D information, i. e. a reconstructed point or a point in a reconstructed line. The textured 3D model of the scene is initialized as the back-projected triangulation using texture of the reference image. However the Delaunay edges can be inconsistent with the scene, e. g. if a Delaunay edge crosses the projection of a scene edge separating two planar regions of the scene with different colors. This provides visual artifacts when the 3D model is seen from a view point that is not that of the reference image. Such artifacts are removed by a method based on edge flipping of the 2D triangulation [42]. This method symmetrically uses two images of the input sequence (including the reference image) and does not need threshold tuning. In short it tries flip edges to minimize an image error, i. e. it replaces two adjacent triangles **bca** and **bcd** by two other adjacent triangles **adb** and **adc** such that a discrepancy between a first image and the reprojection of the other image on the first image decreases.

A coarse 3D model of an environment is reconstructed by using a smartphone [44] in three steps: first compute a few panoramic images by rotating the smartphone while video acquisition, then robustly match and reconstruct feature points between the panoramic images, last reconstruct a surface by using a 3D Delaunay triangulation (Sec. 1.7). The first step uses the central assumption for the generation of a panorama (without tripod), since

the distance between scene and camera is quite larger than the distances between different camera centers. Each panorama is done in a few tens of seconds, both rotation estimation and image stitching are real-time (30Hz). The second step pre-rotates the panoramas around their axis (roughly vertical) to reduce the search space for matching and avoid false matches due to repetitive textures. SfM starts by detecting features points in each panorama and ends by a bundle adjustment for estimating both panorama poses and 3D points. The last step computes the 3D Delaunay triangulation of the reconstructed points. Then the list of outside tetrahedra grows in the list of all Delaunay tetrahedra. (The Delaunay tetrahedra are “carved”.) The first outside tetrahedra are finite and include the camera centers (since the camera centers are in the convex hull of the reconstructed points in the omnidirectional context). Last neighboring tetrahedra are progressively added to the set of outside tetrahedra if their intersections by rays are strong enough in a probabilistic sense. In practice,  $2048 \times 512$  panoramas are generated from  $320 \times 240$  videos, the second and third steps take about 15-25s for 3-7 panoramas.

### 6.3 Sparse approaches for global models

Simplified 3D models of outdoor environments are computed from catadioptric images [35] by using a manifold method (Sec. 1.7) in two cases: still images taken by a hand-held camera mounted on a monopod (Fig. 8.b), or a video taken by a helmet-held camera (Fig. 3.a). The computation starts by adequate SfM [29] or [41], respectively. The output of the latter is similar to that of the former: still images automatically selected from the video and a sparse cloud of points reconstructed from tracked Harris corners. The next computations are the same. The 3D Delaunay triangulation is computed from input points (including a ray intersection counter for each tetrahedron) and the set  $O$  of outside tetrahedra continuously grows in the free-space while maintaining the manifold property on the surface  $\partial O$ . Since a point has bad accuracy if it is roughly collinear to the camera locations that reconstruct it, a pre-filtering is needed. (Reject a point if all angles formed by its reconstruction rays are below a threshold.) This degenerate case is not negligible since epipoles are always reprojected in omnidirectional images. Then the growing has several steps. First the main part of  $O$  is obtained by adding tetrahedra one-by-one to  $O$  without topology change:  $\partial O$  is homeomorphic to a sphere. However this is insufficient if the camera trajectory has a complete loop around a large obstacle like a building. Indeed  $\partial O$  needs a handle like a torus. Thus a second step solves this problem by adding to  $O$  several tetrahedra at once. Now spurious handles can occur and a third step removes the largest ones. The third step has a detect-force-and-repair strategy which is summarized as follow: get the largest edges of triangles in  $\partial O$ , for each one try to force in  $O$  the free-space tetrahedra that share the edge and to grow  $O$  such that  $\partial O$  becomes manifold again. The two cases are experimented: a surface with 152k triangles from 343 still images around a church, a surface of a city with 416k triangles from 2.5k images selected in a video taken at a 1.4km long trajectory.

The manifold method above is improved in several ways and experimented on videos taken by a helmet-held multi-camera [32]. The first step is a SfM designed for such a camera system [43]: with synchronization and self-calibration and keyframe selection, without image stitching. To capture more scene details, the sparse cloud of 3D points is completed by reconstructing both Harris and curve (Canny edge) points, that are matched between the selected images using the match propagation (Sec. 1.6) and the epipolar constraint. Then a 3D Delaunay triangulation is computed from pre-filtered points and  $O$  grows in the free-space tetrahedra for maximizing the visibility consistency. This growing is not only regularized by manifoldness but also by lowered genus. Remind that the genus is the number of handles of  $\partial O$ , and it is provided by the Euler formula. The lower the genus, the simpler the topology of  $\partial O$ . The main idea is to apply operations that can increase the genus (they are similar to those in steps two and three of [35]) only if they generate visually non-negligible updates of  $\partial O$ . Furthermore, the previous repair operation is replaced by a new one that reduces

the risk of failure since it is guided by an analysis of non-manifold vertices and edges. (The new one is also quite faster than the previous one.) Other improvements include an acceleration of manifold tests and better escapes from local extrema. In the experiments, the input is a set of four  $1280 \times 960$  videos at 100Hz taken by walking during 16 minutes in an university campus using a do-it-yourself multi-camera: four Gopro Hero3 cameras mounted on a helmet (Fig. 3.c). The 3D model has 4.2M triangles and is reconstructed using 3.4k keyframes in less than 1 minute on a laptop (after the computation of the 3D points). Here a keyframe is a concatenation of four frames, one per Gopro camera, that have frame-accurate synchronization.

The most notable mistakes of the surface reconstruction methods from sparse point cloud based on 3D Delaunay triangulation are falsely-labeled free-space tetrahedra. They occur with both graph-cut- and manifold-based methods (Sec. 1.7) and are due to bad points or lack of points. They appear in incomplete shapes like 3D box (e. g. buildings) with spurious concavities, thin 1D structure (e. g. posts) that is disconnected, thin 2D structure (e. g. traffic signs) with tunnels connecting both sides. Both [31] and [33] improve these methods thanks to a pre-processing or a post-processing and experiment from input videos taken by a 360 camera. They complete the shapes, i. e. relabel matter falsely-labeled tetrahedra, by using a prior: the predominance of vertical structures in usual environments. This means that the relabeling from free-space to matter is more important in the vertical direction than in the horizontal ones. Thin 1D structures are completed [31] in a pre-processing by using reconstructed curves from image gradient. First structures are detected in the 3D Delaunay triangulation as connected sets of vertices coming from vertical curves, that have small matter neighborhood in the horizontal directions. Then series of tetrahedra are forced to matter along vertices of connected sets. Last graph-cut and manifold methods can be applied with slight modifications. Several completions in [33] deal with the 1D/2D/3D structures above as pre-processing of the manifold methods or post-processing of the graph-cut methods. In short, they increase the local-convexity of the shapes: they select a lot of sets of free-space tetrahedra, then all tetrahedra in a set are relabeled matter if there are enough matter tetrahedra that surround the set. Several criteria are used to select the sets and to quantify their matter surrounding, most of them encourage the matter completion in the vertical direction. In the experiments of the papers, the input videos are two  $2496 \times 2496$  videos at 30Hz provided by a Garmin Virb 360 camera with two opposite fisheye pointing on the left and the right (Fig. 4). City and town segments are reconstructed by mounting the 360 camera on a helmet or a car and by walking/biking/driving along trajectories up to a few kilometers. Textured 3D models, that are computed by methods in [32] and [33], are also available for interactive exploration using common consumer-grade VR headsets (PC VR <sup>1</sup>, Oculus Quest and Go <sup>2</sup>, SteamVR workshop <sup>3</sup>) or without VR <sup>4</sup>. One example is detailed in Fig. 9 for a semi-medieval city reconstructed by a helmet-held Gopro Max 360 camera.

The methods previously summarized in Sec. 6.3 are batch methods: they compute a surface once the complete image sequence and its geometry are available. They cannot be used for online applications, that need an updated surface during the video acquisition. For this reason, incremental versions of batch methods are developed by taking advantage of both incremental 3D Delaunay triangulation and incremental SfM. A method [38] locally updates the manifold surface  $\partial O$  encoded in a 3D Delaunay triangulation using new points reconstructed by an incremental SfM [41], and experiments on videos taken by an omnidirectional camera. These new points cannot be directly added to the Delaunay, otherwise  $\partial O$  becomes non-manifold and it is difficult to correct this while maintaining a Delaunay triangulation. Thus the idea is to shrink  $O$  before adding the new points, such that  $\partial O$

<sup>1</sup><https://maximelhuillier.fr>

<sup>2</sup><https://sidequestvr.com/user/330664>

<sup>3</sup><https://steamcommunity.com/profiles/76561198051374313/myworkshopfiles/>

<sup>4</sup><https://sketchfab.com/flymax63>

is still manifold and  $O$  does not include a tetrahedron that will be destroyed by the add of points. The shrinking is nothing but an inverted growing with similar operations. It pushes  $O$  to be out of a bounding ball that contains all destroyed tetrahedra. Such a ball is computed thanks to moderate constraints enforced on the new points (bounded distance to the new camera center) and the Delaunay (bounded tetrahedron diameter thanks to Steiner vertices in a regular grid). Once  $O$  shrinks, the new points are added to the Delaunay, the set of free-space tetrahedra is locally updated with their ray counters (mostly by using the new reconstruction rays provided by SfM), and last  $O$  grows with similar operations as in the batch versions. This method is improved in [39] by accelerating the most time consuming operations: the removal of artifacts including spurious handles. A previous version [61] of these works replaces the shrinking by handling of several levels of sets  $O$  at different times, but its time complexity is not bounded if the SfM closes a loop. In the experiments, both [38] and [39] take as input six  $1024 \times 768$  videos at 15Hz taken by a 360 multi-camera (PointGrey Ladybug in Fig. 3.b) mounted on the roof of a car that moves on a 2.5km long trajectory in a city. The input video in [61] is taken by a hand-held equiangular catadioptric camera (a mirror in front of the Canon Legria HSF10 in Fig. 3.a) and by walking on a 800m long trajectory in a city.

## 6.4 Available software

This section provides a list of commercial and open source software packages that reconstruct environment from omnidirectional images in 2020. The list can become non exhaustive with time. It is restricted to software that computes a dense cloud of 3D points or a triangulated surface. It does not include software that only computes SfM. (SfM is not the topic of the current chapter.)

If the omnidirectional camera is a multi-camera, most of the software packages assume that the input images are stitched. Thus they implicitly assume that both central approximation and calibration are accurate enough for stitching and the downstream 3D computations. A minority of them can take as input original images of multi-cameras and benefit from the assumption of constant relative pose(s) between its monocular component cameras. In this last case, they estimate the relative pose(s) and are the best solution for the geometry accuracy. The catadioptric images could also be used after cylindrical or spherical reprojections. Furthermore, almost all of the listed software packages take as input still images. If the input is a video, the user needs to select images in the video, e.g. one image every second by using a tool like FFmpeg [12]. Alternatively, one can use SfM software designed for multi-cameras to select still images and try to convert the geometry output to the input of a listed software.

The listed software packages are mostly designed for images taken by standard cameras, but have options for omnidirectional images. Metashape [2] can take as input equirectangular and cylindrical images, but recommend to process the original images by a Metashape function instead of stitching by another software. In [7], both Pix4Dmapper and Photoscan are experimented with images taken by the Xiaomi Mijia Sphere 360 camera. 3DF Zephyr [1] is one of the only software package that can take a video as input. It also converts equirectangular images into cubemaps for 3D computations, i.e. it does as if six pinhole cameras are used (one per face cube). In [6], both ContextCapture and PhotoScan are experimented on the Samsung Gear 360 camera. Photoscan is also experimented in [55] to input 360 videos (taken by Xiaomi Mi Sphere) sampled at different rates and the results are compared to ground truth. There are also open source software packages: MicMac [51] and Meshroom [3]. The former accepts a lot of camera models including spherical cameras. Furthermore it can take as input original images of a multi-camera if their relative poses are rotations (estimated by bundle adjustment). It includes SGM (Sec. 1.3) among other dense stereo methods. The latter accepts as input the original images of a multi-camera. It includes SGM and a surface reconstruction [24] that improves a method based on graph-cut

and 3D Delaunay triangulation (Sec. 1.7).

## 7 Conclusion

This chapter surveys the previous work on the reconstruction of environments from omnidirectional images, in the form of a dense cloud of points or a set of triangles (without focusing on the geometry estimation step). The publication rate in this topic accelerates during years 2015-2020 for several reasons including the apparition on the market of a lot of consumer-grade 360 cameras. Since the experimental context varies a lot, the methods are classified by their assumptions and input: a few images for dense stereo, one central image with strong (indoor) scene prior, videos taken by a stationary non-central camera (the environment can be non-rigid here), and a lot of still or video images taken by a moving camera. Most of them are adapted from popular methods for perspective cameras, which are reminded at the beginning to obtain a more self-contained chapter. The summaries of the methods include principle of algorithms, explicit assumptions, target applications and datasets used in experiments.

Thanks to both hardware improvements (resolutions and frame rates increase) and software improvements (e. g. deep learning), one can guess that the consumer-grade 360 cameras will be progressively used to reconstruct large scale and complete environments from terrestrial imagery with a minimal effort of the user, in a similar way as the standard cameras for reconstructing limited objects likes statues or single facade. Up to 2020, very few convincing results are shown on the internet (e. g. in Youtube and Sketchfab) in spite of the available software and potential applications such as VR. Alternatives to the pure image approach can also appear, as the use of solid-state LiDAR for robotic applications.

## References

- [1] 3DFlow. 3dflow academy - videotutorial 11- using 360 cameras, 2018. [https://www.youtube.com/watch?v=\\_SDa2GfFCUg](https://www.youtube.com/watch?v=_SDa2GfFCUg) , last accessed on 2021/04/05.
- [2] Agisoft. Metashape, professional edition, 2021. <https://www.agisoft.com/downloads/user-manuals/> , last accessed on 2021/04/05.
- [3] AliceVision. Meshroom, 2021. <https://alicevision.org/> , last accessed on 2021/04/05.
- [4] D. Anguelov, C. Dulong, D. Filip, C. Frueh, S. Lafon, R. Lyon, A. Ogale, L. Vincent, and J. Weaver. Google street view: capturing the world at street level. *IEEE Computer*, 43(6), 2010.
- [5] Z. Arican and P. Frossard. Dense disparity estimation from omnidirectional images. In *IEEE conference on advanced video and signal based surveillance*, 2007.
- [6] L. Barazzetti, M. Previtali, and F. Roncoroni. 3d modelling with the samsung gear 360. In *The international archive of photogrammetry, remote sensing and spatial information sciences, volume XLII*, 2017.
- [7] L. Barazzetti, M. Previtali, and F. Roncoroni. Can we use low-cost 360 degree cameras to create accurate 3d models ? In *The international archive of photogrammetry, remote sensing and spatial information sciences, volume XLII*, 2018.
- [8] B. Bartczak, K. Koser, F. Woelk, and R. Koch. Extraction of 3d freeform surfaces as visual landmarks for real-time tracking. *Journal of real-time image processing*, 2(2), 2007.

- [9] R. Bunschoten and B. Krose. Robust scene reconstruction from an omnidirectional vision system. *IEEE transactions on robotics and automation*, 19(2), 2003.
- [10] A. Chang, A. Dai, T. Funkhouser, M. Halber, M. Niessner, M. Savva, S. Song, A. Zeng, and Y. Zhang. Matterport3d: learning from rgb-d data in indoor environments. In *3DV*, 2017.
- [11] M. Eder, P. Moulon, and L. Guan. Pano popups: indoor 3d reconstruction with a plane-aware network. In *3DV*, 2019.
- [12] FFmpeg. Ffmpeg, 2021. <https://ffmpeg.org> , last accessed on 2021/04/05.
- [13] S. Fleck, F. Busch, P. Biber, W. Strasser, and H. Andreasson. Omnidirectional 3d modeling on a mobile robot using graph cuts. In *ICRA*, 2005.
- [14] Y. Furukawa and C. Hernandez. Multi-view stereo: a tutorial. *Foundations and Trends in Computer Graphics and Vision*, 9(1), 2015.
- [15] D. Gallup, J.M. Frahm, P. Mordohai, Q. Yang, and M. Pollefeys. Real-time plane-sweeping stereo with multiple sweeping directions. In *CVPR*, 2007.
- [16] J. Gluckman, S.K. Nayar, and K.J. Thoresz. Real-time omnidirectional and panoramic stereo. In *DARPA image understanding workshop*, 1998.
- [17] T. Gokhool, R. Martins, P. Rives, and N. Despres. A compact spherical rgb-d keyframe-based representation. In *ICRA*, 2015.
- [18] J.J. Gonzales-Barbosa and S. Lacroix. Fast dense panoramic stereovision. In *ICRA*, 2005.
- [19] R. Hartley and A. Zisserman. *Multiple View Geometry*, chapter Epipolar Geometry and the Fundamental Matrix. Cambridge University Press, 2000.
- [20] L. He, C. Luo, F. Zhu, Y. Hao, J. Ou, and J. Zhou. Depth map regeneration via improved graph cuts using a novel omnidirectional stereo sensor. In *ICCV*, 2007.
- [21] H. Hirschmuller. Accurate and efficient stereo processing by semi-global matching and mutual information. In *CVPR*, 2005.
- [22] J. Huang, Z. Chen, D. Ceylan, and H. Jin. 6-dof vr videos with a single 360-camera. In *IEEE virtual reality*, 2017.
- [23] S. Im, H. Ha, F. Rameau, H.G. Jeon, G. Chloe, and I.S. Kweon. All-around depth from small motion with a spherical panoramic camera. In *ECCV*, 2016.
- [24] M. Jancosek. *Large scale surface reconstruction based on point visibility*. PhD thesis, Czech technical university, Prague, 2014.
- [25] S.B. Kang and R. Szeliski. 3d scene data recovery using omnidirectional multibaseline stereo. *IJCV*, 25(2), 1997.
- [26] R. Kawanishi, A. Yamashita, and T. Kaneko. *Recent advances in mobile robotics*, chapter Three-dimensional environment modeling based on structure from motion with point and line features by using omnidirectional camera. InTech, 2011.
- [27] H. Kim and A. Hilton. 3d scene reconstruction from multiple spherical stereo pairs. *IJCV*, 104(1), 2013.
- [28] H. Kim and A. Hilton. Block world reconstruction from spherical stereo image pairs. *CVIU*, 139, 2015.

- [29] M. Lhuillier. Automatic scene structure and camera motion using a catadioptric system. *CVIU*, 109(2), 2008.
- [30] M. Lhuillier. A generic error model and its application to automatic 3d modeling of scenes using a catadioptric camera. *IJCV*, 91(2), 2011.
- [31] M. Lhuillier. Improving thin structures in surface reconstruction from sparse point cloud. In *ECCV Workshop*, 2018.
- [32] M. Lhuillier. Surface reconstruction from a sparse point cloud by enforcing visibility consistency and topology constraints. *CVIU*, 175, 2018.
- [33] M. Lhuillier. Local convexity reinforcement for scene reconstruction from sparse point clouds. In *IC3D*, 2019.
- [34] M. Lhuillier and L. Quan. Match propagation for image-based modeling and rendering. *IEEE TPAMI*, 24(8), 2002.
- [35] M. Lhuillier and S. Yu. Manifold surface reconstruction of an environment from sparse structure-from-motion data. *CVIU*, 117(11), 2013.
- [36] S. Li. Real-time spherical stereo. In *ICPR*, 2006.
- [37] K.E. Lin, Z. Xu, B. Mildenhall, P.P. Srinivasan, Y. Hold-Geoffroy, S. DiVerdi, Q. Sun, K. Sunkavalli, and R. Ramamoorthi. Deep multi depth panoramas for view synthesis. In *ECCV*, 2020.
- [38] V. Litvinov and M. Lhuillier. Incremental solid modeling from sparse omnidirectional structure-from-motion data. In *BMVC*, 2013.
- [39] V. Litvinov and M. Lhuillier. Incremental solid modeling from sparse structure-from-motion data with improved visual artifacts removal. In *ICPR*, 2014.
- [40] M. Meilland, A.I. Comport, and P. Rives. Dense omnidirectional rgb-d mapping of large scale outdoor environments for real-time localisation and autonomous navigation. *Journal of Field Robotics*, 32(4), 2015.
- [41] E. Mouragnon, M. Lhuillier, M. Dhome, F. Dekeyser, and P. Sayd. Generic and real time structure from motion. In *BMVC*, 2007.
- [42] A. Nakatsuji, Y. Sugaya, and K. Kanatani. Optimizing a triangular mesh for shape reconstruction from images. *IEICE Transactions on information and systems*, 88(10), 2005.
- [43] T.T. Nguyen and M. Lhuillier. Self-calibration of omnidirectional multi-camera including synchronization and rolling shutter. *CVIU*, 162, 2017.
- [44] Q. Pan, C. Arth, G. Reitmayr, E. Rosten, and T. Drummond. Rapid scene reconstruction on mobile phones from panoramic images. In *ISMAR*, 2011.
- [45] S. Pathak, A. Moro, H. Fujii, A. Yamashita, and H. Asama. 3d reconstruction of structures using spherical cameras with small motion. In *16th international conference on control, automation and systems*, 2016.
- [46] G. Pintore, V. Garro, F. Ganovelli, M. Agus, and E. Gobbetti. Omnidirectional image capture on mobile devices for fast automatic generation of 2.5d indoor maps. In *WACV*, 2016.

- [47] G. Pintore, R. Pintus, F. Ganovelli, R. Scopigno, and E. Gobbetti. Recovering 3d existing-conditions of indoor structures from spherical images. *Computer and Graphics*, 77, 2018.
- [48] A.P. Pozo, M. Toksvig, T. Filiba Schragar, J. Hsu, U. Mathur, A. Sorkine-Hornung, R. Szeliski, and B. Cabral. Automatic panoramic image stitching using invariant features. *ACM Transactions on Graphics*, 38(6), 2019.
- [49] A. Pretto, E. Menegatti, and E. Pagello. Omnidirectional dense large-scale mapping and navigation based on meaningful triangulation. In *ICRA*, 2011.
- [50] M. Roxas and T. Oishi. Variational fisheye stereo. *IEEE Robotics and Automation Letters*, 5(2), 2020.
- [51] E. Rupnik, M. Daakir, and M.P. Deseilligny. Micmac-a free, open-source solution for photogrammetry. *Open Geospatial Data, Software and Standards*, 2, 2017.
- [52] M. Schonbein and A. Geiger. Omnidirectional 3d reconstruction in augmented manhattan worlds. In *IROS*, 2014.
- [53] S. Shen. Accurate multiple view 3d reconstruction using patch-based stereo for large-scale scenes. *IEEE transactions on image processing*, 22(5), 2013.
- [54] P. Sturm. A method for 3d reconstruction of piecewise planar objects from single panoramic image. In *OMNIVIS*, 2000.
- [55] Z. Sun and Y. Zhang. Accuracy evaluation of videogrammetry using a low-cost spherical camera for narrow architectural heritage: an observation study with variable baseline and blur filters. *Sensors*, 19, 2019.
- [56] J. Thatte, J.B. Boin, H. Laksham, and B. Girod. Depth augmented stereo panorama (dasp) for cinematic virtual reality with head-motion parallax. In *IEEE International Conference on Multimedia and Expo*, 2016.
- [57] H.H. Vu, P. Labatut, J.P. Pons, and R. Keriven. High accuracy and visibility-consistent dense multiview stereo. *IEEE TPAMI*, 34(5), 2012.
- [58] F.E. Wang, Y.H. Yeh, M. Sun, W.C. Chiu, and Y.H. Tsai. Bifuse: monocular 360 depth estimation via bi-projection fusion. In *CVPR*, 2020.
- [59] Wikipedia. List of omnidirectional (360-degree) cameras. [https://en.wikipedia.org/wiki/List\\_of\\_omnidirectional\\_\(360-degree\)\\_cameras](https://en.wikipedia.org/wiki/List_of_omnidirectional_(360-degree)_cameras) , last accessed on 2021/03/12.
- [60] C. Won, J. Ryu, and J. Lim. Sweepnet: wide-baseline omnidirectional depth estimation. In *ICRA*, 2019.
- [61] S. Yu and M. Lhuillier. Incremental reconstruction of manifold surface from sparse visual mapping. In *3DIMPVT*, 2012.
- [62] N. Zioulis, A. Karakottas, D. Zarpalas, and P. Daras. Omnidepth: dense depth estimation for indoors spherical panoramas. In *ECCV*, 2018.

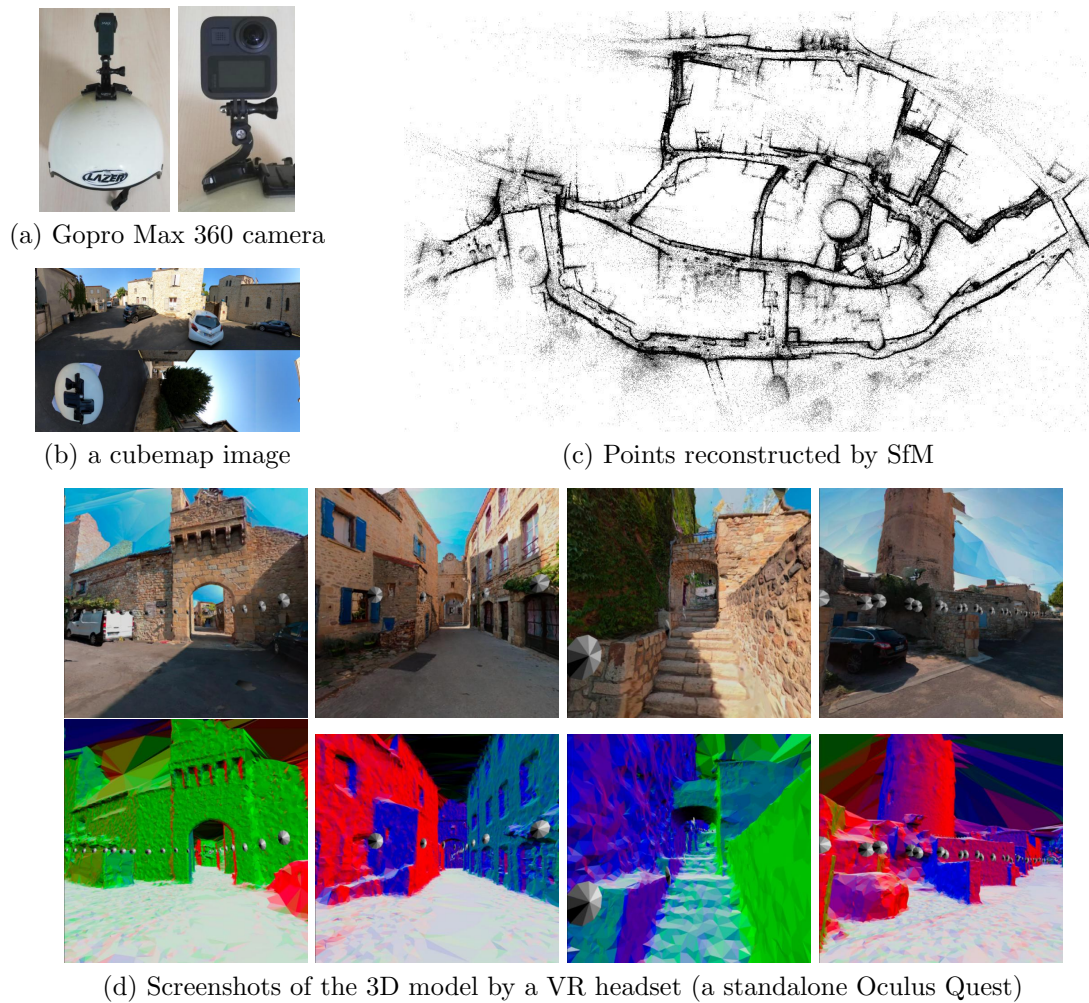


Figure 9: 3D modeling for VR. A helmet-held Gopro Max 360 camera records two  $4096 \times 1344$  videos at 30Hz by walking 15 min. in a semi-medieval city. SfM reconstructs 5.9M points and selects 1.9k keyframes. The reconstructed surface has 3.7M triangles. After simplification and texturing, it is explorable like a pedestrian using common consumer-grade VR headsets (<https://www.oculus.com/experiences/quest/5781980855168348>).