# Multi-View Stereo: A Tutorial

Yasutaka Furukawa
Washington University in St. Louis
furukawa@wustl.edu

Carlos Hernández
Google Inc.
carloshernandez@google.com

# Contents

## Abstract

This tutorial presents a hands-on view of the field of multi-view stereo with a focus on practical algorithms. Multi-view stereo algorithms are able to construct highly detailed 3D models from images alone. They take a possibly very large set of images and construct a 3D plausible geometry that explains the images under some reasonable assumptions, the most important being scene rigidity. The tutorial frames the multi-view stereo problem as an image/geometry consistency optimization problem. It describes in detail its main two ingredients: robust implementations of photometric consistency measures, and efficient optimization algorithms. It then presents how these main ingredients are used by some of the most successful algorithms, applied into real applications, and deployed as products in the industry. Finally it describes more advanced approaches exploiting domain-specific knowledge such as structural priors, and gives an overview of the remaining challenges and future research directions.

# 1

# Introduction

Reconstructing 3D geometry from photographs is a classic Computer Vision problem that has occupied researchers for more than 30 years. Its applications range from 3D mapping and navigation to online shopping, 3D printing, computational photography, computer video games, or cultural heritage archival. Only recently however have these techniques matured enough to exit the laboratory controlled environment into the wild, and provide industrial scale robustness, accuracy and scalability.

Modeling the 3D geometry of real objects or scenes is a challenging task that has seen a variety of tools and approaches applied such as Computer Aided Design (CAD) tools [3], arm-mounted probes, active methods [110, 131, 11, 10] and passive image-based methods [162, 165, 176]. Among all, passive image-based methods, the subject of this tutorial, provide a fast way of capturing accurate 3D content at a fraction of the cost of other approaches. The steady increase of image resolution and quality has turned digital cameras into cheap and reliable high resolution sensors that can generate outstanding quality 3D content.

The goal of an image-based 3D reconstruction algorithm can be described as *"given a set of photographs of an object or a scene, estimate*

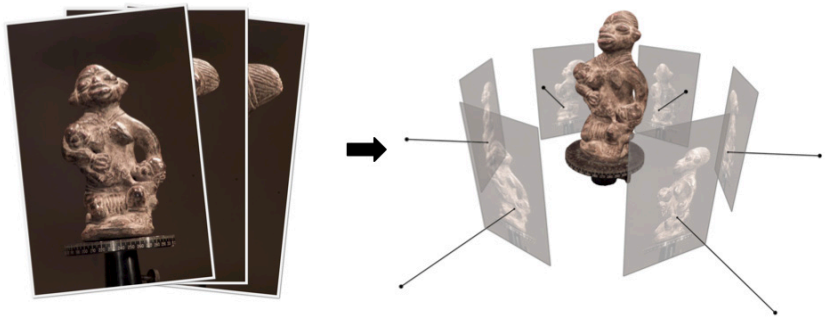


**Figure 1.1:** Image-based 3D reconstruction. Given a set of photographs (left), the goal of image-based 3D reconstruction algorithms is to estimate the most likely 3D shape that explains those photographs (right).

*the most likely 3D shape that explains those photographs, under the assumptions of known materials, viewpoints, and lighting conditions"* (See Figure 1.1). The definition highlights the difficulty of the task, namely the assumption that materials, viewpoints, and lighting are known. If these are not known, the problem is generally ill-posed since multiple combinations of geometry, materials, viewpoints, and lighting can produce exactly the same photographs. As a result, without further assumptions, no single algorithm can correctly reconstruct the 3D geometry from photographs alone. However, under a set of reasonable extra assumptions, e.g. rigid Lambertian textured surfaces, state-of-the-art techniques can produce highly detailed reconstructions even from millions of photographs.

There exist many cues that can be used to extract geometry from photographs: texture, defocus, shading, contours, and stereo correspondence. The latter three have been very successful, with stereo correspondence being the most successful in terms of robustness and the number of applications. Multi-view stereo (MVS) is the general term given to a group of techniques that use stereo correspondence as their main cue and use more than two images [165, 176].

All the MVS algorithms described in the following chapters assume the same input: a set of images and their corresponding camera parameters. This chapter gives an overview of an MVS pipeline starting from

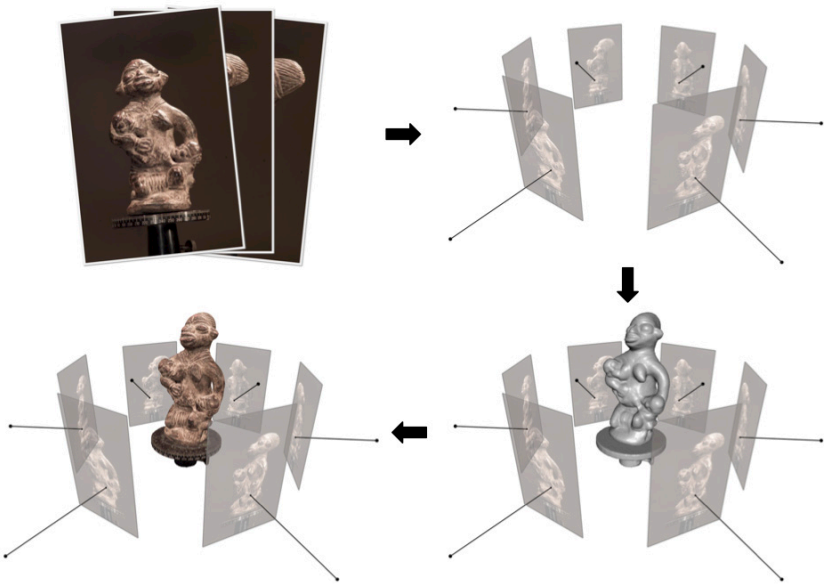


**Figure 1.2:** Example of a multi-view stereo pipeline. Clockwise: input imagery, posed imagery, reconstructed 3D geometry, textured 3D geometry.

photographs alone. An important take-home message of this chapter is simple: An MVS algorithm is only as good as the quality of the input images and camera parameters. Moreover, a large part of the recent success of MVS is due to the success of the underlying Structure from Motion (SfM) algorithms that compute the camera parameters.

Figure 1.2 provides a sketch of a generic MVS pipeline. Different applications may use different implementations of each of the main blocks, but the overall approach is always similar:

- Collect images,
- Compute camera parameters for each image,
- Reconstruct the 3D geometry of the scene from the set of images and corresponding camera parameters.
- Optionally reconstruct the materials of the scene.

**Figure 1.3:** Different MVS capture setups. From left to right: a controlled MVS capture using diffuse lights and a turn table, outdoor capture of small-scale scenes, and crowd-sourcing from online photo-sharing websites.

In the chapter we will give more insight into the first three main stages of MVS: imagery collection, camera parameters estimation, and 3D geometry reconstruction. Chapter 2 develops the notion of photo-consistency as the main signal being optimized by MVS algorithms. Chapter 3 presents and compares some of the most successful MVS algorithms. Chapter 4 discusses the use of domain knowledge, in particular, structural priors in improving the reconstruction quality. Chapter 5 gives an overview of successful applications, available software, and best practices. Finally Chapter 6 describes some of the current limitations of MVS as well as research directions to solve them.

## 1.1 Imagery collection

One can roughly classify MVS capture setups into three categories (See Figure 1.3):

- Laboratory setting,
- Outdoor small-scale scene capture,
- Large-scale scene capture using fleets or crowd-sourcing, e.g., cars, planes, drones, and Internet.

MVS algorithms first started in a laboratory setting [184, 147, 58], where the light conditions could be easily controlled and the camera

could be easily calibrated, e.g. from a robotic arm [165], rotation table [93], fiducial markers [2, 43, 192], or early SfM algorithms [62]. MVS algorithms went through two major developments that took them to their current state: They left the laboratory setting to a small-scale outdoor scenes [174, 102, 85, 169, 190], e.g. a building facade or a fountain, then scaled up to much larger scenes, e.g. entire buildings and cities [129, 153, 97, 69].

These major changes were not solely due to the developments in the MVS field itself. It was a combination of new hardware to capture better images, more computation power, and scalable camera estimation algorithms.

**Improvements in hardware:** Two areas of hardware improvements had the most impact on MVS: digital cameras and computation power. Digital photography became mainstream and image digital sensors constantly improved in terms of resolution and quality. Additionally, mass production and miniaturization of geo positioning sensors (GPS) made them ubiquitous in digital cameras, tablets, and mobile phones. Although the precision of commercial units is not enough for MVS purposes, it does provide an initial estimate on camera parameters that can be refined using Computer Vision techniques. The second significant hardware improvement was computation power. The rise of inexpensive computer clusters [5] or GPU general computation [6] enabled SfM algorithms [25, 64] and MVS algorithms [69] to easily handle tens of thousands of images.

**Improvements in Structure-from-Motion algorithms:** Researchers have been working on visual reconstruction algorithms for decades [183, 182]. However, only relatively recently have these techniques matured enough to be used in large-scale industrial applications. Nowadays industrial algorithms are able to estimate camera parameters for millions of images. Two slightly different techniques have made great progress in recent years: Structure-from-Motion (SfM) [88] and Visual Simultaneous Localization and Mapping (VSLAM) [53]. Both rely on the correspondence cue and the assumption that the scene is rigid. SfM is most commonly used to compute camera models of unordered sets of images, usually offline, while VSLAM specializes in computing the

location of a camera from a video stream, usually real-time. In this tutorial we focus on SfM algorithms, since a large majority of MVS algorithms are designed to work on unordered image sets, and rely on SfM to compute camera parameters. Note however that VSLAM has made very quick progress recently in the context of MVS [145, 180].

The term "camera parameters" refers to a set of values describing a camera configuration, that is, camera pose information consisting of location and orientation, and camera intrinsic properties such as focal length and pixel sensor size. There are many different ways or "models" to parameterize this camera configuration. In the following section, we discuss some of the most common camera projection models used in MVS applications.

## 1.2 Camera projection models

As mentioned in the introduction, MVS algorithms need additional knowledge in order to make the reconstruction problem well posed. In particular, MVS algorithms require that every input image has a corresponding camera model that fully describes how to project a 3D point in the world into a 2D pixel location in a particular image. The most commonly used camera model for MVS is the pinhole camera model, which is fully explained by a 3x4 projection camera matrix [88], defined up to a scale. This is the model commonly used with off-the-shelf digital cameras capturing still photographs. Any $3 \times 4$ projection matrix $P$ can be decomposed into the product of a $3 \times 3$ upper triangular matrix $K$ and a $3 \times 4$ pose matrix $[R|T]$

$$P = \underbrace{\begin{pmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}}_{K} \cdot \underbrace{\left(\begin{array}{ccc|c} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{array}\right)}_{R \qquad T}. \tag{1.1}$$

The matrix $K$ is commonly referred to as the intrinsics matrix, because it is composed of quantities intrinsic to the camera: vertical and horizontal focal lengths $(f_x, f_y)$, principal point $(c_x, c_y)$, and skew $s$. The matrix $[R|T]$ is commonly known as the extrinsics matrix, where $R$ is

**Figure 1.4:** Common deviations from pinhole camera model. Left: a fish eye lens exhibiting large radial distortion (top) and a rectified version of the same image after removing radial distortion (bottom). Right: rolling shutter artifacts caused by a fast moving object in the scene [155].

the rotation of the camera and $T$ is the translation of the camera. Note that, due to the quality of digital sensors, one rarely estimates the 11 parameters of the projection matrix. In particular, pixels are assumed to have no skew ($s = 0$), and be square ($f_x = f_y$). Also, if an image has not been cropped, it is safe to assume the principal point is at the center of the image. As a result, a common pinhole camera model is just composed of 7 parameters: the focal length $f$, the rotation matrix $R$ and the translation vector $T$.

If the attached lens is low quality, or wide-angle (See Figure 1.4 left), the pure pinhole model is not enough and often extended with a radial distortion model. Radial distortion is particularly important for high-resolution photographs, where small deviations from the pure pinhole model can amount to multiple pixels near the image boundaries.

Radial distortion can typically be removed from the photographs before they enter the MVS pipeline. If the radial distortion parameters of an image have been estimated, one can undistort the image by resampling as if it had been taken with an ideal lens without distortion (See

Figure 1.4 bottom left). Undistorting the images simplifies the MVS algorithm and often leads to faster processing times. Some cameras, e.g. those in mobile phones, incorporate dedicated hardware to remove radial distortion during the processing of the image just after its capture. Note however that rectifying wide-angle images will introduce resampling artifacts as well as field of view cropping. To avoid these issues MVS pipelines can support radial distortion and more complicated camera models directly, at the expense of extra complexity.

Finally, rolling shutter is another source of complexity particularly important for video processing applications (See Figure 1.4 right). A digital sensor with an electronic rolling shutter exposes each row of an image at slightly different times. This is in contrast to global shutters where the whole image is exposed at the same time. A rolling shutter often provides higher sensor throughput at the expense of a more complicated camera model. As a result, if the camera or the scene are moving while capturing the image, each row of the image captures effectively a slightly different scene. If the camera or scene motion is slow w.r.t. the shutter speed, rolling shutter effects can be small enough to be ignored. Otherwise the camera projection model needs to incorporate the effects [63].

## 1.3 Structure from Motion

There is a vast literature on Structure-from-Motion algorithms, and it is not our intention to throughly review it here. In the following, we will discuss some of the key aspects of SfM and how they relate to MVS algorithms.

SfM algorithms take as input a set of images and produce two things: the camera parameters of every image, and a set of 3D points visible in the images which are often encoded as tracks. A track is defined as the 3D coordinates of a reconstructed 3D point and the list of corresponding 2D coordinates in a subset of the input images. Most of the current state-of-the-art SfM algorithms share the same basic processing pipeline (See Figure 1.5):

- Detect 2D features in every input image.
- Match 2D features between images.
- Construct 2D tracks from the matches.
- Solve for the SfM model from the 2D tracks.
- Refine the SfM model using bundle adjustment.

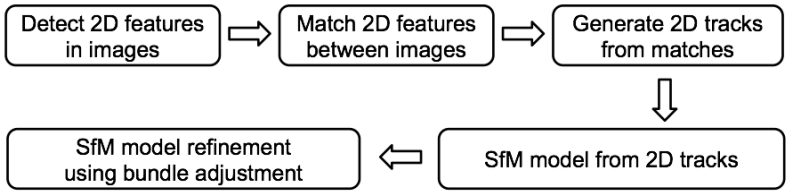



**Figure 1.5:** Main stages of a generic SfM pipeline, clockwise: feature detection, feature matching, track generation, structure-from-motion and bundle adjustment.

Initial work on SfM mainly focused on the geometry of two and three views under the assumption of a rigid scene [88]. Carlo Tomasi's technical perspective on visual reconstruction algorithms [182] presents an overview of the early work. One of the key developments for SfM was the use of RANSAC [61] to robustly estimate the epipolar geometry between two or three views given noisy matches.

Efforts then focused on two key components of the SfM algorithm: 1) computing a Euclidean reconstruction (up to a scale) from multiple cameras, that is, estimating both the camera parameters and 3D positions of the tracks, and 2) building longer 2D tracks. By the end of the 20th century, SfM algorithms were able to robustly compute models from large structured sets of images, e.g. from sequences of images or video sequences [62, 152] and the first SfM industrial solutions started to be commercialized for applications such as movie editing and special effects [4].

These initial systems were mainly designed for structured sets of images i.e., sets where the order of images matters, such as a video

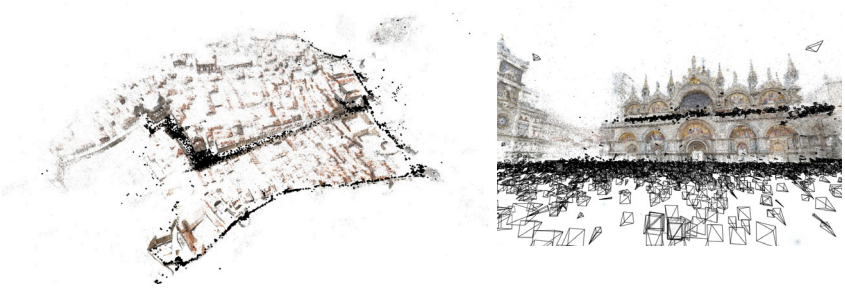


**Figure 1.6:** Large scale SfM examples from [25]. Left: SfM model of the city of Dubrovnik. Right: SfM model of San Marco Square in Venice.

sequence. Although some MVS applications can define such an order, for example, Google's StreetView [81] or Microsoft's Streetside [143], many recent MVS applications also use unordered sets of images captured at different times with different hardware, e.g. 3D maps from aerial images [108, 144, 30]. The development of fast and high quality feature detectors [87, 135, 57] and descriptors [135, 36, 159, 130, 26] was a crucial development towards making SfM work with unstructured datasets. High quality descriptors enabled building longer and higher quality tracks from images captured with very different pose and illumination.

The final ingredient to tackle large-scale SfM of unstructured photo collections was to improve the matching stage. In the case of unstructured photo collections, one does not have any prior knowledge of nearby candidate images that should be matched against. Therefore, every image has to be matched to every other image, which is computationally very expensive. Efficient indexing [146] combined with high quality descriptors allowed efficient pairwise matching of millions of images. Further work on simplifying the connectivity graph of the tracks [172] and parallelization [25, 64] lead to the current state-of-the-art SfM pipelines used in the industry, for example, Microsoft's photosynth [16] and Google's photo tours [15] (See Figure 1.6).

## 1.4 Bundle Adjustment

Although bundle adjustment [183] is not strictly a part of SfM, it is a very common step used to refine the initial SfM model. Given a set of camera parameters $\{P_i\}$, and a set of tracks $\{M^j, \{m_i^j\}\}$, where $M^j$ denotes the 3D coordinate of a track, and $m_i^j$ denotes the 2D image coordinate of its image projection in the $i_{\text{th}}$ camera, bundle adjustment minimizes the following non-linear least squares error

$$E(P, M) = \sum_j \sum_{i \in V(j)} |P_i(M^j) - m_i^j|^2. \tag{1.2}$$

$V(j)$ is the list of camera indices where point $M^j$ is visible, and $P_i(M^j)$ represents the projected 2D image coordinate of 3D point $M^j$ in camera $i$ using the camera parameters $P_i$.

$E(P, M)$ is typically measured in squared pixels, but a more common metric to express the accuracy of the estimation is to use the Root Mean Square Error or RMSE, which is measured in pixels and is defined as:

$$RMSE(P, M) = \sqrt{\frac{E(P, M)}{N}}, \tag{1.3}$$

where $N$ is the number of residual terms being summed up in (1.2). Typical RMSE values before bundle adjustment are in the order of several pixels, while values after bundle adjustment are often sub-pixel.

The bundle adjustment framework enables the combination of multiple sensors with the SfM objective in a principled optimization framework. One way to fuse GPS and IMU constraints with SfM constraints is to simply add additional terms to (1.2) that penalize deviations of $P_i$ from the predicted camera models from the GPS and IMU signals.

MVS algorithms are very sensitive to the accuracy of the estimated camera models. The reason is that, for efficiency purposes, they use the epipolar geometry defined by the camera models to restrict the 2D matching problem into a 1D matching problem (See Section 1.5 for more details). If the reprojection error is large, a pixel might never be compared against its real match, significantly degrading the MVS performance. The robustness of MVS to camera reprojection error depends mainly on how tolerant the matching criterion (namely the photo-

consistency measures presented in Chapter 2) is to misalignments. Usually, the larger the domain $\Omega$ of the photo-consistency measure (See equation 2.1), the more robust the measure is. Unfortunately, large domains also tend to produce over smoothed geometry, so there is a compromise between accuracy and robustness.

Since MVS is so sensitive to reprojection errors, bundle adjustment is often a requirement for MVS, with the goal of sub-pixel reprojection errors. Note that, because reprojection error is measured in pixels, one can downsample the input images and rescale the camera parameters until the reprojection error drops below a certain threshold. This approach will work as long as the downsampled images still contain enough texture and details for MVS to work [72].

## 1.5 Multi-View Stereo

The origins of multi-view stereo can be traced back to human stereopsis and the first attempts to solve the stereoscopic matching problem as a computation problem [139]. Until this day, two-view stereo algorithms have been a very active and fruitful research area [162]. The multi-view version of stereo originated as a natural improvement to the two-view case. Instead of capturing two photographs from two different viewpoints, multi-view stereo would capture more viewpoints in-between to increase robustness, e.g. to image noise or surface texture [184, 147]. What started as a way to improve two-view stereo has nowadays evolved into a different type of problem.

Although MVS shares the same principles with such classic stereo algorithms, MVS algorithms are designed to deal with images with more varying viewpoints, such as an image set surrounding an object, and also deal with a very large number of images, even in the order of millions. The difference in the nature of the MVS problem ends up producing significantly different algorithms than the classic stereo counterpart. As an example, industrial applications for 3D mapping [108, 144, 30], process millions of photographs over hundreds of kilometers at a time, effectively reconstructing large metropolitan areas, countries and eventually the entire world.

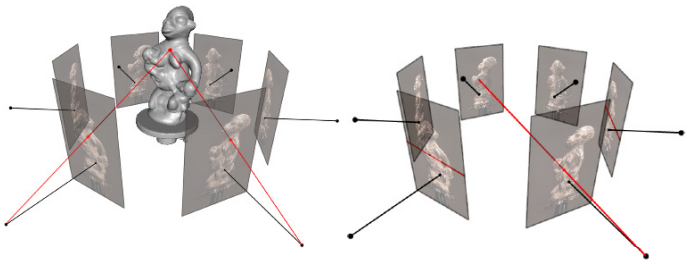


**Figure 1.7:** Matching images with known camera parameters. Left: The 3D geometry of the scene defines a correspondence between pixels in different images. Right: when camera parameters are known, matching a pixel in one image with pixels in another image is a 1D search problem.

Matching pixels across images is a challenging problem that is not unique to stereo or multi-view stereo. In fact, optical flow is another very active field in Computer Vision, tackling the problem of dense correspondence across images [33]. The main differences with MVS being that optical flow is typically a two image problem (similar to two-view stereo), cameras are not calibrated, and its main application is image interpolation rather than 3D reconstruction.

Note that in the case of MVS, where the camera parameters are known, solving for the 3D geometry of the scene is exactly equivalent to solving the correspondence problem across the input images. To see why, consider a 3D point belonging to the 3D scene geometry (See Figure 1.7 left). Projecting the 3D point into the set of visible cameras establishes a unique correspondence between the projected coordinates on each image.

Given a pixel in an image, finding the corresponding pixels in other images needs two ingredients:

- An efficient way to generate possible pixel candidates in other images.
- A measure to tell how likely a given candidate is the correct match.

If the camera geometry is not known, as is typically the case in optical flow, each pixel in an image can match any other pixel in another

image. That is, for each pixel one has to do a 2D search in the other image. However, when the camera parameters are known (and the scene is rigid), the image matching problem is simplified from a 2D search to a 1D search (See Figure 1.7 right). A pixel in an image generates a 3D optic ray that passes through the pixel and the camera center of the image. The corresponding pixel on another image can only lie on the projection of that optic ray into the second image. The different geometric constraints that originate when multiple cameras look at the same 3D scene from different viewpoints are known as epipolar geometry [88].

As for measures to tell how likely a candidate match is, there is a vast literature on how to build so called *photo-consistency* measures that estimate the likelihood of two pixels (or groups of pixels) being in correspondence. Photo-consistency measures in the context of MVS are presented in more detail in Chapter 2.

# 2

## Multi-view Photo-consistency

This chapter develops the concept of multi-view photometric consistency, or photo-consistency in short, as the main signal used in any multi-view stereo algorithm. Multi-view photo-consistency measures the agreement or consistency between a set of input photographs and all the ingredients that take part in their image formation: illumination, materials, and 3D geometry of the scene being captured. Under the right assumptions, multi-view stereo algorithms are capable of "inverting" the image formation process and producing highly detailed 3D geometry, materials, and illumination from images alone. Multi-view stereo is thus seen as a constrained optimization problem, where multi-view photo-consistency is maximized as a function of geometry, viewpoints, materials, and illumination.

Many proposed photo-consistency measures are invariant to material and illumination changes while Structure-from-Motion approaches are able to provide accurate 3D pose for each image. These advances have enabled multi-view stereo formulations where photo-consistency is optimized only as a function of 3D geometry, often under certain prior or smoothness conditions.

Section 2.1 introduces the basic concepts of **multi-view** photo-consistency and the different measures of two-view photo-consistency used in state-of-the-art multi-view stereo systems. Many of these concepts are shared with two-view stereo algorithms [162], but the term **multi-view** holds a key role in MVS and sets it apart from traditional two-view stereo algorithms.

A crucial requirement for photo-consistency measures is to compute photo-consistency on a set of images that see the **same** 3D geometry. However this visibility information is only known once the 3D geometry is available, thus creating a chicken-and-egg dependency where. In order to compute 3D geometry (by maximizing photo-consistency), one needs the correct 3D-geometry to select which images to use in order to compute photo-consistency. Section 2.2 explores different strategies that have been developed to break this dependency.

## 2.1 Photo-consistency measures

Given a set of $N$ input images and a 3D point $p$ seen by all the images, one can define the photo-consistency of $p$ w.r.t. each pair of images $I_i$ and $I_j$ as:

$$\mathcal{C}_{ij}(p) = \rho(I_i(\Omega(\pi_i(p))), I_j(\Omega(\pi_j(p)))), \tag{2.1}$$

where $\rho(f, g)$ is a similarity measure that compares two vectors, $\pi_i(p)$ denotes the projection of $p$ into image $i$, $\Omega(x)$ defines a support domain around point $x$, and $I_i(x)$ denotes the image intensities sampled within the domain. Every photo-consistency measure can be described as a particular choice of $\rho$ and $\Omega$.

Some photo-consistency measures do not need the support domain $\Omega$ to be defined while others do (see Table 2.1). The main purpose of the support domain $\Omega$ is to define the size of a region where the appearance of the scene is expected to be unique and somewhat invariant to illumination and viewpoint changes. Note that uniqueness and invariance are often two competing properties of a photo-consistency measure. The larger the domain $\Omega$ is, the more unique the local appearance inside the domain is, which makes it easier to match to other images. At the same time, the larger the domain is, the harder it is

**Table 2.1:** Summary table of different similarity measures used to compute photo-consistency.

| Measure | required $\Omega$ | invariance |
|---|---|---|
| Sum of Squared Differences (SSD) | no | none |
| Sum of Absolute Differences (SAD) | no | none |
| Normalized Cross Correlation (NCC) | yes | bias/gain |
| Census | yes | bias/gain |
| Rank | yes | bias/gain/rotation |
| Mutual Information (MI) | yes | any bijection |

to maintain the illumination and viewpoint invariance due to reflections, depth boundaries, or smooth geometry assumptions (e.g., planar assumptions).

In MVS algorithms, the simplest way to define $\Omega$ for each image is to use a square grid of pixels with constant size across the images, which works well when input images share roughly the same pixel resolution for the surface being reconstructed. The size of the domain $\Omega$ tends to be relatively small, e.g. a 3x3 or 5x5 grid of image intensities. In more complicated scenarios, where images have varying resolutions and their location is non-uniformly distributed w.r.t the scene, it becomes critical to adjust the size of the domain. The size needs to be proportional to the image resolution and the viewpoint separation, and inversely proportional to the distance to the scene. Some methods even change the shape of the domain, where a particularly successful instance of a view-dependent domain is obtained by computing the domain $\Omega$ as the projection of a 3D local patch centered around the 3D point $p$ [210, 74]. Other methods significantly increase the density of the images, which enables smaller domain sizes, an extreme case being the use of video [145] or Lightfields [195, 117].

A related concept to the domain that is very common in stereo algorithms is photo-consistency aggregation [162], which consists in spatially aggregating the photo-consistency measure to increase its robustness. It is most effective when used with measures that can be

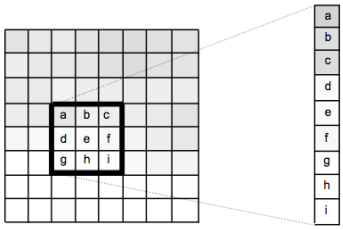



**Figure 2.1:** Example of a rectangular $3 \times 3$ domain $\Omega$ centered around a pixel with intensity $e$ in a grayscale image. The image intensity in the domain is vectorized into a 1D vector $f = (a, b, c, d, e, f, g, h, i)^\top$ in order to compute a photo-consistency measure.

computed without a domain e.g., Sum of Square Differences (SSD) and Sum of Absolute Differences (SAD), since advanced aggregation techniques can be used without over smoothing the score, (See Section 2.1.8 for more details).

Given the photo-consistency definition in (2.1) between two images, one can further define the photo-consistency of $p$ w.r.t. a single image $I_i$ by simply averaging all the pair-wise photo-consistencies

$$\mathcal{C}_i(p) = \frac{1}{N-1} \sum_{j \neq i} \mathcal{C}_{ij}(p), \tag{2.2}$$

and define the photo-consistency w.r.t. all the images by averaging the individual photo-consistencies

$$\mathcal{C}(p) = \frac{1}{N} \sum_{i} \mathcal{C}_i(p) = \frac{1}{N(N-1)} \sum_{i} \sum_{j \neq i} \mathcal{C}_{ij}(p). \tag{2.3}$$

As mentioned above, this definition of photo-consistency requires all the images to see a given point. In other words, this definition only works if one already knows what images see a given 3D point.

In the following we describe some of the most common and successful photo-consistency measures. We point interested readers to [99] for a detailed review. As a convention, we define photo-consistency measures that compare two signals $f$ and $g$, where each signal is represented by a one dimensional vector, obtained by, for example, sampling a rectangular image region in a gray-scale image (See Figure 2.1).

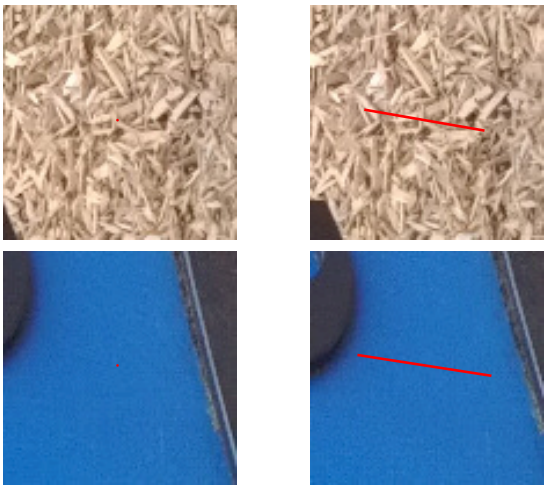


**Figure 2.2:** Example of two calibrated images used as input to a photo-consistency algorithm matching a single pixel (center of the left image) against a second image across its epipolar line. See Figure 2.3 and Figure 2.4 for a comparison of the different photo-consistency values described in this chapter. Note that the two images being compared were taken by the same camera one after the other. Top: textured case. Bottom: textureless case.

Given color images, different strategies exist to deal with the different channels:

- Convert the color image into gray scale *before* computing the photo-consistency.
- Compute the photo-consistency per color channel independently, and return the average.
- Concatenate the vectors from all the color channels into a single larger vector.

Although it is not our intention to provide a thorough evaluation, we do provide a small qualitative comparison to illustrate what are the typical signatures of the different measures for both textured and untextured surfaces (see Figure 2.2). Figure 2.3 shows the photo-consistency plots as we move along the epipolar line for the textured region of Figure 2.2

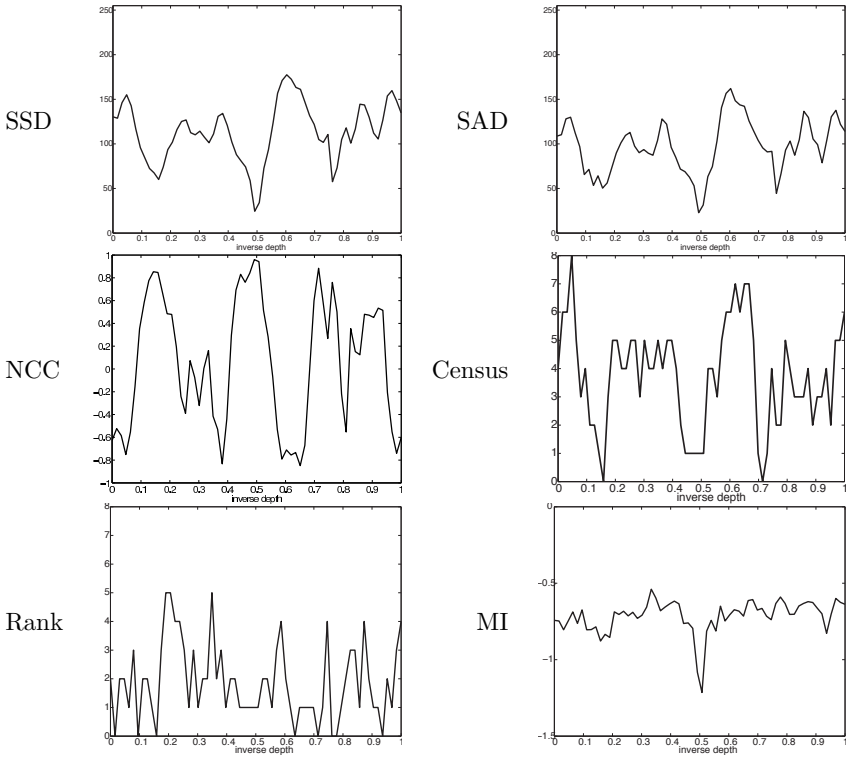



**Figure 2.3:** Different photo-consistency measures computed along the epipolar line for the textured example of Figure 2.2 top. The correct depth is roughly at 0.5 depth units. All the measures use a domain $\Omega$ 3x3 except for MI, which uses a domain of 9x9.

top. Figure 2.4 shows the photo-consistency plots for the untextured region of Figure 2.2 bottom.

### 2.1.1 Normalized Cross Correlation

Zero-mean normalized cross correlation (NCC) is one of the most common and successful photo-consistency measures used in multi-view stereo algorithms. It is invariant to changes in gain and bias and it is mainly used when lighting and material invariance is required, for example, when computing stereo from photo collections with a wide

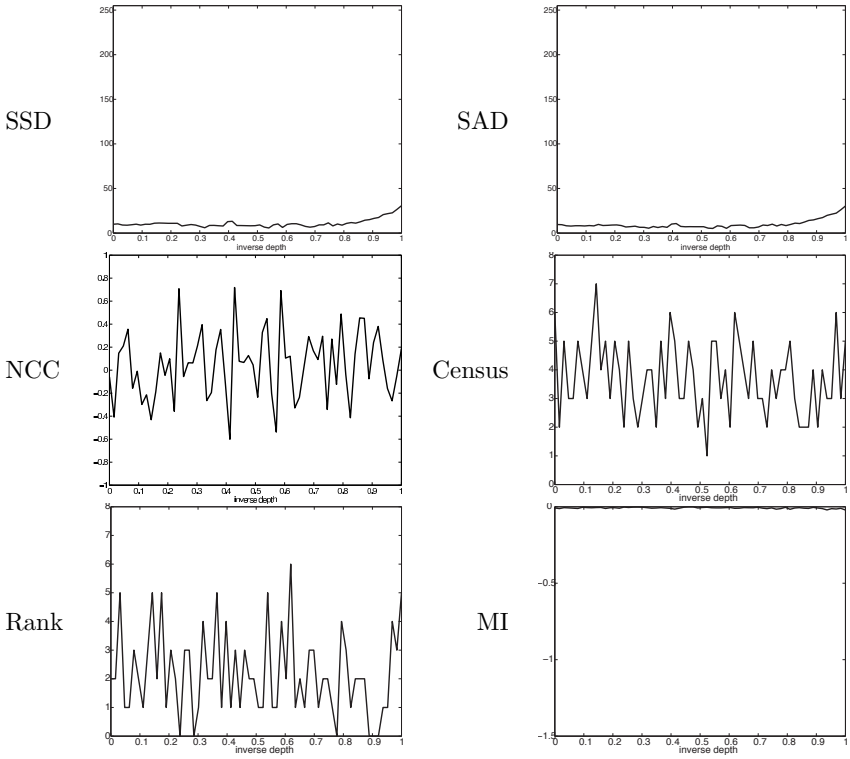



**Figure 2.4:** Different photo-consistency measures computed for the textureless example of Figure 2.2 bottom. All the measures use a domain Ω 3x3 except for MI, which uses a domain of 9x9.

range of illuminations and materials. The main failure modes of NCC are a lack of surface texture and repetitive textures, while the main advantage is its accuracy. The similarity measure is defined as:

$$\rho_{NCC}(f,g) = \frac{(f-\bar{f})\cdot(g-\bar{g})}{\sigma_f \sigma_g} \in [-1,1],$$

with $\bar{f}$ the mean of $f$ and $\sigma_f$ the standard deviation of $f$.

The handling of color images in NCC computation requires extra attention. It does not work well to simply concatenate all the color channels into a single vector and apply the above formula. In difficult cases such as homogeneous textured surfaces, the primary visual cues

are often subtle shadings and shadowing effects on its surface. We want NCC to capture subtle spatial intensity variation in each color channel, which are much smaller than the intensity variation across color channels. With the simple concatenation, NCC amounts to only capturing the intensity variation across color channels. A better solution is to compute NCC per color channel independently and return the average NCC score. A more sophisticated approach is to compute and subtract the average intensity per color channel independently ($\bar{f}$ and $\bar{g}$), but concatenate all the color channels together as a single vector when computing its variance ($\sigma_f$ and $\sigma_g$). This allows NCC to capture spatial intensity variations in each color channel, while down weighting color channels with smaller intensity variations.

### 2.1.2 Sum of Squared Differences

The sum of squared differences (SSD) is defined as the $L^2$ squared distance between vectors $f$ and $g$

$$\rho_{SSD}(f,g) = ||f-g||^2.$$

It is usually mapped through an exponential to the $[0,1]$ range for normalization purposes. SSD with an exponential normalization is the optimal photo-consistency measure if $f$ and $g$ only differ by additive Gaussian noise with standard deviation $\sigma$

$$\rho_{SSD}(f,g) = e^{-\frac{||f-g||^2}{\sigma^2}} \in [0,1].$$

Note that SSD does not require a support domain $\Omega$ to be defined, but it easily generalizes to use one.

The use of the $L^2$ norm makes SSD sensitive to outliers, e.g. visibility outliers or bias and gain perturbations. A normalized variant of SSD exists that helps mitigate some of these issues:

$$\rho_{NSSD}(f,g) = ||\frac{f-\bar{f}}{\sigma_f} - \frac{g-\bar{g}}{\sigma_g}||^2,$$

where $\bar{f}$ is the mean of $f$ and $\sigma_f$ is the standard deviation of $f$. This

version is equivalent to NCC:

$$\begin{aligned}
\rho_{NSSD}(f,g) &= ||\frac{f-\bar{f}}{\sigma_f} - \frac{g-\bar{g}}{\sigma_g}||^2 && (2.4)\\
&= ||\frac{f-\bar{f}}{\sigma_f}||^2 + ||\frac{g-\bar{g}}{\sigma_g}||^2 - 2\frac{f-\bar{f}}{\sigma_f}\cdot\frac{g-\bar{g}}{\sigma_g} && (2.5)\\
&= 2(1 - \frac{(f-\bar{f})\cdot(g-\bar{g})}{\sigma_f\sigma_g}) && (2.6)\\
&= 2(1 - NCC(f,g)) && (2.7)
\end{aligned}$$

### 2.1.3 Sum of Absolute Differences

The sum of absolute differences (SAD) is very similar to SSD, but uses an $L^1$ norm instead of an $L^2$ norm, which makes it more robust to outliers

$$\rho_{SAD}(f,g) = ||f-g||_1.$$

Similarly to SSD it is sensitive to bias and gain, so it is rarely used in algorithms that match images with a wide variability in illumination. It is however a very good measure for applications that can guarantee similar capture conditions for the different images, e.g., real-time applications or mobile applications.

### 2.1.4 Census

Census [206] is one of the best performers for stereo correspondences [100]. Similarly to NCC, it is invariant to changes in gain and bias, and requires an explicit support domain to be computed. As a main difference from NCC, it does not use the intensity values themselves, but their relative order. Given a comparison operator

$$\xi(a,b) = 1\ if\ a < b,\ 0\ otherwise,$$

and a support domain $\Omega$ centered at $p$, census computes a bit string that describes whether a pixel in the support domain is brighter or darker than $p$

$$census(f) = \otimes_{q\in\Omega}\xi(f(p), f(q)),$$

where $\otimes$ is the concatenation operator. The census score is computed as the Hamming distance of the two bit strings, which can be computed as the $L^1$ norm of their difference:

$$\rho_{census}(f, g) = |census(f) - census(g)|_1,$$

with values in $[0, N]$, where $N$ is the size of the domain $\Omega$.

The main advantage of Census vs NCC is around depth boundaries, where it has been shown that Census is more robust than NCC and often outperforms it. Note that depth boundaries are an issue for any photo-consistency measure that explicitly requires a domain like Census or NCC. The reason is that, by definition, photo-consistency measures assume the appearance in the domain is intrinsic to the object and, to some extent, invariant to illumination and viewpoint changes. However this assumption breaks at depth discontinuities because the domain contains foreground and background objects.

Census is complementary to SAD. It provides bad scores for textureless regions, where SAD tends to give good scores, but with poor accuracy. As a result, hybrid measures have been proposed [141] that combine Census and SAD.

### 2.1.5 Rank

Rank was proposed at the same time as Census [206], and shares some of its characteristics. Like Census, it is invariant to changes in bias and gain and requires an explicit support domain to be computed. Unlike Census, it is also invariant to rotation. Instead of summarizing the values in the domain as a bit string, Rank summarizes the domain by computing the percentile of the reference pixel w.r.t. all the other values in the domain

$$rank(f) = \sum_{q \in \Omega} \xi(f(p), f(q)).$$

The rank score of two vectors $f, g$ is computed as the absolute difference of their ranks

$$\rho_{rank}(f, g) = |rank(f) - rank(g)|,$$

with values in $[0, N]$, where $N$ is the size of the domain $\Omega$.

### 2.1.6 Mutual information

Mutual information (MI) was first used as an image similarity measure by Viola and Wells [189]. Its main characteristic is that it is highly invariant, in particular to any transformation that does not change the amount of information of a signal, e.g., bijective mappings. As a result, it is used whenever signals are expected to undergo complex transforms. MI was particularly successful in the medical imagery community in the context of multi-modal imagery alignment [138]. Although it has been used in stereo algorithms [118, 98, 154], it is not a common measure since its invariance comes at the cost of degraded accuracy. Other measures less invariant such as NCC or Census are typically preferred.

In information theory, the mutual information of two random variables $X$ and $Y$ is a measure of how dependent the two variables are:

$$MI(X,Y) = \sum_{x \in X, y \in Y} P(x,y) \log \frac{P(x,y)}{P(x)P(y)},$$

where $P(x,y)$ is the joint probability of $X$ and $Y$, and $P(x)$ and $P(y)$ are the marginals. In the context of image similarity, the mutual information between two images (or image regions) measures how similar the two images are, i.e., how well one image predicts the other image. The photo-consistency measure is defined as

$$\rho_{MI}(f,g) = -MI(f,g).$$

The joint probability is estimated using a Parzen window method [151]

$$P(x,y) = \frac{1}{|\Omega|} \sum_{q \in \Omega} K(f(q) - x, g(q) - y),$$

where $K(\cdot,\cdot)$ is the particular 2d kernel being used, typically a Gaussian, and the range of $x, y$ is the range of the images being matched, for instance, [0, 255] for gray scale images. $P(x)$ and $P(y)$ are obtained by marginalizing the joint probability. Note that, compared to other similarity measures, MI usually requires a large domain $\Omega$ so that $P(x,y)$ properly models the joint distribution. This requirement makes it only useful in iterative methods where the geometry of the previous iteration can be used to warp the images and compute a joint histogram using larger image regions.

### 2.1.7 Interval comparison

A SAD variant that is more robust to sampling was proposed by[39] where, instead of comparing intensity values, the photo-consistency compares min/max intensity intervals. For each image $I_i$, the min/max interval image is computed by eroding/dilating the image with a given radius $r$:

$$I_i^{min}(p) = \min_{q \in \mathcal{N}_r(p)} I_i(q),$$
$$I_i^{max}(p) = \max_{q \in \mathcal{N}_r(p)} I_i(q),$$

where $\mathcal{N}_r(p)$ is a ball of radius $r$ centered at $p$. The radius $r$ is typically 0.5 pixels. The definition of $\rho_{SAD}$ is now changed to compare intervals rather than values:

$$\rho_{SAD}(f^{min}, f^{max}, g^{min}, g^{max}) = max(0, g^{min} - f^{max}, f^{min} - g^{max}).$$

This measure is robust to small sampling offsets, up to radius $r$, and is particularly useful near intensity boundaries, where small sampling errors can introduce large intensity errors, for example, from calibration errors. However, it also looses precision due to the clamping effect. Its impact is most significant with high resolution images, where calibration errors can be relatively large.

#### Photo-consistency normalization

The photo-consistency values computed by the measures described above are rarely used "as-is" in the later stages of MVS. Instead they are usually transformed through a non-linear operation with two goals: i) normalize different photo-consistency values to the same range, ii) transform the original photo-consistency into something closer to "*likelihood of geometry*". Normalization is important for parameter tuning as well as combining photo-consistency measures together, or with other cues. Typical transforms include the "*exponential*", "*linear truncated*" and "*smooth step*" functions, see Figure 2.5. The normalization function also serves as a *"black box"* that transforms the particular photo-consistency score into a geometry likelihood measure, i.e., it decides the range of scores where there is likely to be 3D geometry.

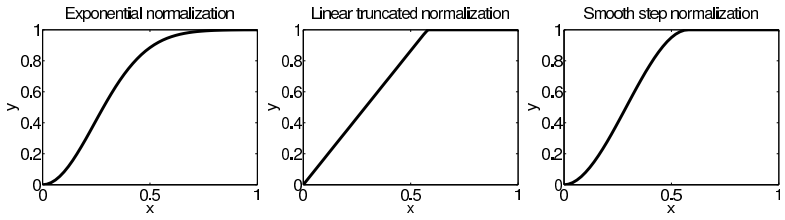



**Figure 2.5:** Different types of photo-consistency normalization functions. Left: exponential normalization $f_e(x) = 1 - e^{-\frac{x^2}{\sigma^2}}$. Middle: linear truncated normalization $f_l(x) = \min(1, \frac{x}{x_{max}})$. Right: smooth step normalization $f_s(x) = 3f_l(x)^2 - 2f_l(x)^3$.

Of the three normalization functions enumerated above, the exponential is the most theoretically sound and also the most common. It is the optimal transform for SSD if the noise is assumed to be Gaussian with no outlier component. Note however that pure Gaussian noise is not realistic, which is why SAD is often preferable to SSD.

When considering an inlier/outlier noise model for the photo-consistency score, the shape of the normalization function is expected to be sigmoid-like, with two plateaus separated by a relatively steep ramp, see Figure 2.5. The rationale behind it is that the steep slope acts as the optimal threshold separating the inlier/outlier model. As an example, for the case of SAD, typical inlier errors are less than 5 levels (out of 255), while typical outlier values are above 10 levels. Once SAD is above 10 levels, it does not matter if it is 20 or 30, both should give a very low likelihood of geometry. Similarly, NCC values below $\frac{1}{\sqrt{2}}$ are typically considered not accurate enough and discarded.

### 2.1.8 Photo-consistency aggregation

Photo-consistency is a noisy measure and often filtered before being used to compute 3D geometry. This additional filtering step is independent of the support domain $\Omega$ used in the photo-consistency definition (2.1). Since the domain can also be seen as a form of aggregation, the filtering step is most often used with measures that do not need a support domain e.g., SAD. It is particularly important for algorithms that are constrained by computation time and use local optimization

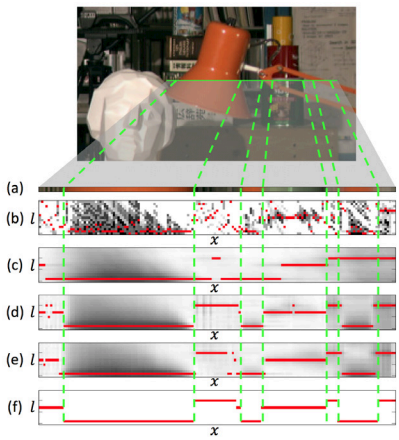



**Figure 2.6:** Cost volume filtering, courtesy of [105]. Anisotropic filters such as the bilateral filter [181] or guided filter [90] provide significantly better results than simple approaches around color boundaries. (a) A close-up view of the green line in the input image. (b) Slice of the cost volume (white/black/red: high/low/lowest costs) for line in (a). (c-e) Cost slice smoothed along x and y-axes (y is not shown here) with box filter, bilateral filter [181] and guided filter [90], respectively. (f) Ground truth labeling.

methods [163] instead of expensive global optimization methods such as graph cuts [122].

The most basic filtering step is to average the measure over a local domain, for example, a square window of fixed size. In the case of SAD, this is exactly equivalent to using the same window as the measure's support domain $\Omega$. More advanced filters exist that use spatially varying domains [205]. In particular, recent state-of-the-art stereo algorithms use weighted filters where, given a reference or guide image, the weights of the filter are a function of the color difference between the neighbor's pixel color and the reference's pixel color [157]. As a result, the filter only averages together photo-consistency samples with corresponding similar colors in the guide image. That is, it does not average across color boundaries and thus provides better photo-consistency values near object boundaries. Figure 2.6 shows the difference between filtering photo-consistency with a simple box filter (Figure 2.6.c) against more advanced anisotropic filters (Figure 2.6.d and 2.6.e).

Efficient implementations of anisotropic filters can be obtained with fast implementations of the bilateral filter [181], for example, using a permutohedral lattice [21] or with a guided filter [90]. The anisotropic filter can also be implemented directly in the photo-consistency measure [141], but more often they are implemented as a stand alone filtering step on the 3D volume containing the photo-consistency measure [105] (See Figure 2.6).

Although bilateral filters are the most common, other anisotropic filters can be used for stereo such as a weighted median filter [137]. The weighted median filter is more expensive to run, but it is more robust to alignment errors between the cost volume and the guide, for example, in the presence of boundary artifacts caused by severe occlusions.

### 2.1.9 Photo-consistency representation

The representation of photo-consistency is very related to the scene representation of the MVS application (see Table 3.1). From its definition in (2.1), photo-consistency is a volumetric quantity and can thus be computed and stored as a 3D volume. However, due to efficiency reasons, many applications do not store it as such. Some volumetric methods [101, 102, 94, 169, 129] are able to discretize the 3D volume in an efficient way by using multi-resolution 3D grids where more accuracy is available near the expected surface. Iterative applications [58, 154, 78, 55] compute the photo-consistency and its gradient on the current geometry. Finally, some methods [74, 85] simply represent the 3D photo-consistency as a list of (position, photo-consistency) pairs $(p_i, \mathcal{C}_i)$, where the 3D photo-consistency is defined as

$$\mathcal{C}(p) = \sum_{i:|p_i - p|<\epsilon} \mathcal{C}_i,$$

and $\epsilon$ is a predefined ball size. A particularly common variant of the point representation is as a set of sparse depthmaps [175, 93, 80], where each depthmap can be trivially converted to a point representation. The point representation can be particularly efficient and, in some cases, be the final output of the MVS algorithm, as in [74]. It also highlights the tight relationship between computing photo-consistency and

computing geometry. In the extreme case, one can see the geometry and its attached confidence as a sparse definition of photo-consistency.

### 2.1.10 Popular choices

Processing time, image variability, and surface coverage are three important variables in the design of the photo-consistency measure, which are dependent on the application from real-time MVS to cloud processing of controlled sequences from airplanes. Some measures such as SAD or SSD are extremely fast to compute and easily adapted to specialized hardware such as GPUs [202, 101, 76, 142, 145]. Others are particularly effective in the presence of bias and gain changes across the images, e.g., NCC and Census. Finally, the more images see the same piece of a surface, the more strict the normalization of the photo-consistency can be (See Section 2.1.7). Two photo-consistency measures that are particularly popular are NCC and SAD. NCC is mainly used to match images with varying lighting conditions and usually good coverage. SAD on the other hand is used when the images are not expected to have bias or gain changes, and coverage is low, e.g., two view stereo. Note that, even when the illumination is constant and the camera response is known, images can still show local gain changes due to the presence of non-Lambertian materials. As a result, some techniques [141] combine SAD with another measure such as NCC or Census.

## 2.2 Visibility estimation in state-of-the-art algorithms

All the photo-consistency measures described above assume a known set of visible images to compute photo-consistency. However, in the general case one does not know beforehand which images see what, since the 3D geometry itself is unknown (See Figure 2.7). This leads to a dependency loop where, in order to compute 3D geometry (by maximizing a photo-consistency measure), one needs the correct 3D-geometry to select visible images for the photo-consistency computation. In the following we will describe common approaches to break this loop.

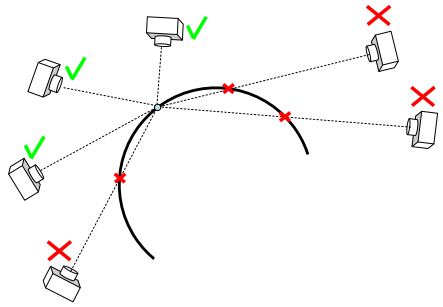


**Figure 2.7:** Occlusion problem. In order to compute geometry using photo-consistency, the camera visibility is required. At the same time, in order to compute the camera visibility, the geometry is required.

### 2.2.1 Space-carving for visibility estimation

The work of space carving [166, 128] was seminal in the proposal of a theory of visibility consistency. Given a 3D volume partitioned into a 3D grid of voxels, the volume is iteratively carved out by removing voxels that are not photo-consistent. The main contribution of the work was the proposal of a geometric constraint on the camera centers such that there exists an ordinal visibility constraint on all the 3D voxels in the scene. That is, starting from an initial bounding volume, one can visit all the voxels in an order that guarantees that any potential occluder voxel is always visited before its potential occluded voxel.

This property effectively solves the visibility issue as it provides a voxel visiting order that guarantees that we never use an image where the voxel being tested is occluded. The ordinal visibility constraint is satisfied whenever no 3D point of the scene is contained in the convex hull of the camera centers. This is a strong constraint, but there are many useful capture configurations that satisfy this, for example, when the camera centers lie on a 3D plane while the camera is looking downwards, as commonly used to capture 3D geometry at large scale by flying high-altitude airplanes or drones.

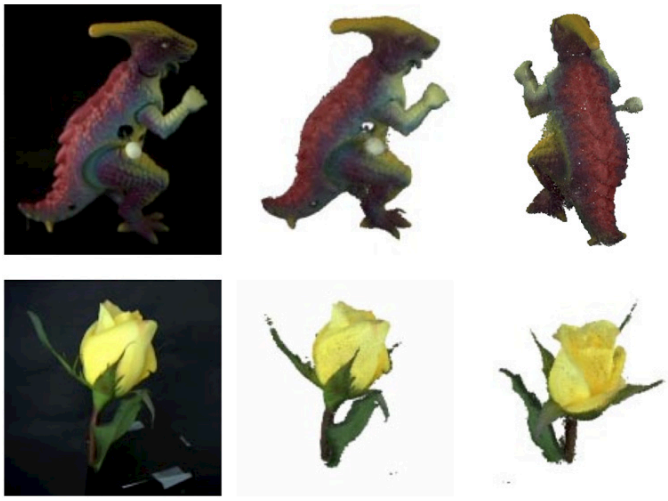


**Figure 2.8:** Voxel coloring [166] results of Dinosaur toy and Rose. The objects were rotated 360° below a camera. Left shows one of the 21 input images. Right shows two different views of the reconstruction.

The photo-consistency measure used in space carving is a variant of SSD for multiple views where the score is the variance of all the intensities collected in the images that see a particular voxel. The support domain $\Omega$ is defined by the projection of the 3D voxel into a given image. Figure 2.8 shows results obtained with this technique on two different datasets.

In recent MVS methods, visibility is handled in a very different way from the original space-carving work. State-of-the-art MVS methods are able to handle millions of images, and visibility estimation is a critical step to achieve computational efficiency so that photo-consistency is evaluated with a relatively small number of images for each location. Visibility estimation for such large image collections typically happen in two phases.

In the first phase, visibility is estimated *coarsely* by clustering the initial set of images and reducing the large-scale MVS problem into a sequence of small sub-problems [93, 79, 74, 69]. For instance, given a single image, this step reduces the number of possible candidate images

for photo-consistency evaluation from an order of millions to an order of a few dozens. In the second phase, more fine-scale visibility estimation is conducted per 3D point basis, often, as one reconstructs a scene with an MVS algorithm.

### 2.2.2 Coarse visibility estimation via pose clustering

Coarse visibility estimation is at the core of how MVS algorithms scale to millions of images, and can be massively parallelized. Given an initial set of images, early MVS algorithms [93, 79, 74] created a small MVS problem for each input image i.e., each input image would be the reference of a view cluster exactly once. This small MVS sub problem can be formulated as a narrow-baseline stereo problem, because a view cluster is relatively small, between 5 and 10 images, with the angle between the reference image and the neighboring images between 5 and 15 degrees.

The work of [79] was the first to use view clustering for large collections of unstructured imagery. It proposed to use the full SfM model to drive view clustering by using the number of shared matches between any two views as an indication of the overlap between the two views. However, stereo performance is sensitive to other metrics such as baseline or resolution, so they proposed a global cost function to rate a candidate cluster

$$score(R, V) = \sum_{f \in \mathbf{F}_R \cap \mathbf{F}_V} w_b(f) \cdot w_s(f). \tag{2.8}$$

$R$ denotes the reference image, $\mathbf{F}_V$ is the set of feature points observed in view $V$, $w_b$ down-weights neighbor views with a small baseline w.r.t. $R$, and $w_s$ encourages images with similar or higher resolution than $R$. The cluster is initialized to the reference image $R$ and the best next view maximizing (2.8) is iteratively added to the cluster until the maximum cluster size is reached.

As the number of images increases to millions, creating a narrow-baseline stereo problem per input image becomes onerous and redundant. As a result, a preliminary stage of view selection is needed to remove redundant images and only keep the images that really make

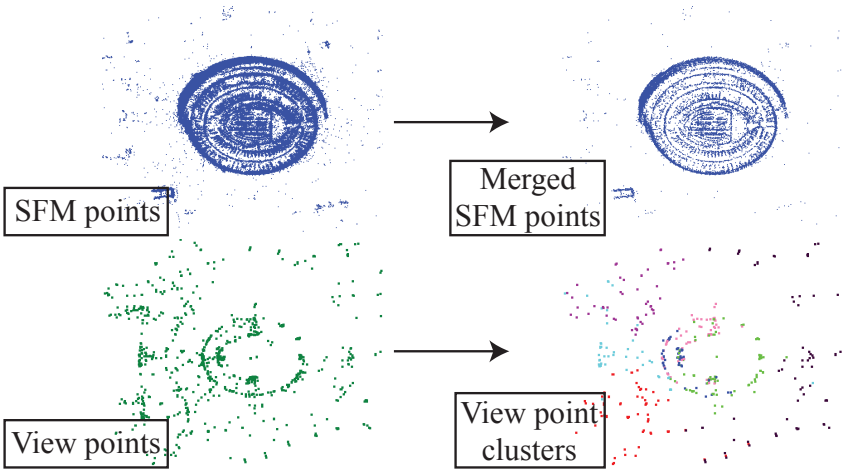



**Figure 2.9:** Large scale view clustering of internet images of the Colosseum [69]. Top: The first step of the algorithm is to merge SFM points to enrich visibility information (SFM filter). Bottom: Sample results of the view clustering algorithm. View points belonging to extracted clusters are illustrated in different colors.

a difference in the MVS framework and provide a good quality reconstruction. The work of [69] proposed an algorithm to minimize the number of view clusters while guaranteeing a good reconstruction (See Figure 2.9). In particular, it uses the SfM 3D points as a proxy for the scene geometry and formulates the view selection problem as an constrained optimization problem. The algorithm minimizes the number of clusters while guaranteeing that:

- No cluster is bigger than a certain maximum size, so that MVS can run on each cluster.
- A large enough percentage of SfM points can be correctly reconstructed by at least one cluster, i.e. the views in the cluster can triangulate the point with enough accuracy.

### 2.2.3 Fine-scale visibility estimation

Fine-scale visibility estimation requires a detailed 3D model to handle occlusions properly and it is a core problem for any MVS reconstruction

algorithm (See Figure 2.7). One popular approach is to use the current reconstructed geometry to compute occlusions (e.g., a z-buffer testing), select which views see which parts of the geometry, and iterate visibility estimation and reconstruction. This iterative approach assumes there is an initial geometry that is being refined [58, 175, 154, 78, 56, 55]. Therefore, the main disadvantage of these methods is that they depend on a good initialization. If the initialization is not accurate, they can easily get stuck in a local minimum. For this reason, iterative methods are often used as a refinement step, once an initial coarser solution is available, e.g. from a volumetric fusion approach [193].

Another popular solution is to rely on robust photo-consistency statistics without explicitly estimating occlusion. An intuition is that "bad" photographs would yield poor photo-consistency measures, whose effects can be suppressed by the use of robust statistics if there are enough "good" images present. The exact choice and usage of robust statistics is algorithm dependent, see Chapter 3 for more details.

# 3

# Algorithms: From Photo-Consistency to 3D Reconstruction

In this chapter we will provide details of recent popular multi-view stereo (MVS) algorithms that reconstruct 3D geometry by using photo-consistency functions in Chapter 2. There are many factors that differentiate MVS algorithms, such as the photo-consistency function, scene representation, visibility computation, and initialization requirements. Therefore, it is not an easy task to come up with a single classification. In this article we use the output scene representation as an axis of taxonomy, because it often determines the range of possible applications. Interested readers are referred to [165] for other useful ways of categorizing MVS algorithms.

Figure 3.1 illustrates four popular output scene representations that are used by modern MVS algorithms: a depthmap(s), a point cloud, a volume scalar-field, and a mesh. For each representation, a reconstruction example by a state-of-the-art algorithm is shown. Note that a point cloud reconstruction is visualized by a point-based rendering technique [160, 83], which may exhibit an appearance of a complete texture-mapped mesh model, but they are just independent 3D points with colors. Volume scalar-fields are often used in Computer Vision and Computer Graphics to represent a 3D surface, where a common

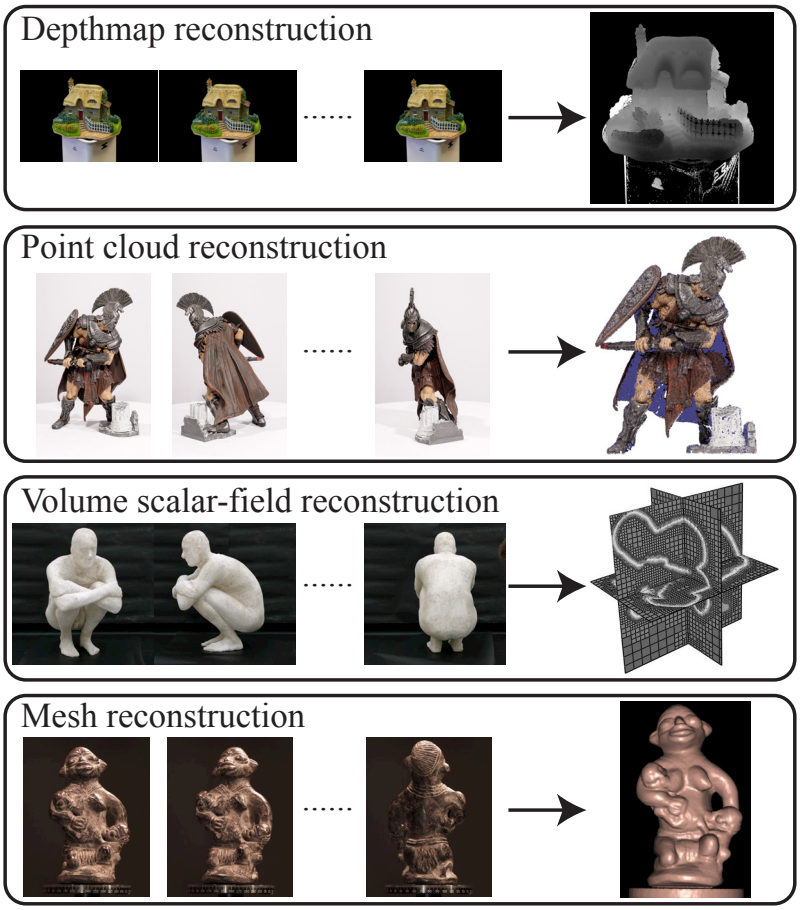



**Figure 3.1:** MVS algorithms can be classified based on the output scene representation. The four popular representations are a depthmap(s), a point cloud, a volume scalar-field, and a mesh. Note that a point cloud is very dense and may look like a textured mesh model, but is simply a collection of 3D points. Reconstruction examples are from state-of-the-art MVS algorithms presented in [48], [74], [94], and [93] respectively, from top to bottom.

convention is to treat the scalar-field as a (signed) distance function field from a surface. The surface can be extracted as a zero iso-surface of the function field.

Figure 3.2 illustrates the relationships of MVS reconstruction steps as well as the intermediate/final geometry data types. Many MVS algorithms focus on a single reconstruction step, while others combine multiple steps to form a pipeline. This diagram explains most MVS algorithms or systems, while one exception is an approach that reconstructs a mesh directly from the photo-consistency volume via the volumetric fusion [190, 102]. In this approach, the photoconsistency volume serves as the replacement of the depthmaps or the point clouds.

There are certain reconstruction techniques that have been explored in the past but are not listed here. For example, *level-set* was once a popular technique for MVS due to its ability to handle topological changes [58]. A typical reconstruction procedure was to initialize then refine a model, where the topology of the initial reconstruction can be incorrect. However, level-set is not popular for MVS reconstruction any more, because better initialization or reconstructions techniques have been developed, as seen in this chapter, and high quality models with correct topology can be directly computed from photo-consistency functions without the refinement steps. Similarly, in early days, many algorithms initialized meshes based on the visual hull [35]. However, they are not popular any more either, because silhouette extraction is often a tedious manual procedure, and visual hull is not an effective approximation or simply impossible for many scenes with a lot of concave structure. The development of better initial reconstruction technique is also the reason as above [1].

[1] There exist fully automated silhouette extraction algorithms for scenes where there is a clear separation between an object and a background [47, 49, 50]. However, they are outside the scope of this article and details are referred to those papers.

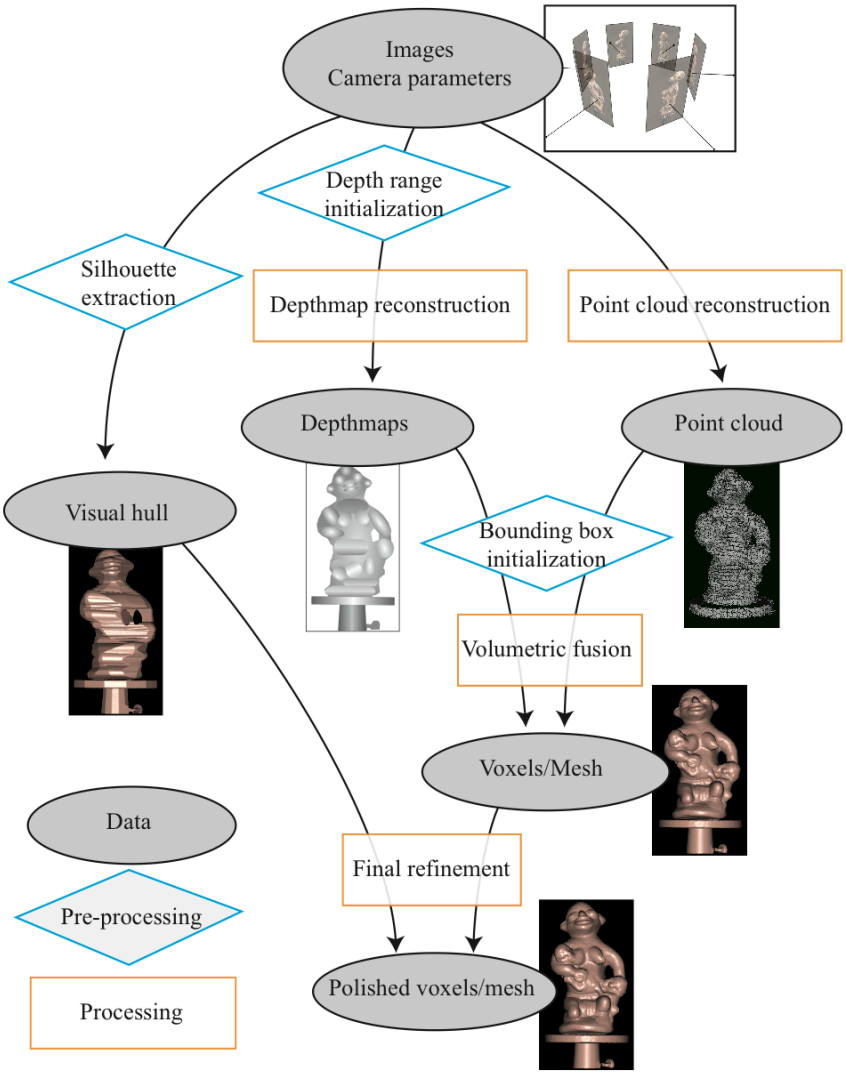



**Figure 3.2:** MVS processing often consists of multiple stages (algorithms), and the figure illustrates typical processing flow. 3D data representation determines how different algorithms can be put together, and gray ovals correspond to data representations, which are connected by either an MVS algorithm or a pre-processing step. Note that "Polished voxels/mesh" is not necessarily the goal of every MVS system. Depending on the applications, the final step of the MVS system would be different.

**Table 3.1:** A scene representation determines the range of available applications. The table shows four popular scene representations in rows and three popular applications or post-processing usages in columns.

| Application / Scene representation | View-dependent texture-mapping | Free-viewpoint rendering | Ease of manipulation (e.g., merging/splitting) |
|---|---|---|---|
| Depthmaps | Good | Fair (point-based rendering) | Fair |
| Point-cloud | Poor | Fair (point-based rendering) | Good |
| Voxels | No | No | Good |
| Mesh | Poor | Good | Bad |

### 3D Representations and Applications

Table 3.1 summarizes the availability of the four scene representations for three popular applications. The major use of 3D reconstruction is for Graphics applications (i.e., visualization), and the table lists two different types of visualization applications.

View-dependent texture-mapping techniques change the images used for texture mapping depending on the view of the rendering camera. This technique yields immersive visualization experiences, as the rendering is primarily based on real photographs and can convey even complex real photometric effects such as specular highlights or transparency/translucency, which are difficult to model [54, 68, 170]. Google Streetview [81] is a good example of view-dependent texture mapping. However, the rendering camera has to be near an input image to avoid rendering artifacts, and the motion of the rendering camera is severely limited by the coverage of the input photographs. A depthmap representation is particularly effective for the view-dependent texture mapping, because its geometry can be optimized for rendering per view [127]. The sky often poses challenges for outdoor scene visualization, because its geometry is not well defined and cannot be reconstructed. The depthmap representation allows one to generate a

geometric proxy for the sky per view for better rendering. This is not easy for a mesh, a point cloud, or a voxel, whose definitions are usually independent of the views.

Free-viewpoint rendering, on the other hand, allows one to move freely in the space, which serves better for navigation and exploration purposes. Google Earth is a good example of free-viewpoint rendering. However, the rendering is usually view-independent and lacks realism. For this task, a mesh or a point cloud is naturally more suitable. Texture-mapped MVS meshes have been successfully used in real products for outdoor city visualization [30, 108, 144]. Point based rendering techniques have been extensively studied in the Computer Graphics literature [83]. High quality visualization results have been demonstrated for high fidelity MVS point clouds or depthmaps, which can be treated as a point cloud [69, 117, 104]. However, relatively small amount of work exist for the point-based rendering of MVS point clouds. MVS point clouds often suffer from severe noise and large reconstruction holes, and the rendering quality may degrade significantly.

The last application is the geometry manipulation. This is getting more and more important, as MVS techniques evolve and make it possible to reconstruct complex and large scenes. The handling of multiple MVS reconstructions are necessary to complete a model of a scene. A mesh representation faces challenges for this task, as it is often difficult to control the topology of the mesh through the merging and splitting operations, then enforce the manifoldness.

In Figure 3.2, polished voxels (volumetric scalar-field) and mesh are at the bottom of the diagram. However, this may not be the goal of every MVS system. For example, if view-dependent texture mapping is the application, one should simply pick a depthmap reconstruction algorithm. If free-viewpoint rendering is the application, one can conduct point-cloud reconstruction from images, then can use a point-based rendering technique for an application without running any of the other steps in the diagram. Of course, a high-quality polygonal mesh model is often a preferred scene representation, and all the processing converges to the mesh reconstruction in the diagram.

### Evaluations

MVS researchers have conducted quantitative evaluations to verify the accuracy of MVS algorithms [165, 176]. Seitz, Curless, Diebel, Scharstein, and Szeliski set up a foundation for MVS quantitative evaluation in 2006 [165], which evaluated MVS algorithms on two object datasets with low resolution ($640 \times 480$) images, that were carefully acquired in a lab environment with fully controlled lighting. This evaluation is known as the *Middlebury MVS evaluation*. Although the use of low resolution images may not reflect the existence of high resolution digital cameras in modern consumer markets, it has the advantage of minimizing the influence of calibration errors: Higher image resolution requires more accurate and repeatable mechanical device (e.g., a robot arm). A few years later, Strecha, Hansen, Van Gool, Fua, and Thoennessen published complementary MVS benchmark datasets and evaluation system that focus on outdoor scenes and high resolution input images, which reflects the trend and needs of MVS research [176]. Many algorithms recorded impressive numbers in reconstruction accuracy (e.g., 0.5mm accuracy within a 20cm volume from $640 \times 480$ images) and also produced qualitatively compelling 3D models including all the state-of-the-art algorithms presented in this chapter.

One missing evaluation is the visual quality of the reconstructed models. The Middlebury MVS evaluation has revealed the fact that the pure geometric quantitative metrics do not always reflect the visual quality of the models. In other words, models with clear visual artifacts sometimes achieve better geometric accuracy. Recent MVS algorithms produce visually high quality 3D models beyond being geometrically accurate [164, 167]. Future MVS evaluations should potentially take into account both the geometric accuracy and the visual quality.

We now provide details of MVS reconstruction algorithms for each of the four output scene representations.

## 3.1 Depthmap Reconstruction

Depthmap scene representation is one of the most popular choices due to the flexibility and scalability. Suppose one is given tens of thousands

of images and camera parameters as input. One can simply reconstruct a depthmap for each input image, maybe after finding a small number of neighboring images to be used together for photo-consistency evaluation. By treating a depthmap as a 2D array of 3D points, multiple depthmaps can be considered as a merged 3D point cloud model already. This processing is simple and easily scalable to a large number of images.

Depthmap reconstruction in MVS is usually performed under a narrow baseline assumption, and its formulation is the same as the traditional *two view stereo* [162]. The method takes a set of images with camera parameters, discretizes the valid depth range into a finite set of depth values, then reconstructs a 3D geometry for a reference image. Uniform depth sampling may suffice for simple and compact objects. However, for complex and large scenes, a proper sampling scheme is crucial to achieve high speed and quality. Researchers proposed perspectively corrected or logarithmic parameterization to sample depth values, where details are referred to the papers [203, 104, 75].

However, MVS depthmap reconstruction algorithms tend to be simpler than their two view stereo counterparts, because there are often a lot more images and thus more redundancy. In other words, in the context of MVS, the completeness of a single depthmap is not important as long as the merged model is accurate and complete.

In the remainder of this section, we describe the details of several representative MVS depthmap algorithms, then talk about a few more advanced techniques.

### 3.1.1 Winner-Takes-All Depthmaps

Suppose one is given a reference image, whose depthmap is to be computed, a set of neighboring images, and a range of depth values that should contain a scene to be reconstructed. A simple depthmap reconstruction algorithm is to evaluate photo-consistency values throughout the depth range, and pick the depth value with the highest photo-consistency score for each pixel independently. This process is called "Winner-takes-all" strategy. Figure 3.3 illustrates the process, where the Normalized Cross Correlation (NCC) is used as the photo-

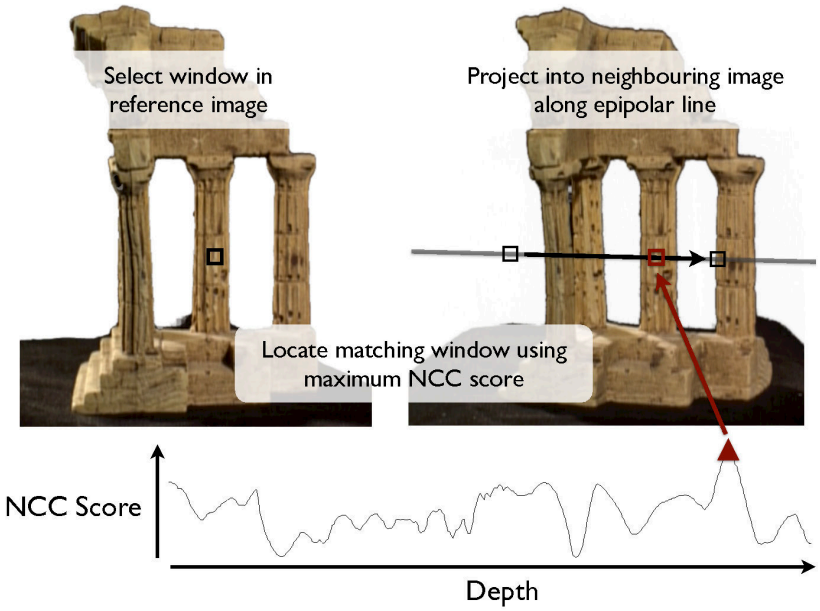



**Figure 3.3:** Winner-takes-all strategy for depthmap reconstruction. The figure illustrates a process to estimate a depth value for a pixel highlighted by a black rectangle in the left image. The global maximum of the photo-consistency function such as the NCC score is chosen to be the reconstructed depth for the pixel.

consistency measure, and is expected to have a peak at the correct depth. The full algorithm description is given in Algorithm 1. In addition to the depth value with the highest photo-consistency, the algorithm often evaluates a confidence measure so that low-confidence depth values can be ignored or down-weighted in the later model merging step [106]. This simple algorithm works surprisingly well and was first demonstrated by Hernández and Schmitt [93]. Various improvements have been made, and we now turn our attention to more sophisticated methods in the following sections.

**Algorithm 1** Simple depthmap estimation algorithm

```
1: procedure SimpleDepthmap(I_ref,N(I_ref),𝔻)
2: // I_ref: reference image
3: // N(I_ref): neighboring images of I_ref
4: // 𝔻: depth range
5:     for all p ∈ I_ref do // p: pixel
6:         C_best ← −∞
7:         for all d ∈ 𝔻 do
8:             if C(p,d) > C_best then
9:                 C_best ← C(p,d)
10:                d_best ← d
11:                f_best ← F(p,d) // confidence value
12:            end if
13:        end for
14:        return (p, d_best, f_best)
15:    end for
16: end procedure
```

### 3.1.2 Robust Photo-Consistency Depthmaps

While Algorithm 1 works fairly well, there is no guarantee in general that the appearance in a matching window is unique across the surface of an object. A larger window size more likely leads to unique matches. However, the associated peak will be broader and less well localized, reducing the accuracy of the depth estimate. Occlusions and non-Lambertian photometric effects such as specular highlights also add noise to the photo-consistency function. Therefore, simply taking the average of such scores as in (2.3) may not work well (See Figure 3.4). Vogiatzis, Hernández, Torr, and Cipolla [190] proposed a robust photo-consistency function to overcome these challenges. More concretely, given photo-consistency curves at a pixel, which are calculated between a reference image and each of the nearby images, the algorithm first identifies local maxima from all the photo-consistency curves. Let $d_k$ be the depth and $\mathcal{C}_k$ be the corresponding photo-consistency score of such $k_{th}$ local maximum. Then, the robust photo-consistency function

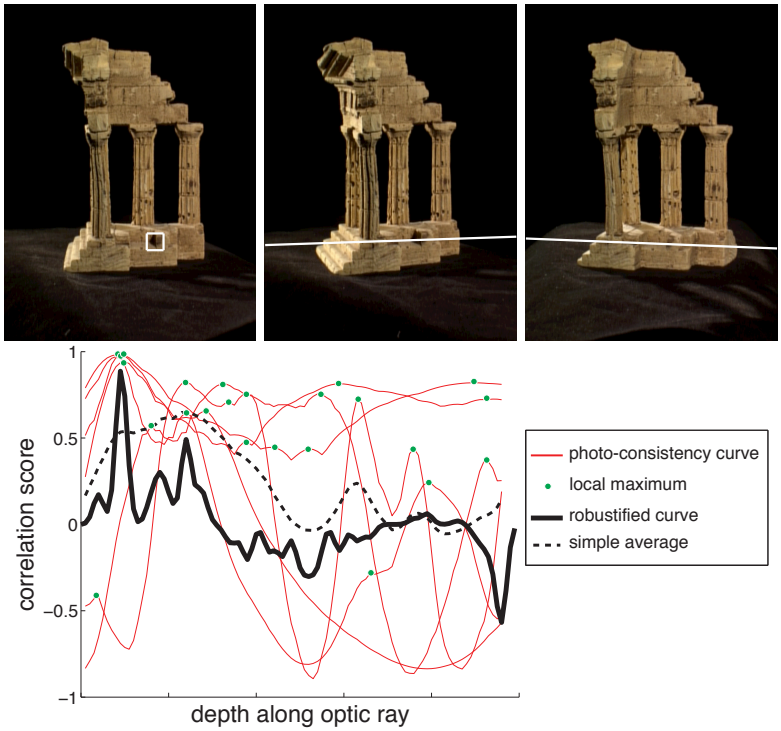



**Figure 3.4:** Robust photo-consistency function by Vogiatzis, Hernández, Torr, and Cipolla in the depthmap estimation framework [190], which is a sum of kernel functions (e.g., Gaussian) at major local maxima. The six photo-consistency curves between a reference image and the six nearby images are plotted in red. Curves from occluded viewpoints (such as the top-right image) do not have an optimum in that location and hence a simple averaging of the curves (dashed line) does not work.

$\mathcal{C}^R(d)$ is constructed using a Parzen window [151] approach as:

$$\mathcal{C}^R(d) = \sum_k \mathcal{C}_k W(d - d_k), \tag{3.1}$$

where $W$ is a kernel function such as a Gaussian function [190]. The effects of this step are illustrated in Figure 3.4: the simple average selects the wrong depth as the global maximum, while the robust photo-consistency successfully suppresses outliers. Figure 3.5 illustrates how the noise in photo-consistency can be reduced by such improvements.

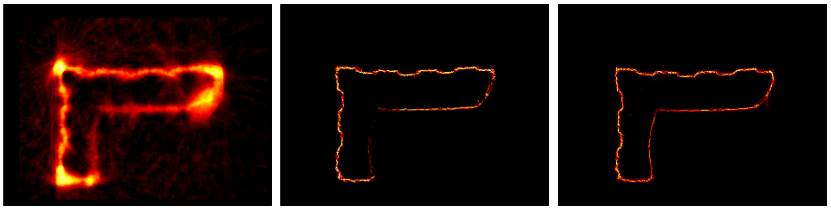


**Figure 3.5:** Noise reduction in photo-consistency. Left: a slice of the photo-consistency used in [191] contains falsely photo-consistent regions (*e.g.* near the corners). Middle: robust photo-consistency proposed in [93] significantly suppresses noise and the correct surface can be accurately localized. One side of the vertical wall is missing due to heavy occlusions. Right: MRF formulation and optimization enforces spatial consistency for further improvements [48]. The vertical wall is correctly represented.

Another much simpler yet effective approach is to ignore photo-consistency scores that are below a certain threshold. Goesele, Curless, and Seitz simply compute the average of pairwise photo-consistency scores after ignoring values below a certain threshold [80]. While such thresholding is a very sensitive operation, and results severely depend on the parameter selection in general, NCC for photo-consistency is known to be surprisingly robust and stable across different input data. Therefore, a constant value is often used to threshold the NCC score. Similar handling of photo-consistency function is employed in the point-cloud reconstruction framework, where details are referred to Section 3.2.1.

Reconstruction results of a winner-takes-all depthmap algorithm with a robust photo-consistency function by Goesele, Curless, and Seitz [80] are given in Figure 3.6. The top row shows a sample reference image at the left, and two reconstructed depthmaps with different thresholds on the depth confidence at the right. A depth estimate of a pixel is discarded when the confidence estimate is below the threshold. The threshold is tighter (higher) for the left depthmap, and hence, there is less noise but more holes are observed. Note that a depthmap is usually visualized as an image by converting an estimated depth value into a valid image intensity. However, in this figure, a depthmap

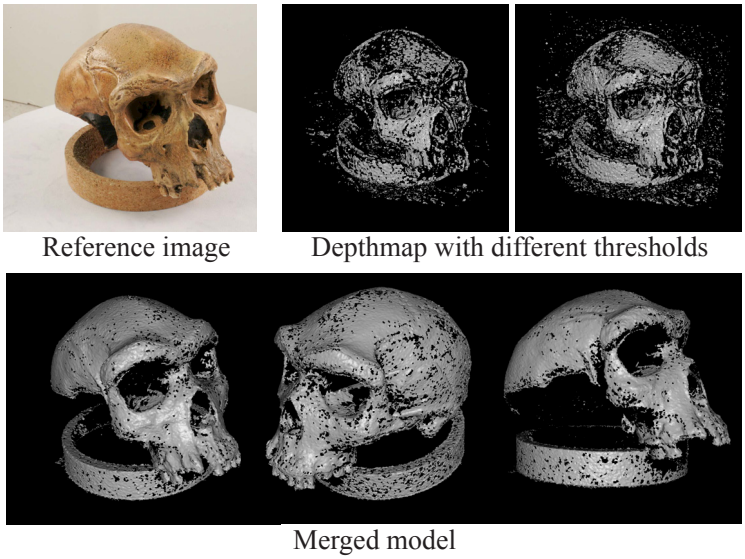



**Figure 3.6:** Reconstruction results by Goesele, Curless and Seitz [80]. The first row shows a reference image and two depthmaps with different thresholds for the robust photo-consistency evaluation. The second row shows that even though a single depthmap may contain many holes, multiple depthmaps can be merged into a complete 3D model. (Figure courtesy of Goesele et al.)

is visualized as a shaded polygonal model, which was obtained via a simple volumetric fusion technique [52] (See Section 3.3). The dataset consists of twenty four images, and they reconstructed twenty four depthmaps, then merged them into a single polygonal model by the same fusion technique, whose results are shown at the bottom row. A single depthmap is very noisy and contains many holes, but the merged model becomes much cleaner and exhibits less reconstruction holes. The effects of the number of input images on the reconstruction quality are shown in Figure 3.7. With more than 300 images, the *temple* model becomes complete, while the *dino* model still has some holes due to homogeneous textures, which make photo-consistency evaluation more challenging.

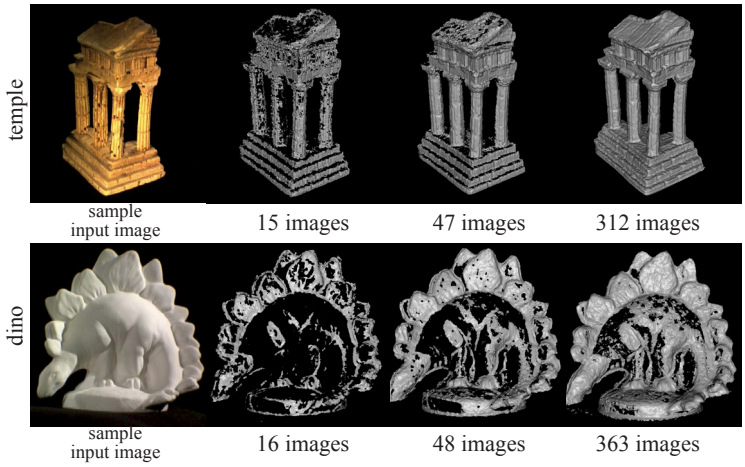



**Figure 3.7:** The effects of the number of input images after depthmap merging for the two datasets. The algorithm by Goesele, Curless, and Seitz is used [80]. (Figure courtesy of Goesele et al.)

### 3.1.3 MRF Depthmaps

Despite the use of the robust photo-consistency function in the previous section, the peak of a photo-consistency curve may not correspond to the true depth in challenging cases. In the presence of severe occlusions, there may not exist a corresponding match in most other images. A standard solution for these problems is to enforce spatial consistency, under the assumption that neighboring pixels have similar depth values, where Markov Random Field (MRF) is a very popular and successful formulation for the task. The MRF depthmap formulation [120] can be seen as a combinatorial optimization problem, where an input depth range is discretized into a finite set of depth values. The problem is then to assign a depth label $k_p$ from the label set to each pixel $p$, while minimizing the following cost function:

$$E(\{k_p\}) = \sum_p \Phi(k_p) + \sum_{(p,q)\in\mathcal{N}} \Psi(k_p, k_q). \tag{3.2}$$

The first summation is over all the pixels in the image, while the second summation is over all the pairs of neighboring pixels denoted

as $\mathcal{N}$. Neighboring pixels are often defined via 4-neighborhood or 8-neighborhood systems. In the former, a pixel is connected to the horizontal and vertical adjacent pixels. In the later, adjacent diagonal pixels are connected in addition. The 4-neighborhood system has less interaction terms and is cheaper, but may suffer more from discretization artifacts. The following sections discuss the formulation of the unary potentials $\Phi(\cdot)$ and pairwise interaction potentials $\Psi(\cdot,\cdot)$.

### Unary Potentials

The unary labeling cost reflects the photo-consistency information, where the cost should be set inversely proportional to the photo-consistency score. Exact definition of the unary cost varies. However, suppose NCC is the choice of photo-consistency function, whose score is guaranteed to be in the range $[-1, 1]$. Then, the unary cost can be defined as the following truncated linear loss function:

$$\Phi(k_p = d) = \min(\tau_u, 1 - \mathcal{C}(p, d)). \tag{3.3}$$

$\tau_u$ is a cut-off threshold. Of course, another arbitrary robust function, such as Huber loss or Cauchy loss, can be used instead.

### Pairwise Interaction Potentials

The pairwise cost enforces the spatial regularization and is set to be proportional to the amount of depth discrepancy at neighboring pixels, so that neighboring pixels have similar depth values. The definition of the pairwise cost also varies, but a simple implementation can be given below again as a truncated linear loss function to avoid penalizing the depth discontinuity too much:

$$\Psi(k_p = d_1, k_q = d_2) = \min(\tau_p, |d_1 - d_2|). \tag{3.4}$$

### Optimization

The problem of the form (3.2) is in general an NP-hard problem, but there exist many efficient approximations, in particular, when the pair-

wise cost at every pair of neighboring pixels satisfies the following *submodularity* condition [122]:

$$\Psi(\alpha,\alpha)+\Psi(\beta,\gamma)\leq\Psi(\beta,\alpha)+\Psi(\alpha,\gamma). \tag{3.5}$$

For submodular functions, one of the most popular techniques is called alpha-expansion [122, 45, 44], which repeatedly solves max-flow min-cut algorithm to improve label assignments.

Fortunately, the submodularity condition holds true for many standard pairwise terms. More concretely, as a distance metric, $\Psi(\alpha,\alpha)$ should be 0, because both labels are the same. Then, the remaining conditions becomes a triangular inequality:

$$\Psi(\beta,\gamma)\leq\Psi(\beta,\alpha)+\Psi(\alpha,\gamma). \tag{3.6}$$

The smoothness prior is usually defined as a distance metric, and satisfies this triangular inequality [2]. Examples of such metrics are linear, truncated linear, or Cauchy loss functions. However, quadratic or Huber loss functions are not submodular, because the quadratic function does not obey the triangular inequality. Note that unlike the pairwise cost, there is no restriction in the unary potential, and can be arbitrarily set. MRF is a popular formulation for many other Computer Vision problems as well, and more details on MRF can be found in the following articles [114, 179].

### 3.1.4 Multiple Hypothesis MRF Depthmaps

Campbell, Vogiatzis, Hernández, and Cipolla extended the standard MRF formulation in the previous section to further improve results [48]. Instead of using blindly discretized depth values as possible label set for an entire image, their algorithm extracts local maxima from photo-consistency curves for each pixel, then the MRF formulation is used to assign the depth of one of such local maxima to each pixel. Therefore,

[2] Submodularity optimization is a hot research topic in machine learning community, where the submodularity describes a mathematical property of a general set function. However, in Computer Vision, submodularity is often used to describe a property of an objective function of a multi-labeling combinatorial optimization problem. They are mathematically equivalent but treated in a very different way in the two communities.

different pixels have different label sets. They also allow the "unknown" label to indicate that no correct depth can be estimated in certain cases. In this situation, they acknowledge that the depth at this pixel is unknown and should therefore offer no contribution for the surface location. This process means that the returned depth map should only contain accurate depths, estimated with a high degree of certainty.

The process consists of two phases: 1) extraction of depth labels; and 2) MRF optimization to assign extracted depth labels. We now describe the details of the algorithm.

### Depth Label Extraction

The first phase is to obtain a hypothesis set of possible depths for each pixel $p$ in a reference image $I_{ref}$. After computing a photo-consistency curve within a depth range between $I_{ref}$ and each of the neighboring images, they store the top $K$ peaks $\{d_i(p)|i \in [1, K]\}$ with the greatest scores from all the curves. NCC is the photo-consistency function. As described before, another key feature of the algorithm is the inclusion of an unknown state $\mathcal{U}$, which is to be selected when there is insufficient evidence. Therefore, for each pixel, they form an augmented depth label set $\{\{d_i(p)\}, \mathcal{U}\}$.

### MRF Optimization

Depth label assignment is formulated as a MRF, where each pixel has a set of up to $(K + 1)$ labels. The first $K$ labels, fewer if an insufficient number of peaks were found during the label extraction stage, correspond to the peaks in the photo-consistency function and have associated depths $d_i(p)$ and scores $\mathcal{C}(p, d_i(p))$. The final state is the unknown state $\mathcal{U}$ as described before.

The unary cost is straightforward. We wish to penalize local maxima with lower matching scores, since they are more likely to correspond to an incorrect match. They take an inverse exponential function to map this score to a positive cost [190], while a constant penalty $\Phi_{\mathcal{U}}$ is imposed for the unknown state to avoid assigning depth values that have poor photo-consistency and do not have pairwise support from

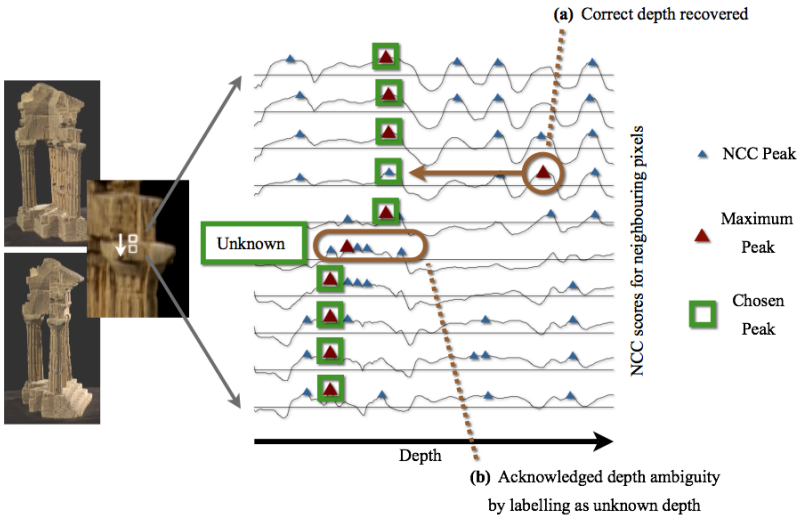



**Figure 3.8:** Illustration of the MRF optimization applied to neighboring pixels. Existing methods return the maximum peak which results in outliers in the depth estimate. The MRF optimization corrects an outlier to the true surface peak (a) and introduces an unknown label at the occlusion boundary (b). (Figure courtesy of Campbell et al.)

neighboring pixels:

$$\Phi(k_p = x) = \begin{cases} \exp\left[-\beta \cdot \mathcal{C}(p,x)\right] & \text{If} \quad x \in \{d_i(p)\} \\ \Phi_{\mathcal{U}} & \text{If} \quad x = \mathcal{U}. \end{cases} \tag{3.7}$$

The pairwise term enforces spatial regularization. There are two types of labels (depths and the unknown state), and the pairwise cost is defined in the following $4 (= 2 \times 2)$ cases:

$$\Psi(k_p = x, k_q = y) = \begin{cases} 2\frac{|x-y|}{(x+y)} & \text{If} \quad x \in \{d_i(p)\}, \quad y \in \{d_i(p)\} \\ \Psi_{\mathcal{U}} & \text{If} \quad x = \mathcal{U}, \quad y \in \{d_i(p)\} \\ \Psi_{\mathcal{U}} & \text{If} \quad x \in \{d_i(p)\}, \quad y = \mathcal{U} \\ 0 & \text{If} \quad x = \mathcal{U}, \quad y = \mathcal{U} \end{cases} \tag{3.8}$$

In the first case above where both labels are depth values, the cost simply measures the amount of discrepancy as in (3.6). Note that the discrepancy is normalized by the average of the depth values to make it

less scale dependent. In the second and the third cases where one of the labels is the unknown state, the constant penalty is imposed to prevent frequent switches between depth labels and the unknown state. In the last case, both labels are the same unknown state and the penalty is set to 0 to favor the spatial consistency.

The pairwise cost is unfortunately not submodular in this formulation, because depth labels are extracted for each pixel independently, and the meanings of the $i_{th}$ label are arbitrarily different for different pixels. For example, $\Psi(d_i(p), d_i(q))$ is 0 in the standard MRF formulation (3.6), because $d_i(p)$ and $d_i(q)$ are pixel-independent and correspond to the same depth value. However, that is not the case in this formulation. Therefore, alpha-expansion is not applicable, but message passing algorithms such as loopy belief propagation (LBP) [204] and tree-reweighted message passing (TRW) [194], which are other popular optimization techniques for MRF, can be used. In particular, TRW has been successfully applied to solve many Computer Vision problems including depthmap reconstruction [123, 179], and is used in their work.

Figure 3.8 illustrates the photo-consistency curves and the locations of their local maxima at ten contiguous pixels across an occlusion boundary. Notice that the unknown label is assigned to a pixel at the occlusion boundary (sixth pixel from the top), where the spatial regularization is enforced to assign a correct depth label even where the global maximum of the curve corresponds to a false depth (fourth pixel from the top). Figure 3.9 lists more experimental results together with some intermediate reconstructions for evaluation. As the figure illustrates, a single depthmap contains holes both at the pixels where the unknown state label is assigned, and at the surface regions that are not visible in the reference image. However, it is important to only reconstruct regions with high confidence to minimize the presence of noise in the following merging step. Figure 3.9g illustrates that the model becomes near complete in the superimposition of only two depthmaps.

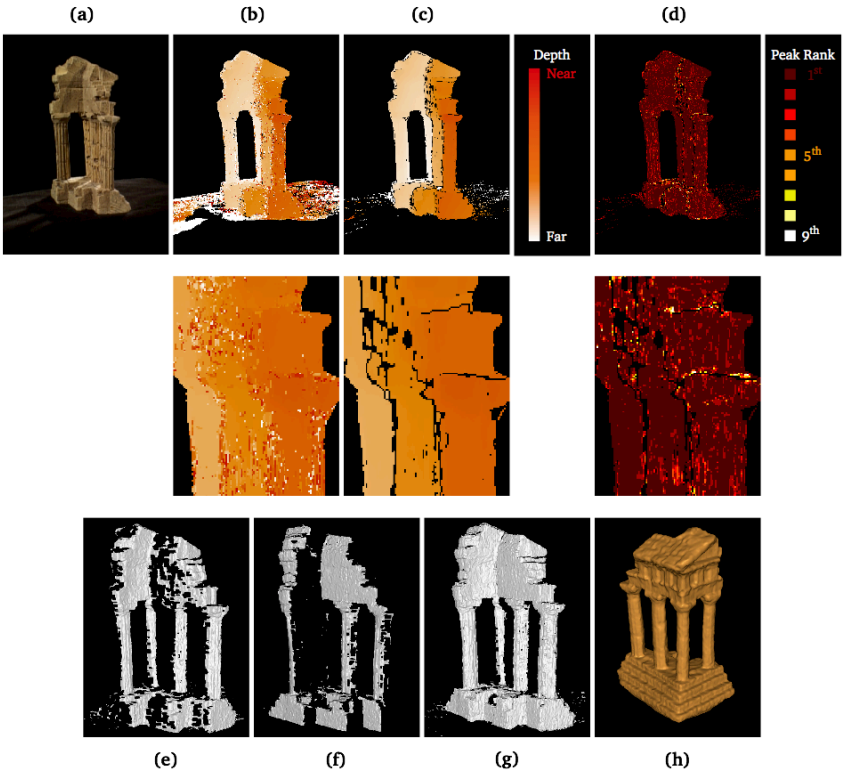



**Figure 3.9:** Results of the hypothesis depthmap algorithm by Campbell, Vogiatzis, Hernández, and Cipolla [48]. Two neighboring images are combined with the reference image (a). If the NCC peak with the maximum score is simply taken, as in [93], (b) is obtained. The result of the algorithm (c) shows a significant reduction in noise. *Unknown* state has also been used to denote clearly occlusion boundaries and remove poorly matched regions. The number of the correct surface peak returned, ranked by NCC score, is displayed in (d) where dark red indicates the peak with the greatest score. The rendered depth-map is shown in (e) along with the neighboring depth-map (f) with (g) showing the two superimposed. The final reconstruction (h) for the sparse temple sequence (16 images) of [165]. (Figure courtesy of Campbell et al.)

### 3.1.5 More Depthmap Reconstruction Algorithms

Numerous depthmap reconstruction algorithms have been proposed in the past in addition to what are described in the previous sections. This section introduces a few more important algorithms and techniques in the literature.

#### Real-Time Plane Sweeping Depthmap Reconstruction

Depthmap reconstruction is not a computationally cheap operation as the photo-consistency function needs to be evaluated over multiple images at every single pixel and at every single hypothesized depth. However, Gallup, Frahm, Mordohai, Yang, and Pollefeys demonstrated that real-time execution is possible with a clever use of GPU [76]. The algorithm is called "Plane Sweeping Stereo", because it sweeps a family of parallel planes in a scene, projects images onto a plane via planar homographies, then evaluates photo-consistency values on each plane. A depth value at each pixel is chosen by the "winner-take-all" strategy, where there are two key features in the algorithm.

First, on the algorithm side, as shown at the bottom row of Figure 3.10, it sweeps along multiple directions, which are extracted from a scene, so that the sweeping directions follow scene structure to be reconstructed. Most algorithms assume that a surface is front-parallel with respect to a reference image to evaluate photo-consistency, which is equivalent to sweep planes along a single fixed direction. As the top row of Figure 3.10 shows, when a scene surface does not follow the plane orientation, correlation windows in different images do not match exactly on the plane. On the other hand, when a scene surface is on the plane, exactly the same 3D surface region projects onto the correlation windows, yielding accurate photo-consistency evaluation. Each sweeping direction produces a depthmap, and multiple depthmaps are merged to produce the final result (See the paper [76] for more details). This strategy is particularly effective for urban scenes where there often exist a few dominant (e.g., Manhattan) directions, which can be extracted from sparse 3D point cloud reconstructed by the SfM system.

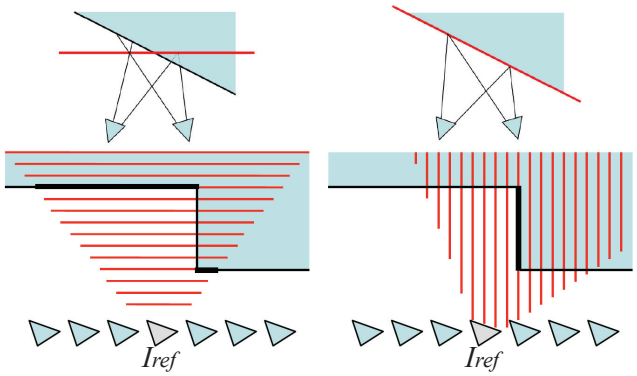



**Figure 3.10:** Plane sweeping stereo algorithm by Gallup, Frahm, Mordohai, Yang, and Pollefeys [76]. Top: photo-consistency evaluation becomes the most accurate and exact when the sweeping plane direction aligns with the surface orientation. Bottom: multiple sweeping directions are extracted from an SfM point cloud, and used for the stereo algorithm. (Figure courtesy of Gallup et al.)

The second key difference is the efficient GPU implementation of image reprojection and photo-consistency evaluation, which achieves real time performance. More concretely, reprojection of an image texture onto a sweeping plane follows planar homography, which is a standard rendering procedure and can be efficiently executed on GPU. After collecting reprojected texture images, the photo-consistency evaluation can also be executed on GPU per pixel. The photo-consistency function used in the original paper was gain-corrected sum of squared differences (SSD). The system was demonstrated on video sequences recorded by the same camera under roughly the constant illumination condition for each sequence, which is one reason why SSD is still effective and the use of more expensive functions such as NCC is not critical.

A dual-core AMD Opteron processor at 2.4GHz and an NVidia GeForce 8800 series GPU were used to process $512 \times 384$ video streams collected at 30 frames per second. Each depthmap computation is performed with 7 images and 48 sweeping planes, which takes only 24 milliseconds. After reconstructing a per-frame depthmap, the system has a step to merge all the depthmaps into a mesh model. The details are referred to their paper [76], while we cover popular fusion techniques in

**Figure 3.11:** Streetside reconstructions by plane sweeping stereo algorithm by Gallup, Frahm, Mordohai, Yang, and Pollefeys [76].

Section 3.3. Figure 3.11 shows several street-side reconstructions by the proposed system. Their largest dataset consists of 170,000 frames, and the system generated nearly 30 billion ($\approx 512 \times 384 \times 170,000$) 3D points based on a simple calculation, ignoring high redundancy and holes in the reconstructions, whose scale is nonetheless much larger than those of competing methods at the time of the publication (2007).

### Second Order Smoothness

MRFs have been successfully used in various depthmap reconstruction algorithms as well as many other Computer Vision tasks that can be discretized into a relatively small number of labels. A typical smoothness prior acts on a pair of pixels, and tries to minimize the depth difference at the two pixels. In the depthmap reconstruction frame-

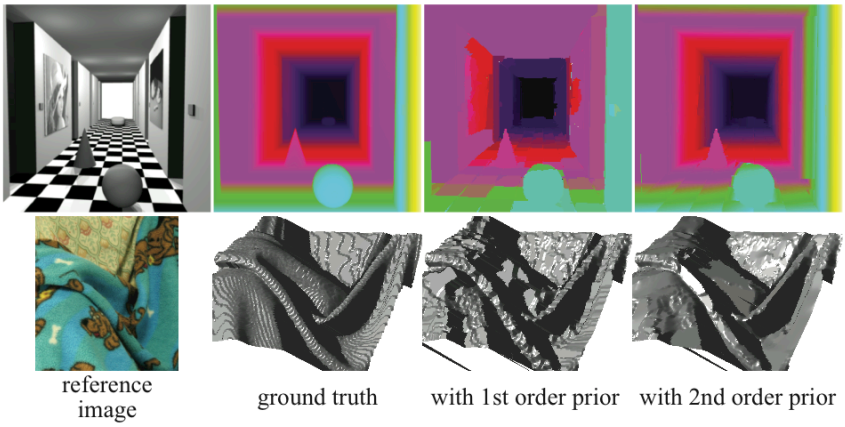



**Figure 3.12:** Depthmap reconstruction algorithm with piecewise planar smoothness prior (2nd order) by Woodford, Torr, Reid, and Fitzgibbon [196]. The top and the bottom rows show their reconstruction results for synthetic and real examples, respectively, compared against a standard MRF formulation with a front-parallel smoothness prior (1st order). The 2nd order prior produces better results both for piece-wise planar and curved surfaces. (Figure courtesy of Woodford et al.)

work, this prior amounts to preferring front-parallel surfaces, where all the pixels have the same depth values. However, in any real scene, most surfaces are not front-parallel with respect to a reference image, and this prior often causes reconstruction artifacts where the assumption does not hold (See Figure 3.12).

Woodford, Torr, Reid, and Fitzgibbon presented a stereo algorithm, which still employs an MRF formulation, but introduces a second order smoothness prior over triple cliques (i.e., three pixels) that enforces piece-wise planar surfaces [196]. More concretely, given three adjacent pixels $(p, q, r)$, the term penalizes $|d_p + d_r - 2d_q|$ as a smoothness cost, which is a finite difference approximation of the second order derivative, and becomes 0 when the first order derivative is constant, that is, when a surface is piece-wise planar. The introduction of triple cliques makes the optimization complicated (at least the cost function is not submodular any more) and a more sophisticated optimization algorithm is required to solve the problem. Details of the optimization algorithm are referred to their paper [196].

The effects of the second order smoothness prior are illustrated in Figure 3.12 for synthetic and real examples. In the top synthetic example, most structure is piece-wise planar and is not reconstructed well by a standard method (i.e., 1st order prior), which produces piece-wise front-parallel surfaces instead. Their algorithm succeeded in reconstructing most piece-wise planar surfaces as expected. The bottom real example is a case where most structure is curved and not really piece-wise planar. Nonetheless, their reconstruction result is much more accurate than that of the standard method, because piece-wise planar smoothness prior is much more flexible and follows more closely curved surfaces in contrast to the front-parallel smoothness prior. The standard method suffers from stair-casing artifacts at many places.

## 3.2 Point-cloud Reconstruction

We have seen in the previous section that multiple depthmaps have been a very popular scene representation and that it is scalable to a larger scene, as the core reconstruction task is still a single depthmap estimation. While depthmaps suffice for various interesting applications such as scene analysis and visualization, their main problem is how to merge them into a global 3D model. There exist many different techniques that merge multiple depthmaps [52, 76, 94, 142, 153, 207]. Their effectiveness depends on the type of noise present in the depthmap. Densely sampled images can substantially improve reconstruction quality at challenging places such as thin structures and depth discontinuities [117]. However, the depthmap quality tends to degrade significantly at depth discontinuities and occlusion boundaries. Such low quality 3D estimates need to be filtered out or suppressed in the process of model merging. Similarly, in a large scene, the same surface region could be visible in many cameras, some of which may be very far. The accuracy of depthmap estimates is typically inversely proportional to the distance to the surface, and depthmap estimates from far cameras need to be again filtered out or suppressed, even at the absence of depth discontinuities or occlusions [65, 69]. Researchers explored ways to estimate multiple depthmaps simultaneously while enforcing

geometric consistency across multiple images [121, 174, 177]. However, these approaches tend to make the optimization problem very large and computationally expensive.

Point-cloud or patch based surface representations overcome these difficulties since they reconstruct a single point-cloud 3D model by using all the input images, while keeping the advantage of easy model manipulation such as merging and splitting. Note that a 3D point with a surface normal estimation or a local region support is referred to as an oriented point or a patch.

A common characteristic of point-cloud reconstruction algorithms is that they make use of an spatial consistency assumption and grow or expand the point-cloud on the surface of the scene during the reconstruction process, as opposed to reconstructing each point independently. This expansion or region-growing idea is not unique in MVS, and has been exploited in many other computer vision tasks such as feature matching, segmentation, and recognition [149, 126, 60]. In the MVS context, Lhuillier and Quan proposed an early work, where patches are expanded in a greedy fashion [132]. Habbecke and Kobbelt also proposed a similar algorithm that iteratively expands patches to reconstruct an object [85]. Similar ideas would be later exploited in the two-view stereo setting for a depthmap reconstruction. Notably, "PatchMatch Stereo" is a successful algorithm [40], which randomly initializes depth values, then refines them based on the local propagation and the random search strategies. Their source of inspiration comes from the "PatchMatch" algorithm [34], which was originally developed for general image matching.

This article focuses the description of the point-based reconstruction to the work by Furukawa and Ponce [74] [3]. The algorithm also follows a greedy expansion approach, but one key difference is that it iterates between the expansion and the filtering steps after reconstructing an initial seed of patches via feature matching. The filtering step

[3] Their open-source software "Patch-based Multi-View Stereo" (PMVS) has been extensively used by diverse communities, including artists, civil engineers and archaeologists for academic purposes, as well as corporate companies such as Industrial Light & Magic and Weta Digital for real film production and Google for digital map making.

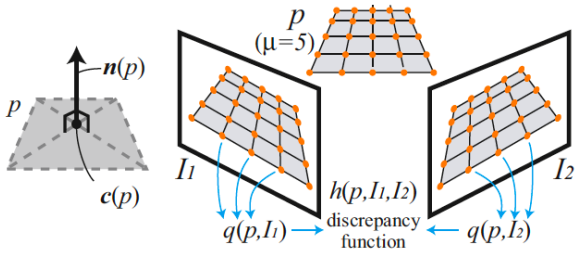



**Figure 3.13:** Left: a patch $p$ is a (3D) rectangle with its center and normal denoted as $\mathbf{c}(p)$ and $\mathbf{n}(p)$, respectively.

analyzes consistency of patches across all the views and removes falsely reconstructed ones. We will first explain several fundamental building blocks for the algorithm, then provide details of the three processing steps, namely, *initial feature matching*, *expansion* and *filtering*. Note that a simplified version of the algorithm is described in this article to be concise, and the full details are referred to their journal paper [74].

### 3.2.1 Key Elements

The output of their algorithm is a set of patches for an entire scene, and this section explains their image-based data structure to keep track of reconstructed patches, and the patch model/optimization technique that directly estimates both the depth and the surface normal.

#### Patch Model

A patch $p$ is essentially a local tangent plane approximation of a surface, whose geometry is determined by its center $\mathbf{c}(p)$ and unit normal vector $\mathbf{n}(p)$. Therefore, while a typical photo-consistency function only takes the position as an input for the evaluation, one can extend the function to take both the position as well as the surface normal as input. The photo-consistency function is simply evaluated by using the patch as a proxy geometry to sample pixel colors (See Figure 3.13). While NCC is the core metric as in many other algorithms, they apply a robust function to the NCC score to make their photo-consistency ro-

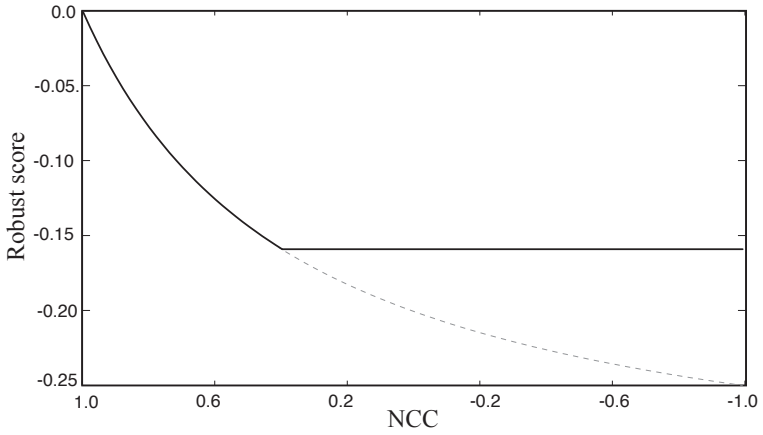



**Figure 3.14:** A robust function is applied to the raw NCC score. The curve shows that an NCC score below 0.4 is automatically ignored, where it has more effects (slope in the curve) when its value is closer to 1.0.

bust against outlier signals (See Section 3.1.2 for a similar robust photo-consistency technique). Let $\mathcal{C}$ denote the NCC score, then the robust photo-consistency is defined as $-\mathcal{C}'/(3\mathcal{C}'+1)$, where $\mathcal{C}' = min(\tau, 1-\mathcal{C})$. $\tau$ is a truncation threshold and set around 0.4. The shape of the robust function is illustrated in Figure 3.14.

Having defined the photometric consistency measure for a patch as a function of its position and the normal, reconstructing a patch is simply achieved by maximizing the photo-consistency function with respect to those parameters. At first sight, the function has five parameters to be optimized, because the position consists of three parameters and the normal consists of two parameters. However, a patch should not move tangentially on a surface during optimization, where only its vertical offset of the positional parameters should be optimized. The vertical direction depends on the patch normal, which is also optimized. Therefore, in practice, the perpendicular direction is fixed before and throughout the optimization, where one parameter for position and two parameters for normal are optimized via a standard non-linear least squares technique.

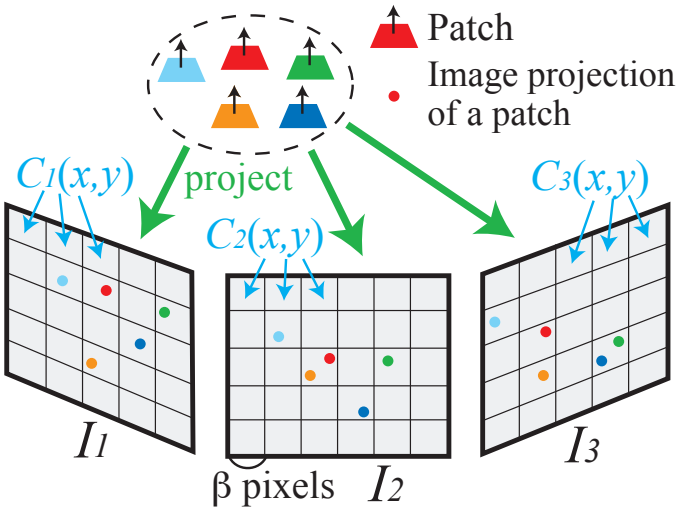



**Figure 3.15:** Image projections of reconstructed patches in their visible images are used to perform fundamental tasks such as accessing neighboring patches, enforcing regularization, etc. See text for more details.

### Image-based Data Structure

The main advantage of the patch based surface representation is its flexibility. However, due to the lack of connectivity information, it is not easy to just search or access neighboring patches, then enforce regularization, for instance. In their approach, image projections of reconstructed patches in the visible images are used to help performing these tasks. Concretely, they associate with each image $I_i$ a regular grid of $\beta_1 \times \beta_1$ pixels cells $C_i(x, y)$ as in Fig. 3.15 ($\beta_1 = 2$ in their experiments). Given a patch $p$ and its visible images $V(p)$, which is estimated as a part of the reconstruction process, they project $p$ into each image in $V(p)$ to identify the corresponding cell. Then, each cell $C_i(x, y)$ remembers the set of patches $Q_i(x, y)$ that project into it. Neighboring patches can be collected by looking at neighboring cells in the visible images.

### 3.2.2 Algorithm

Their patch-based MVS algorithm attempts to reconstruct at least one patch in every image cell $C_i(x, y)$. It is divided into three steps: (1) initial feature matching, (2) patch expansion, and (3) patch filtering. The purpose of the initial feature matching step is to generate a sparse set of patches (possibly containing some false positives). The expansion and the filtering steps are iterated $n$ times (typically $n = 3$) to make patches dense and remove erroneous matches. The three steps are detailed in the following sections.

#### Initial Feature Matching

The first step of the algorithm is to detect blob and corner features in each image using the Difference-of-Gaussian (DoG) and Harris operators [178]. Consider an image $I_i$ with its optical center denoted by $O(I_i)$. For each feature $f$ detected in $I_i$, they collect in the other images the set $F$ of features $f'$ of the same type (Harris or DoG) that lie within two pixels from the corresponding epipolar line, and triangulate the 3D points associated with the pairs $(f, f')$. Then, they consider these points in order of increasing distance from $O(I_i)$ as potential patch centers, and attempt to generate a patch from the points one by one until they succeed using the following procedure. Given a pair of features $(f, f')$, they first construct a patch candidate $p$ with its center $\mathbf{c}(p)$ and normal vector $\mathbf{n}(p)$ initialized as

$$\mathbf{c}(p) \leftarrow \{\text{Triangulation from } f \text{ and } f'\}, \tag{3.9}$$

$$\mathbf{n}(p) \leftarrow \overrightarrow{\mathbf{c}(p)O(I_i)}/|\overrightarrow{\mathbf{c}(p)O(I_i)}|. \tag{3.10}$$

The set of visible images $V(p)$ is initialized by collecting a fixed number (often five) of nearby views based on the viewing angle differences. Then, the patch optimization procedure is applied to refine these parameters. After the optimization, $V(p)$ contains the images whose pairwise photo-consistency with $I_i$ is more than a certain threshold. The patch is kept as a success if $|V(p)|$ is at least $\gamma_v$, which is often set to 3. Empirically, this heuristic has proven to be effective in selecting mostly correct matches at a modest computational expense. The over-

all algorithm description for this step is given in Fig. 3.16. Of course, this relatively simple procedure may not be perfect and yield mistakes, but the filtering step will handle such errors.

### Expansion

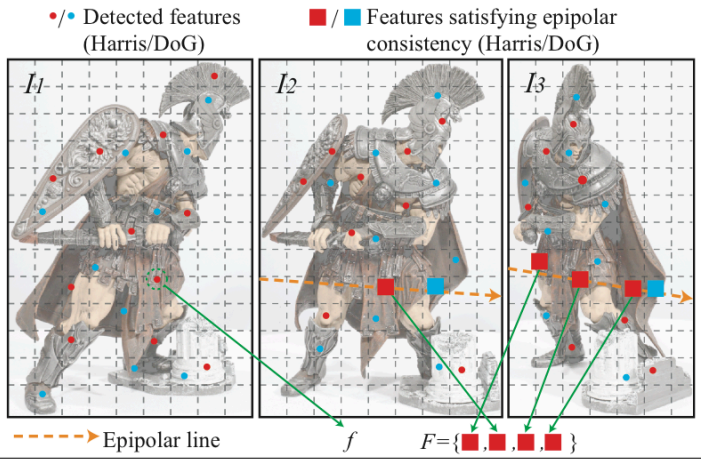
The goal of the expansion step is to reconstruct at least one patch in every image cell $C_i(x, y)$, where they repeat taking existing patches and generating new ones in nearby empty spaces. More concretely, given a patch $p$, they first identify a set of *neighboring* image cells $\mathbf{Cells}(p)$ that do not contain any patches yet:

$$\begin{aligned} \mathbf{Cells}(p) \quad = \quad & \{C_i(x', y') | \qquad (3.11) \\ & p \in Q_i(x, y),\ Q_i(x', y') = \emptyset,\ |x - x'| + |y - y'| = 1\}. \qquad (3.12) \end{aligned}$$

For each collected image cell $C_i(x, y)$ in $\mathbf{Cells}(p)$, the following expansion procedure is performed to generate a new patch $p'$. They first initialize $\mathbf{n}(p')$ and $V(p')$ by the corresponding values of $p$. $\mathbf{c}(p')$ is, in turn, initialized as the point where the viewing ray, passing through the center of $C_i(x, y)$, intersects the plane containing the patch $p$. They then refine $\mathbf{c}(p')$ and $\mathbf{n}(p')$ by the optimization procedure described in Sect.3.2.1. They remove images from $V(p')$, whose average pairwise photo consistency score with the remaining images in $V(p')$ is less than a threshold. They also add images to $V(p')$ if their average pairwise photo consistency scores are above the threshold. Finally, if $|V(p')| \geq \gamma_v$, they accept the patch as a success and update $Q_i(x, y)$ for its visible images. The process repeats until the expansion process is performed from every patch that has been reconstructed. The overall algorithm description is given in Fig. 3.17.

### Filtering

The expansion step is greedy and solely relies on photo consistency measures for reconstructing patches, where it is difficult to avoid generating any erroneous patches. In the last step of the algorithm, the following two filters are used to remove erroneous patches. The first filter relies on visibility consistency. Let us define that patches $p$ and $p'$




*Input:* Features detected in each image.
*Output:* Initial sparse set of patches $P$.

---

$P \leftarrow \emptyset$;
*For* each image $I$ with optical center $O(I)$
  *For* each feature $f$ detected in $I$
    $F \leftarrow$ {Features satisfying the epipolar consistency};
    Sort $F$ in an increasing order of distance from $O(I)$);
    *For* each feature $f' \in F$
      // Test a patch candidate $p$;
      Initialize $\mathbf{c}(p), \mathbf{n}(p)$ and $R(p)$; // Eqs. (3.9, 3.10)
      Initialize $V(p)$ based on viewing angles;
      Refine $\mathbf{c}(p)$ and $\mathbf{n}(p)$;
      Update $V(p)$ based on the pairwise photo-consistency with $I_i$;
      If $|V(p)| < \gamma_v$
        Go back to the innermost *For* loop (failure);
      Add $p$ to the corresponding cells $Q_j(x, y)$ after projection;
      Remove features from the cells where $p$ was stored;
      Add $p$ to $P$;
      Exit innermost *For* loop;

**Figure 3.16:** Feature matching algorithm. Top: An example showing the features $f' \in F$ satisfying the epipolar constraint in images $I_2$ and $I_3$ as they are matched to feature $f$ in image $I_1$ (this is an illustration only, not showing actual detected features). Bottom: The matching algorithm.

*Input:* Patches $P$ from the feature matching step.
*Output:* Expanded set of reconstructed patches.

---

*While* $P$ is not empty
  Pick and remove a patch $p$ from $P$;
  *For* each image cell $C_i(x, y)$ containing $p$
    Collect a set $\mathbf{C}$ of image cells for expansion;
    *For* each cell $C_i(x', y')$ in $\mathbf{C}$
      // Create a new patch candidate $p'$
      $\mathbf{n}(p') \leftarrow \mathbf{n}(p), \quad V(p') \leftarrow V(p)$;
      Initialize $c(p')$ via ray to plane intersection;
      Refine $\mathbf{c}(p')$ and $\mathbf{n}(p')$; // (Sect.3.2.1)
      Update $V(p')$;
      If $|V(p')| < \gamma_v$
        Go back to *For*-loop (failure);
      Add $p'$ to $P$;
      Add $p'$ to corresponding $Q_j(x, y)$;

**Figure 3.17:** Patch expansion algorithm. The expansion and the filtering procedure is iterated $n(= 3)$ times to make patches dense and remove outliers.

are *neighbors* if their distance along the normals is less than a threshold:

$$|(\mathbf{c}(p) - \mathbf{c}(p')) \cdot \mathbf{n}(p)| + |(\mathbf{c}(p) - \mathbf{c}(p')) \cdot \mathbf{n}(p')| < \gamma_d. \tag{3.13}$$

$\gamma_d$ is the upper-bound on the allowed amount of vertical offset between the two patches. Let $U(p)$ denote the set of patches $p'$ that are inconsistent with the current visibility information—that is, $p$ and $p'$ are not neighbors, but are stored in the same cell of one of the images where $p$ is visible (Fig. 3.18). Then, $p$ is filtered out as an outlier if the following inequality holds

$$|V(p)|(1 - \mathcal{C}(p)) < \sum_{p_i \in U(p)} 1 - \mathcal{C}(p_i). \tag{3.14}$$

$\mathcal{C}(p)$ is the average pairwise photo-consistency score of $p$. Intuitively, when $p$ is an outlier, both $1-\mathcal{C}(p)$ and $|V(p)|$ are expected to be small, and $p$ is likely to be removed. In the second filter, we enforce a weak form of regularization: For each patch $p$, we collect the patches lying in its own and adjacent cells in all images of $V(p)$. If the proportion

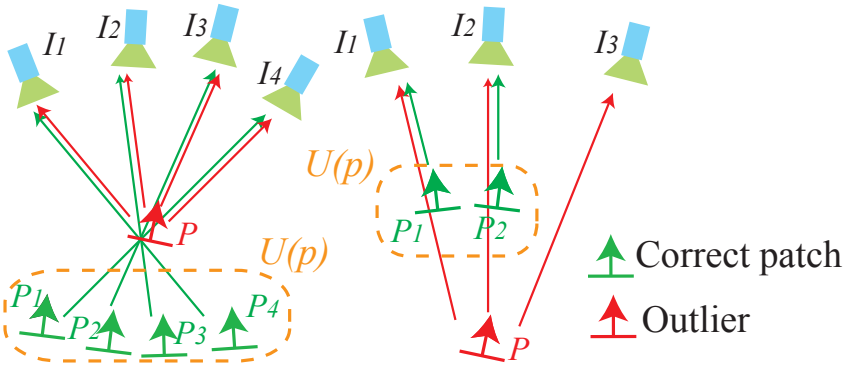



**Figure 3.18:** The first filter enforces global visibility consistency to remove outliers (red patches). An arrow pointing from $p_i$ to $I_j$ represents a relationship $I_j \in V(p_i)$. In both cases (left and right), $U(p)$ denotes a set of patches that is inconsistent in visibility information with $p$.

of patches that are neighbors of $p$ (Eq. 3.13) in this set is lower than 0.25, $p$ is removed as an outlier.

### 3.2.3 Reconstruction Results

Figure 3.19 shows a sample input image, the image resolution, and the number of input images for each dataset. Patch based representation is flexible and can handle both "object" like datasets (top row of Figure 3.19), where cameras surround an object, and"scene" like datasets, where cameras are surrounded by a scene. Their reconstructed patches are shown in Figure 3.20. Note that patches are dense and look like a surface model, but are merely point clouds. The figure illustrates that reconstructed patches are free from noise and shows the robustness of the algorithm despite that patches are reconstructed independently without explicit regularization. The bottom half of Figure 3.20 shows the polygonal surface models converted from the patches, which verifies the geometric accuracy of reconstructed patches (See Section 3.3 for surface meshing techniques).


**Figure 3.19:** Sample input images for point-cloud reconstruction algorithm [74].

## 3.3 Volumetric data fusion

The creation of high quality meshes has been a fundamental problem in Computer Graphics, where laser range sensors were often used as an input. There is a vast literature in converting the generated 3D point clouds into clean mesh models. Seminal work by Curless and Levoy [52] proposes an algorithm to accumulate surface evidence into a voxel grid using signed distance functions, where the final mesh is extracted as the zero iso-surface of the aggregated signed distance functions. Poisson Surface Reconstruction by Kazhdan et al. [115, 41, 116] is another successful and popular meshing software from an oriented point cloud. There also exist methods to reconstruct a mesh from unoriented point cloud, and hence, unsigned distance functions [103, 29]. Some of these techniques, notably the Poisson Surface Reconstruction software,

**Figure 3.20:** Reconstruction results of a point-cloud reconstruction algorithm [74].

are robust against noisy point clouds with full of reconstruction holes, which are more common in Computer Vision applications. However, more sophisticated optimization and regularization techniques have also been sought in tackling challenges by Computer Vision researchers. This section explains such techniques.

Volumetric surface extraction is flexible and the input 3D information can come from many different sources such as photo-consistency volumes, depthmaps, MVS point clouds, laser scanned 3D points, or any combination of those. It is a challenging task to fuse such diverse set of 3D measurements into a single clean mesh model with the right

topology. A standard yet powerful solution is to accumulate evidence in a 3D voxel grid and extract a surface model via Marching Cube algorithm [134]. One formulation is to compute a signed distance function field over the voxel grid, then pose a surface reconstruction as zero iso-surface extraction [52, 207]. The other formulation is to pose as a 3D binary segmentation problem, where the boundary of the two segments can be extracted as a surface model [191, 94, 169, 68, 207]. The latter formulation is more popular in Computer Vision, and we introduce two algorithms that formulate the reconstruction problem as a 3D binary segmentation via Markov Random Field. Their differences are in the 3D space discretization scheme. [4]

### 3.3.1 Volumetric Graph-Cuts on a Voxel Grid

Given a bounding box that contains the solution surface, the space is discretized with a voxel grid. In general, the input 3D information (e.g., a set of depthmaps) can be analyzed to determine a bounding box. The problem is formulated to label each voxel as "interior" or "exterior", where the label boundary can be extracted as a surface model. Let $k_v$ denote a variable for a voxel $v$, which takes one of these two labels. The objective is to find the optimal label assignment to all the voxels that minimizes the following cost function:

$$E(\{k_v\}) = \sum_v \Phi(k_v) + \sum_{(v,w)\in\mathcal{N}} \Psi(k_v, k_w). \tag{3.15}$$

The first term is the summation of per voxel cost over the entire domain, where $\Phi(k_v)$ encodes the cost of assigning a label to voxel $v$. The second term is the summation over all the pairs of adjacent voxels denoted as $\mathcal{N}$. Notice its resemblance to the Markov Random Field formulation for a depthmap reconstruction (3.2). $\Phi$ is a *unary term*, which depends on a single variable, while $\Psi$ is a pairwise *interaction term*. We first explain how these cost terms should be set, then introduce optimization algorithms to solve the problem.

[4] In the 90's, Roy and Cox proposed a reconstruction algorithm with the max-flow min-cut optimization method, whose problem formulation is very similar [158]. However, they reconstruct a single depthmap (disparity map) as opposed to a full polygonal mesh model.

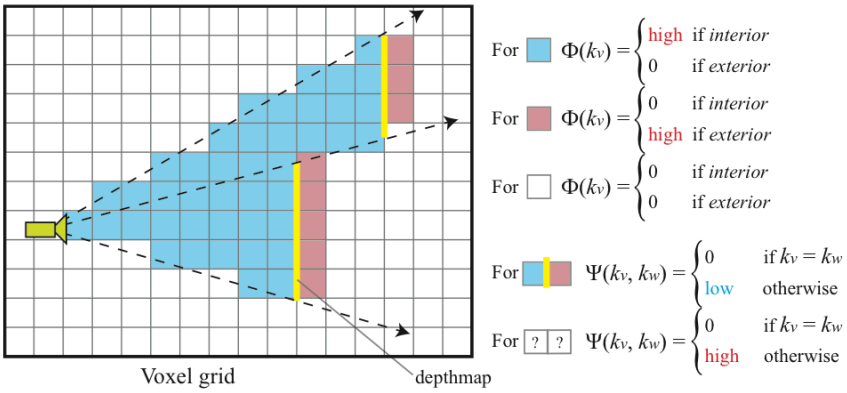



**Figure 3.21:** An example of how 3D MRF cost function should be set from a single depthmap.

### Setting Ψ and Φ

Suppose a single depthmap is given as a toy example, $\Phi(k_v)$ and $\Psi(k_v)$ should be set as follows (See Figure 3.21). First, all the voxels that are between the camera center and the depthmaps should be "exterior", and hence $\Phi(k_v)$ should be set so that the penalty of assigning "interior" label becomes expensive for such "exterior" voxels. On the contrary, all the voxels that are immediately behind the depthmap should be "interior", and hence $\Phi(k_v)$ should be set so that the penalty of assigning "exterior" label becomes expensive for such "interior" voxels. $\Psi(k_v, k_w)$ can be set as a simple smoothness prior, that is, set to 0 when $k_v = k_w$, and is assigned a penalty when the labels are different. $\Psi(k_v, k_w)$ can be also data adaptive. More concretely, $\Psi(k_v, k_w)$ should be small where the depthmap surface exists, so that the label boundary is more likely to occur there. Therefore, while $\Psi(k_v, k_w)$ is still set to 0 when $k_v = k_w$, the amount of penalty at label difference can be set inversely proportional to the amount of depth evidence. When multiple depthmaps are provided as input, the information is simply accumulated in $\Phi$ and $\Psi$ over all the inputs.

Of course, this is just one way to set $\Phi$ and $\Psi$, and there exist many variants [94, 169, 68]. In the earlier days of multi-view stereo research, $\Psi$ was often the only source of data information, while $\Phi$ was defined

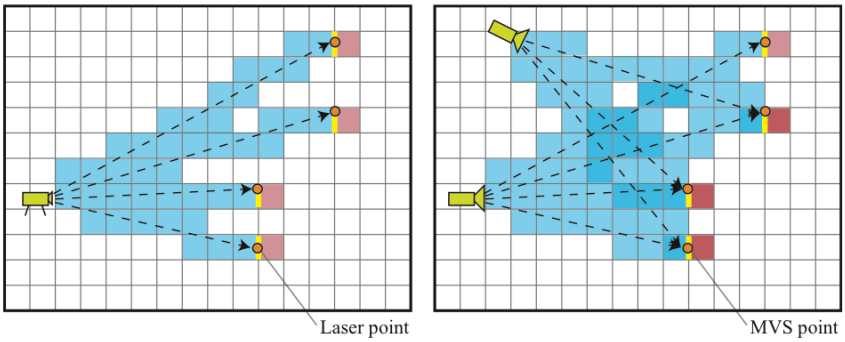



**Figure 3.22:** 3D scene information inferred from a single depthmap (left) is similar to that from a point cloud. The main difference is that a single 3D point is associated with multiple cameras for a point cloud, but with a single camera for a depthmap. Both information can be used in exactly the same way to accumulate geometric evidence in the 3D space.

as a constant *ballooning term* to prefer a solution with non-zero interior volume:

$$\Phi(k_v) = \begin{cases} 0 & \text{if } \textit{interior} \\ \alpha(\text{constant}) & \text{if } \textit{exterior}. \end{cases} \tag{3.16}$$

This formulation still yields high quality mesh models, because photo-consistency information is usually very strong, and $\Psi$ alone can provide enough information. However, reconstruction results become more sensitive to the choice of the parameter $\alpha$. When $\alpha$ is 0, a trivial solution exists, where the same label (either "interior" or "exterior") is assigned to all the voxels, producing empty output but zero errors. On the other hand, if $\alpha$ is too large, details would be lost. In general, this formulation has a bias to produce models with smaller volume and lose *thin* structures (i.e., *shrinkage bias*). As a result using 3D visibility information in $\Phi$ (as shown in Figure 3.21) is always preferred to setting $\Phi$ as a constant ballooning term.

When the input 3D information is a point cloud from a laser range sensor, the data can be treated exactly as a depthmap, where the center of laser and a 3D point are to be treated as the camera center and a pixel on the depthmap, respectively (See Figure. 3.22).

### Optimization

The MRF formulation in (Equation 3.15) has only two possible labels ("interior" or "exterior") and is much simpler than that in (Equation 3.2) for depthmap reconstruction. Therefore, the problem in the form of (Equation 3.15) can be solved exactly and efficiently with a graph-cuts algorithm, as long as each pairwise term $\Psi(k_v, k_w)$ is submodular [122]:

$$\Psi(interior, interior) + \Psi(exterior, exterior) \leq \tag{3.17}$$

$$\Psi(interior, exterior) + \Psi(exterior, interior) \tag{3.18}$$

Usually, pairwise terms satisfy the above condition for our reconstruction problems, because the submodularity goes well with the smoothness prior, and the left hand side of the inequality is typically 0.

### Reconstruction Results

Figure 3.23 shows the reconstruction results by Vogiatzis, Torr, and Cipolla [191], who proposed one of the earliest volume fusion techniques based on graph-cuts in 2005. They used the constant ballooning term and took the photo-consistency volume as the input.

One limitation of the use of a voxel grid is that the memory allocation quickly becomes very expensive. One effective solution to this problem is an octree data structure, which is essentially an adaptive voxel grid. The grid is subdivided based on the input depthmaps so that the grid is subdivided where the surface likely exists. Figure 3.24 shows input images, and a sample reconstruction result of a volumetric fusion technique with the octree space discretization by Hernández, Vogiatzis, and Cipolla [94]. The top row shows four of the seventy two images of a "crouching man" sculpture by the modern sculptor Antony Gormley. The second row shows the reconstruction when $\Phi(k_u)$ is set to a constant ballooning. In the third row, $\Phi(k_u)$ is set to the 3D visibility information computed from the depthmap data. Due to the shrinkage bias, the reconstruction with the constant ballooning term fails in reconstructing deep concavities in various parts of the model. Figure 3.25 illustrates the unary and pairwise potentials on a octree that are collected from multiple depthmaps [94]. The unary potential alone clearly

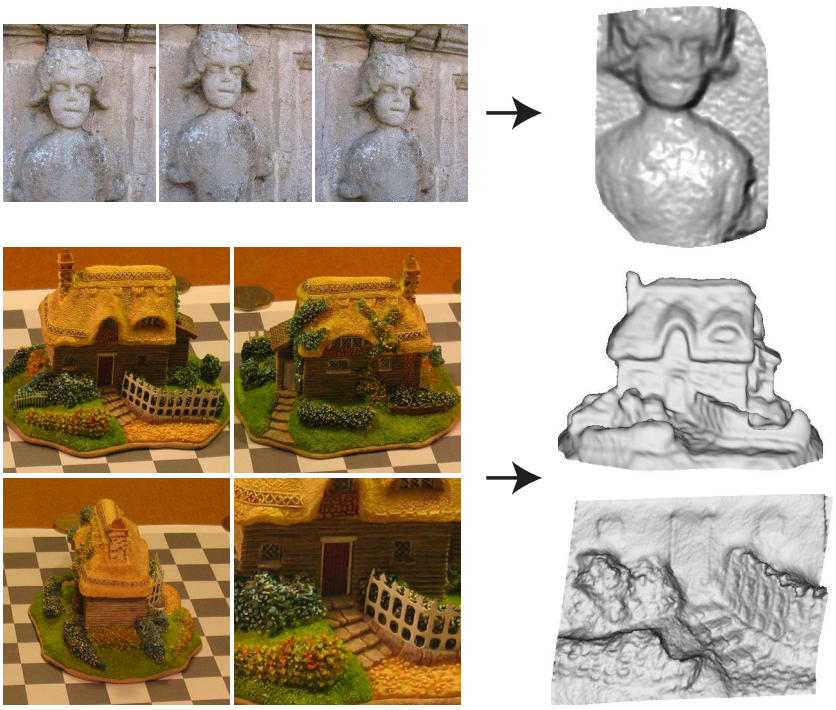


**Figure 3.23:** One of the earliest volume fusion techniques based on the volumetric graph-cuts by Vogiatzis, Torr and Cipolla [191]. (Figure courtesy of Vogiatzis et al.)

shows the separation between the object interior and exterior spaces, where the pairwise terms concentrate on surface boundaries. The right of Figure 3.25 shows the octree structure.

Sinha, Mordohai, and Pollefeys presented a similar approach [169], where a grid of tetrahedra (with a technique similar to the octree to save space) is used to discretize the space, and the graph-cuts algorithm is used to extract a surface model. In their work, the photo-consistency is used to drive the tetrahedral subdivision (See Figure 3.26). They refine the model for polishing after extracting the mesh from the tetrahedral grid. The figure shows the model after the refinement. Refer to Section 3.4 for the details of similar mesh refinement techniques.

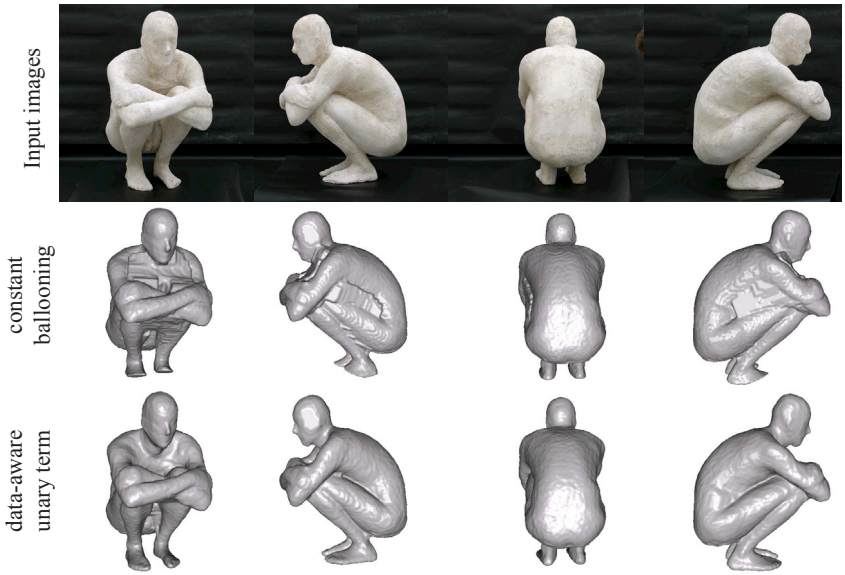



**Figure 3.24:** Reconstruction results of a volumetric graph-cuts technique by Hernández, Vogiatzis, and Cipolla [94]. The second row shows the reconstruction when the unary term is set to a constant ballooning term as in Equation3.16. The third row shows the result when the 3D information is encoded into the unary term, which is more accurate especially at deep concavities.

### 3.3.2 Volumetric Graph-Cuts on Delaunay Tetrahedralization

Uniform voxel (or tetrahedral) grid is a simple but effective way to encompass the reconstruction space for volumetric data-fusion. While octree can be used to reduce the memory footprint, these methods still require the knowledge of a scene extent. A better alternative is to first reconstruct a sparse 3D point cloud of a scene, then use the 3D points as nodes of the space discretization grid. This approach does not require the specification of a scene extent. Furthermore, space discretization is adaptive to the available 3D information: a space with denser point samples is discretized more finely, and a space with fewer point samples is discretized sparsely.

Labatut, Pons, and Keriven presented an algorithm that first matches image features based on the SIFT descriptor [135] to recon-

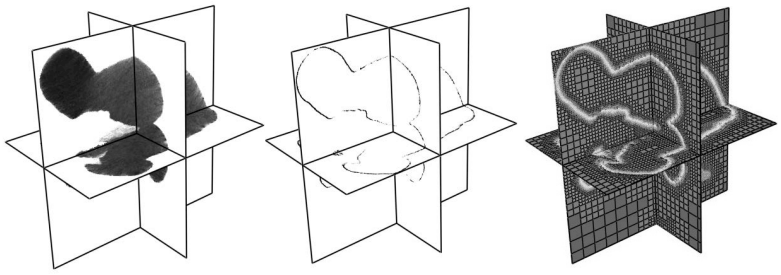


**Figure 3.25:** Visualization of (left) multi-resolution octree grid structure, (middle) pairwise term $\Psi(k_v, k_w)$, and (right) unary term $\Phi(k_v)$ for a volumetric graph-cuts technique by Hernández, Vogiatzis, and Cipolla [94].

struct sparse 3D points, then performs 3D Delaunay triangulation of the points to discretize the space [129]. Figure 3.27 shows the 2D illustration of the Delaunay triangulation, where red points denote the input point samples and divide the reconstruction space (i.e., convex hull of the point samples) into a set of triangles. Given a 3D Delaunay triangulation, they solve a volumetric MRF problem that is very similar to the one in the previous section to extract a surface model, whose details are given next.

The volumetric MRF consists of the unary and pairwise terms as before, where the unary term is defined for each tetrahedron, and the pairwise term is defined for each face that is shared by the two tetrahedra. In the work of Labatut et al. [129], for each ray connecting a reconstructed point and one of its visible camera, the unary and pairwise terms are calculated as follows (Figure 3.28). First, the unary term is defined only for a triangle that either contains the camera center ($p_0$ in the figure) or is immediately behind the 3D point ($q_2$ in the figure):

$$\Phi(p_0) = \begin{cases} \infty & \text{if } \textit{interior} \\ 0 & \text{if } \textit{exterior}, \end{cases} \tag{3.19}$$

$$\Phi(q_2) = \begin{cases} 0 & \text{if } \textit{interior} \\ \alpha & \text{if } \textit{exterior}. \end{cases} \tag{3.20}$$

The unary term at $p_0$ enforces that the cell containing the camera

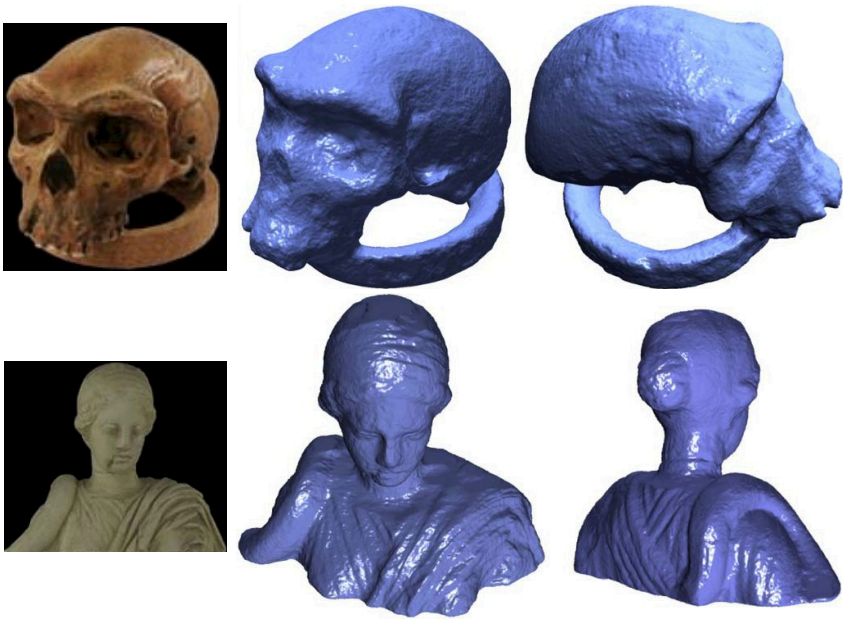


**Figure 3.26:** Reconstruction results of a volumetric graph-cuts technique by Sinha, Mordohai and Pollefeys [169]. (Figure courtesy of Sinha et al.)

center must be in the scene exterior space, while the term at $q_2$ adds a bias for the cell to be in the scene interior space. For every pair $(p_1, q_1)$ of triangles whose common face intersects with the visible ray, where $p_1$ is on the camera side, a pairwise interaction term is defined as follows to penalize a pairwise configuration $(p_1 = interior, q_1 = exterior)$:

$$\Psi(p_1, q_1) = \begin{cases} \beta & \text{if } p_1 = interior \text{ and } q_1 = exterior \\ 0 & \text{in the other 3 cases.} \end{cases} \tag{3.21}$$

For every ray between a 3D point and its visible camera, the above unary and pairwise terms are computed and accumulated as opposed to overwriting. Due to the pairwise term construction, the above energy is guaranteed to be submodular. Therefore, the graph-cuts algorithm can be used to find the optimal labeling to triangular cells, where the label boundary is extracted as a surface model.

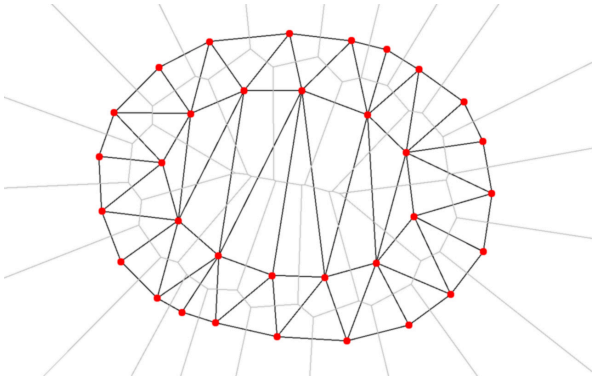


**Figure 3.27:** Delaunay tetrahedralization on the input 3D point cloud is used to discretize the space into 3D cells. (Figure courtesy of Labatut et al.)

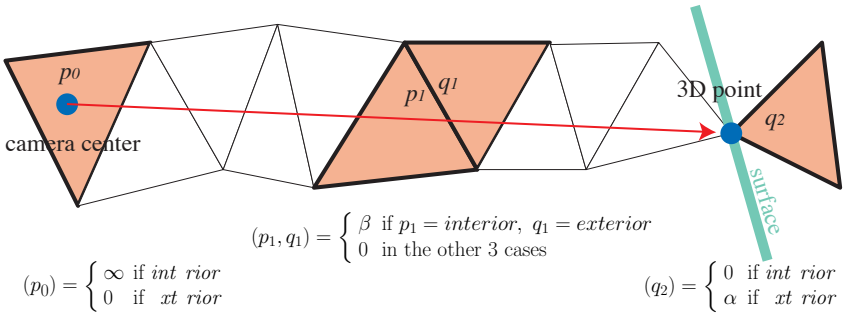



**Figure 3.28:** Given a single pixel in a depthmap or a single 3D point, unary and binary cost terms are defined for cells that intersect with the visible ray.

Note that explicit regularization term to penalize label changes is usually necessary at every pair of adjacent cells for a uniform voxel grid structure. The reason is that many voxels may not be intersected by visible rays and hence have a zero unary potential, especially where the input 3D evidence is sparse and/or noisy. On the other hand, in the case of a 3D Delaunay triangulation, each vertex of a cell is a reconstructed 3D point and most cells near the surface have non-zero unary potential, and hence, explicit regularization in the pairwise term is not crucial.

Figure 3.29 presents some reconstruction examples of the Delaunay tetrahedralization technique proposed by Labatut, Pons, and

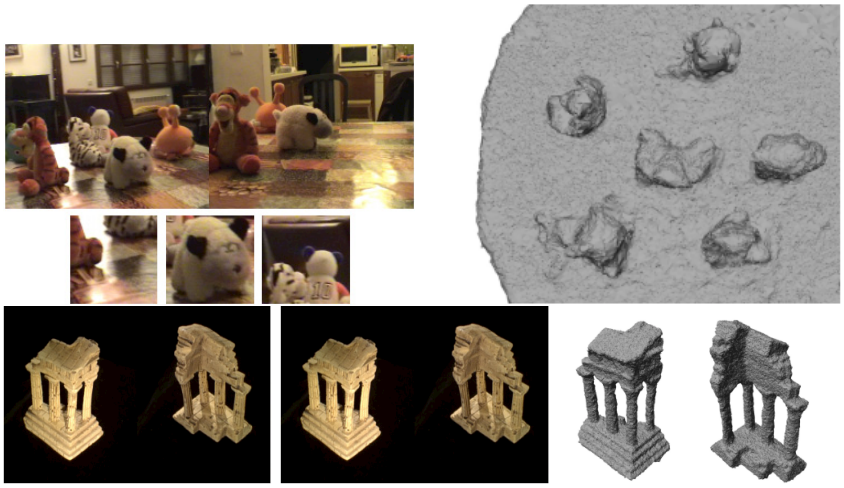


**Figure 3.29:** Reconstruction results of a Delaunay tetrahedralization based volumetric graph-cuts approach [129]. (Figure courtesy of Labatut et al.)

Keriven [129]. Although the reconstructed surfaces may look slightly noisy, this is their mesh initialization step, which is the starting point for the multi-view stereo refinement step explained in the next section.

A problem with the presented technique is that, if the 3D evidence is weak, it will miss thin structures due to the MRF regularization. Jancosek and Pajdla extended the formulation from Labatut et al. [129], and added a step to reinforce the evidence of structure by analyzing the gradient of exterior evidence [109]. The intuition is visualized in Figure 3.30 with a toy example. Suppose a circular object is placed in the middle of a square-shaped room (left in the Figure), where it is hard to reconstruct the object. The room surfaces are reconstructed well and have ample interior evidence (highlighted in red), where inside of the room is full of exterior evidence (highlighted in blue) except at the object in the middle. The figure indicates that sudden absence of exterior evidence can suggest the presence of a scene interior. More concretely, interior evidence is reinforced where the derivative of the exterior evidence is high. This reasoning may not be always true, but in practice works well and leads to better reconstructions especially at thin structures. Their reconstruction examples are shown in Figure 3.31.

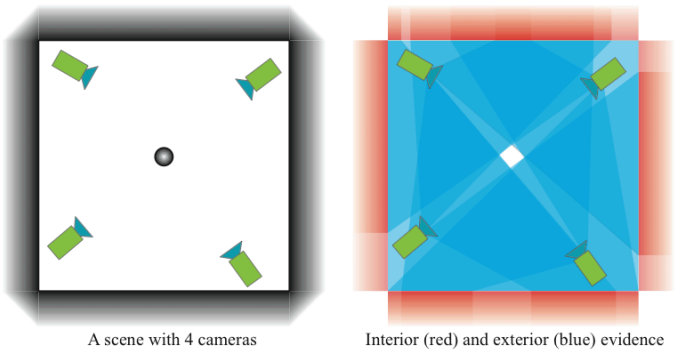



**Figure 3.30:** Gradient of exterior evidence can be used to reinforce the evidence of structure [109]. The circular structure in the middle is hard to be reconstructed, but sudden drops in the amount of exterior evidence suggest the existence of a structure there.

## 3.4 MVS Mesh Refinement

As illustrated in Figure 3.2, the final refinement step can be added to the end of any other MVS algorithms that produce a mesh model. In the refinement step, all the images are once again used to evaluate the photometric consistency score over the surface, where vertex locations are optimized to increase the score iteratively. In the early days of MVS research, the surface topological change was a critical issue during refinement, because the topology of the initial model and that of the true shape may not be equivalent. For this reason the level-set formulation was an attractive surface representation during surface refinement as in a seminal work by Faugeras and Keriven [58]. However, as more sophisticated and robust reconstruction techniques were developed, such as volumetric data fusion techniques in Section 3.3, better mesh initialization could be obtained to start the refinement process. A high computational and memory cost was also a limiting factor of the level-set formulation for high resolution reconstruction. As a result the triangular mesh has become a popular surface representation for the refinement techniques [175, 93, 154, 78, 208, 70, 193, 74, 55]. Note also that topology changes can still be modeled with a triangular mesh representation by checking for surface self intersections at each

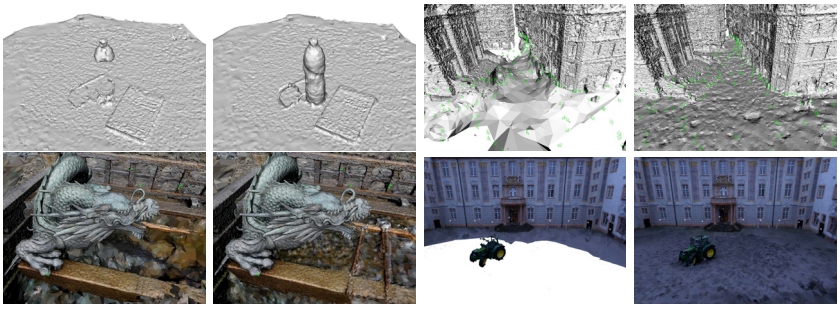


**Figure 3.31:** Jancosek and Pajdla modify the Delaunay tetrahedralization based volumetric graph-cuts approach to handle "weak" surfaces that would have been missed with a standard method. (Figure courtesy of Jancosek et al.)

iteration [208], although the behavior of these techniques are not well understood and may pose issues for challenging cases.

In the remainder of this section, we first explain algorithmic details and framework that are common to many mesh refinement methods, provide specific examples, then finally show their reconstruction results.

### 3.4.1 Common Framework

Mesh refinement algorithms move the location of each vertex $v_i$ one by one or simultaneously, while minimizing an objective function defined over the mesh model. The objective function $E(\{v_i\})$ typically consists of a data term $E_p$, which is based on a photometric consistency measure, and a regularization term $E_r$, which measures the smoothness of the mesh. Optionally, when image silhouettes are given as input, silhouette consistency measure $E_s$ can be added to enforce that the mesh is consistent with the image silhouettes:

$$E(\{v_i\}) = E_p(\{v_i\}) + E_r(\{v_i\}) + [E_s(\{v_i\})]. \tag{3.22}$$

Gradient descent is usually the method of optimization, where the movement of each vertex at each iteration is determined by the gradient of the objective function. More concretely, let $(x_i, y_i, z_i)$ denote the 3D coordinates of $v_i$. The gradient of the energy is computed for each

vertex $v_i$ based on the current mesh:

$$\left(\frac{\partial E}{\partial x_i}, \frac{\partial E}{\partial y_i}, \frac{\partial E}{\partial z_i}\right). \tag{3.23}$$

Instead of defining the error function and computing its derivatives [154, 78, 56, 55], one can also directly define "forces" as derivatives for gradient descent, based on the photometric consistency and mesh information [93]. In practice, the gradient of the photo-consistency term at each vertex $v_i$ is not calculated three times with respect to $x_i, y_i$ and $z_i$, but just once along the surface normal direction at $v_i$ [93, 70, 74]. The main advantage is that this reduces the expensive evaluation of the photometric consistency information by a factor of three. The surface evolution can be fully described by their normal components without the tangential ones, and the gradient of $E_p$ is usually projected along the surface normal. The gradient of the silhouette consistency term $E_s$ is also usually projected along the surface normal, too. On the contrary, the regularization term $E_r$ is used to prefer uniform vertex distribution, and the direct derivative is calculated with respect to $x_i, y_i$, and $z_i$. The mesh refinement step is iterated until convergence.

### 3.4.2 Photometric Consistency Term

We provide three examples of the photometric consistency terms [74, 93, 193].

In the work of Hernández and Schmitt [93], the gradient of the photometric consistency volume is used to define forces, because the surface of an object or a scene coincides with the region of high photometric consistency. However, the photometric consistency is known to be noisy and have many local maxima, where the gradient may have no effects for an initial surface that is far away from the solution. Therefore, the gradient vector flow [200], which is essentially the gradient field smoothed by the Laplacian operator, is used as the photometric consistency force instead. Note that for the reason mentioned in the previous section, the gradient vector flow is projected along the surface normal direction before being used as the force.

In the work of Furukawa and Ponce [74], let $\mathcal{C}(v_i)$ denote the photometric consistency score evaluated at vertex $v_i$ on its tangent plane

of the mesh. They search for the optimal offset $\hat{d}_i$ along the surface normal direction that maximizes the photometric consistency score on the tangent plane

$$\hat{d}_i = \text{argmax}_{d_i}\ \mathcal{C}(\hat{v}_i + d_i\hat{\mathbf{n}}_\mathbf{i}). \tag{3.24}$$

A non-linear optimization library [24, 82] is used to solve this problem for each vertex independently. In order to distinguish the vertex location as a variable to be optimized against the estimated location with the maximum photo-consistency, they use $v_i$ to denote the variable and $\hat{v}_i$ to denote the estimation. Similarly, $\hat{n}_i$ is used to denote the surface normal estimation based on the current mesh model. Then, the photometric consistency term at each vertex for mesh refinement can be defined to be the squared distance between $v_i$ and the location of the optimal offset, where the summation over all the vertices defines $E_p$:

$$E_p(\{v_i\}) = \sum_{v_i} \left| v_i - (\hat{v}_i + \hat{d}_i\hat{\mathbf{n}}_\mathbf{i}) \right|^2. \tag{3.25}$$

The squared distance error metric is prone to outliers, and more robust function can also be used to define the penalty term [74]. They recompute the surface normal and the offset at each vertex in every iteration.

Lastly, Hiep, Keriven, Labatut, and Pons [193] defined the term as a sum of photometric consistency scores over all the faces of a discrete triangulated mesh model:

$$E_p(\{v_i\}) = \sum_{f\in\mathbb{F}} \sum_{(I_i,I_j)\in\mathbb{V}_f} \mathcal{C}(f, I_i, I_j). \tag{3.26}$$

$\mathbb{F}$ denotes the set of faces in the mesh model, $\mathbb{V}_f$ is a set of pairs of images in which face $f$ is visible, and $\mathcal{C}(f, I_i, I_j)$ denotes the photometric consistency score evaluated on face $f$ based on the two images $I_i$ and $I_j$. They handle triangulated mesh models, and the definition of the term deduces the fact that its gradient with respect to a vertex position $v_i$ is determined by the incident faces. Different from many methods that use tangent planes or front-parallel surface approximations, the use of triangulated mesh model for photometric consistency evaluation has

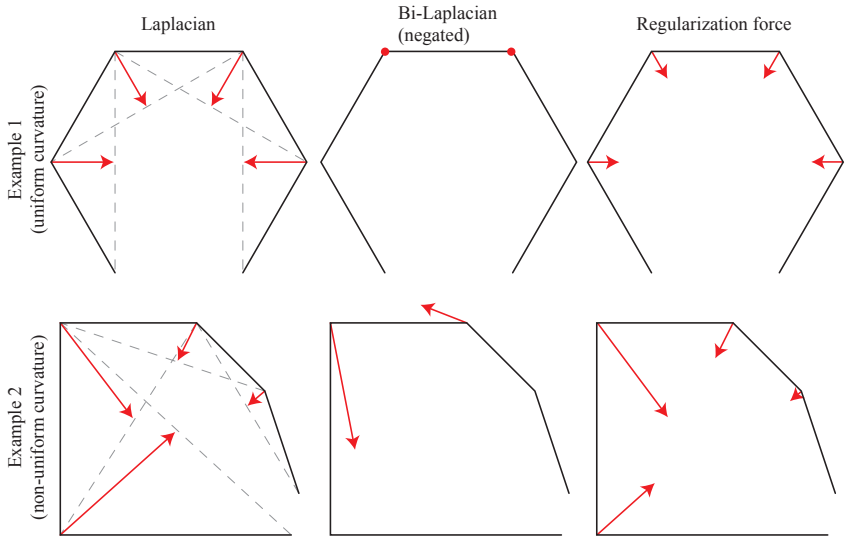



**Figure 3.32:** Regularization force in the mesh refinement framework is given by the Mesh Laplacian and Mesh Bi-Laplacian operators. The figure shows examples for 1D surfaces. Laplacian becomes zero when the surface is planar, while Bi-Laplacian becomes zero when the surface has constant or uniform curvature. Regularization forces are calculated by setting $\alpha = 0.5$ and $\beta = 0.5$ in (3.30).

several advantages. First, the core texture reprojection (i.e., morphing) operations can be efficiently and easily performed over an entire mesh by GPUs, which is as simple as rendering a texture-mapped mesh model. Second, it can handle sharp edge structure such as staircases properly, because the photometric consistency can be measured exactly on the surface, where a tangent plane does a poor job of approximating the surface there.

### 3.4.3 Regularization Term

There are little variations in the definition of regularization terms, where "Mesh Laplacian" and "Mesh Bi-Laplacian" are often used to define the regularization force [42] (See Figure 3.32). More concretely, let $\mathcal{N}_i$ denote a set of neighboring vertices of a vertex $v_i$. Then, the

Mesh Laplacian operation $\Delta(v_i)$ produces a 3D vector at $v_i$:

$$\Delta(v_i) = \frac{\sum_{v_j \in \mathcal{N}_i} v_j}{|\mathcal{N}_i|} - v_i. \tag{3.27}$$

$\Delta(v_i)$ simply computes the difference vector from $v_i$ to the center of its neighbor, and has a strong vertical force to make vertices coplanar and a strong lateral force to make vertices uniformly distributing. Note that instead of simply taking the average of the neighboring vertices, triangle areas and angles can be used to compute the weighted average for more accuracy. Bi-Laplacian operation can be similarly defined and also produces a 3D vector at each vertex

$$\Delta^2(v_i) = \frac{\sum_{v_j \in \mathcal{N}_i} \Delta(v_j)}{|\mathcal{N}_i|} - \Delta(v_i), \tag{3.28}$$

which is equivalent to applying the Mesh Laplacian operation twice. Note that for non-regular meshes where $|\mathcal{N}_i|$ is not constant $\Delta^2(v_i)$ needs to be further normalized by

$$1 + \sum_{v_j \in \mathcal{N}_i} \frac{1}{|\mathcal{N}_i||\mathcal{N}_j|} \tag{3.29}$$

to be stable (See [92] for more details). The Bi-Laplacian force is in general much weaker than the Laplacian force. The linear combination of the outputs of the two operators is usually defined to be the regularization force at each vertex $v_i$

$$\alpha\Delta(v_i) - \beta\Delta^2(v_i), \tag{3.30}$$

where $\alpha$ and $\beta$ are linear combination weights, and their typical values are around 0.5.

### 3.4.4 Silhouette Consistency Term

As the trends of 3D reconstruction shift away from object reconstruction in a lab space to scene reconstruction outside, the use of silhouette consistency becomes less popular. Silhouettes often delineate a foreground object from background, but the definition of a foreground object is not clear for scenes. Furthermore, robust automated extraction of image silhouettes is a challenging problem in itself. Nonetheless,

in certain applications, such as image-based object 3D scanning with a chromakey background, the use of silhouette consistency is potentially an effective technique [93, 171, 70, 119]. especially for an object full of intricate thin structures. There even exists an image based reconstruction system that is purely based on image silhouettes with a point light source and a rear-projection screen [201], where the visual hull [35] models yield impressive reconstructions.

The silhouette consistency simply enforces that an image silhouette of the mesh model after projection matches the input image silhouette. However, this is not an easy constraint to be enforced or an objective to be optimized. The reason is that one essentially needs to identify "contour generators" of the current model, that is, a set of points that are visible at a silhouette boundary in an image. The identification of contour generators is a binary operation, which are difficult to be optimized or enforced during mesh refinement.

In the work of Hernández and Schmitt [93], the silhouette consistency force is computed as soft constraints for each vertex $v_i$ along the surface normal direction $\mathbf{n_i}$, where the magnitude and orientation is determined by the product of two components:

$$(d_{\mathrm{S}}(v_i) \cdot \alpha(v_i))\mathbf{n_i}. \tag{3.31}$$

The first component $d_{\mathrm{S}}(v_i)$ measures the signed distance to the visual hull surface on the image domain, and tries to pull $v_i$ towards it. More concretely, let $d_{\mathrm{S}}(S_j, v_i)$ denote the signed distance between an image silhouette $S_j$ and an image projection of a vertex $v_i$ in that image, where the distance is negative if the projection is outside the silhouette, and positive otherwise. Then, $d_{\mathrm{S}}(v_i)$ is given as minimum distance over all the input images:

$$d_{\mathrm{S}}(v_i) = \min_{I_j} d_{\mathrm{S}}(S_j, v_i), \tag{3.32}$$

where $S_j$ denotes an image silhouette associated with image $I_j$ (See Figure 3.33). Intuitively, if $d_{\mathrm{S}}(S_j, v_i)$ is negative for one image (i.e., outside a silhouette), the first component becomes negative and hence the force to push the vertex inwards. If $d_{\mathrm{S}(S_j,v_i)}$ is positive for all the images (i.e., inside the silhouettes), the first component identifies the

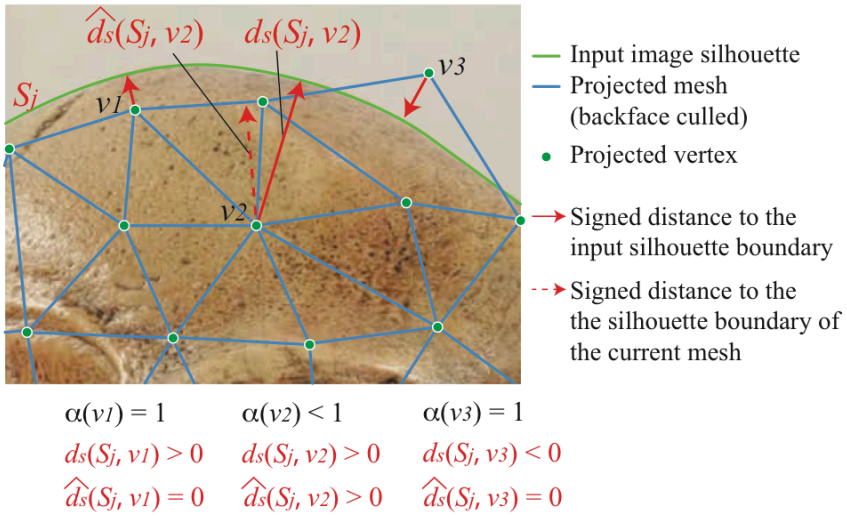



**Figure 3.33:** The magnitude of the silhouette consistency term is calculated for each vertex $v_i$ based on the two quantities. The first quantity $d_S(v_i)$ is simply the signed distance to the input image silhouette. In the figure, $v_1$ and $v_2$ are inside the silhouette, and has positive distances, while $v_3$ is outside and has a negative distance. The second quantity $\alpha(v_i)$ measures how strong the silhouette consistency force should be. If a vertex projects outside the image silhouette (e.g., $v_3$), this immediately violates the consistency and $\alpha(v_i)$ becomes 1. If a vertex projects inside the image silhouette, the quantity becomes inversely proportional to the distance $\widehat{d_S}(v_i)$ from the silhouette boundary of the current model. $v_1$ is on the boundary, and hence, $\alpha(v_1) = 1$, while $v_2$ is not and $\alpha(v_2) < 1$. As a result, the silhouette consistency term should become strong for $v_1$ (outwards) and $v_3$ (inwards), but small for $v_2$.

image where the vertex $v_i$ is closest to the input silhouette boundary, and tries to pull the vertex outwards.

However, the first term alone drives the mesh to the visual hull model, while we do not need all the vertices to be on the boundary of image silhouettes. Therefore, the second term ensures that the silhouette consistency force is active only near contour generators of the current model. In particular, let us define $\widehat{d_S}(v_i)$ to be the same as $d_S(v_i)$ except that the distance is now measured to the silhouette boundaries generated by the current mesh model as opposed to input image silhouettes. Then, $\widehat{d_S}(v_i)$ is always positive by definition and indicates how

close the vertex is to a contour generator. They define $\alpha(v_i)$ to be

$$\alpha(v_i) = \begin{cases} 1 & \text{if } d_{\mathrm{S}}(v_i) \leq 0 \\ \frac{1}{(1+\widehat{d_{\mathrm{S}}}(v_i))^n} & \text{if } d_{\mathrm{S}}(v_i) > 0 \end{cases} \tag{3.33}$$

First of all, $\alpha(v_i)$ is 1 if $v_i$ projects outside an input image silhouette, because this immediately violates the consistency and a vertex must move inwards. If $v_i$ projects inside all the image silhouettes, $\alpha(v_i)$ is 1 only at contour generators and gradually decreases as a vertex moves away from a contour generator. $n$ determines how quickly the influence of the silhouette consistency decays and is set to 2 in their work. In Figure 3.33, the silhouette consistency force for $v_3$ is strong and inwards, because $v_3$ projects outside the image silhouette and is violating the silhouette consistency. The force for $v_1$ is also strong, because $v_1$ is on the contour generator of the current mesh model, which must match the image silhouette. However, the force for $v_2$ is not strong, because $v_2$ projects inside the silhouette and is not a part of the contour generator.

### 3.4.5 Reconstruction Results

As presented in the last section, there are many choices to make when building a mesh refinement algorithm. In this section, we cover two important algorithms in the literature – the first very successful approach and the latest state-of-the-art.

Hernández and Schmitt used the visual hull model computed from input image silhouettes to initialize the mesh model, then iteratively refined it based on the sum of photometric, silhouette, and regularization forces [93] (See Figures 3.34 and 3.35). Photometric consistency volume is constructed based on the octree data structure, so that fine scale voxels are allocated at the space of high photometric consistency as in Section 3.4.2. The surface location is easily identifiable as the middle of the high gradient vector flow (GVF) regions, where the forces are diffused to also have long range effects. The bottom left of the figure shows a shaded rendering of the photo-consistency volume. The bottom middle and the bottom right figures visualize the two quantities composing the silhouette consistency force, namely $d_{\mathrm{S}(v_i)}$ and $\alpha(v_i)$, respectively. Notice that silhouette consistency force arise only near con-

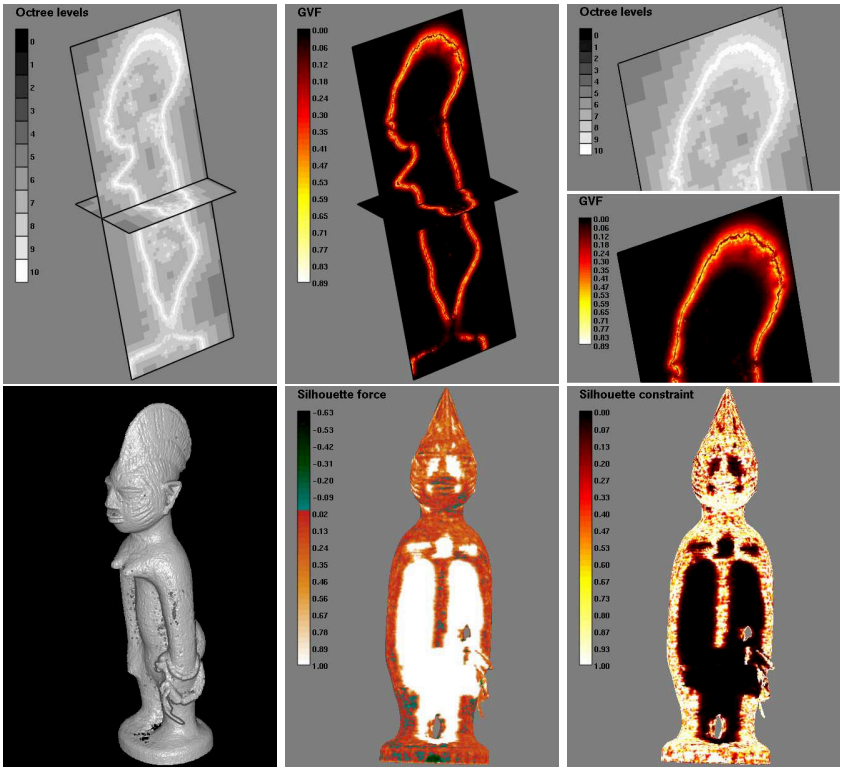



**Figure 3.34:** An MVS algorithm by Hernández and Schmitt calculates the photo-consistency volume in an octree data structure (top left), from which the gradient vector flow (GVF) [200] is used to compute the photometric consistency force (top middle). The close-ups of the octree data structure and the GVF are shown at the top right. Bottom left shows a shaded rendering of the photo-consistency volume before GVF, which is basically a 3D point cloud. Their algorithm enforces silhouette consistency in the mesh refinement process. The bottom middle figure shows the signed distance to the closest image silhouette for each vertex of a mesh ($d_{\mathrm{S}(v_i)}$), where the sign is positive for vertices inside the silhouette and negative for those outside the silhouette. Silhouette consistency should be enforced only near contour generators, that is, vertices that project near input image silhouettes, where the bottom right figure visualizes the strength of the silhouette consistency to be enforced.

**Figure 3.35:** Reconstruction results by Hernández and Schmitt [93]. From left to right, one of the 36 input images, the reconstructed mesh model, and the texture mapped mesh model for each dataset. The visual hull model is used to initialize the mesh, which is then iteratively refined.

**Figure 3.36:** Processing pipeline of an MVS system proposed by Vu, Labatut, Keriven, and Pons [193]. After reconstructing a sparse point cloud from images (top second column), a volumetric graph cuts with Delaunay tetrahedralization is used to initialize a mesh model (bottom second column), which is polished by the MVS refinement algorithm (third column). The right hand side of the figure shows the final 3D model with a mix of shaded and texture mapped rendering.

tour generators, that is, surface regions that are visible as a silhouette boundary in an image. Only the Laplacian operator is used to compute the regularization force without the bi-Laplacian operator, that is, $\beta$ is set to 0 in (3.30). An input image, the reconstructed mesh model, and the texture mapped mesh model of their algorithm are shown for the three datasets in Figure 3.35. They used a turn-table to acquire 36 images of $2008 \times 3040$ pixels of for each dataset. The numbers of the vertices in the final model are 45,843, 83,628, and 114,496, respectively, where more reconstruction results can be found in the paper. In addition to the intricate details, the method succeeded in reconstructing thin structures, where silhouette consistency plays an important role. This work was published in 2004, and really the first to prove the potential of multi-view stereo algorithms. Their total running time can reach 16 hours for a dataset, but this is a running time in 2004, where a Pentium4 1.4 GHz machine was used. The system should run much faster with state-of-the-art multi-core processors potentially with a use of GPUs for further speedup.

One of the most successful state-of-the-art MVS systems was proposed by Vu, Labatut, Keriven, and Pons in 2009 [193], where the mesh refinement algorithm is used in the last step. Their overall pipeline is illustrated in Figure 3.36, where their reconstruction results are shown

in Figure 3.37. The photometric consistency term is defined over the polygonal mesh model as in (3.26). Their system can handle arbitrary datasets including outdoor scenes, and does not incorporate silhouette consistency information. The regularization term only exploits the bi-Laplacian operation, that is, $\alpha$ is set to 0 in (3.30). The input mesh model to the final refinement step is generated by reconstructing a sparse point cloud from images [129], then using a volumetric graph-cuts technique with Delaunay tetrahedralization (Section 3.3.2). As shown in the figures, their system is capable of reconstructing very complicated shapes but with intricate details. It is also efficient, for example, it takes only 45 minutes for an entire system to run and produce the bottom result in Figure 3.37. The top of Figure 3.37 shows the robustness of the system, in particular, at trees, which are not suitable for MVS reconstruction due to their complicated and changing shapes.

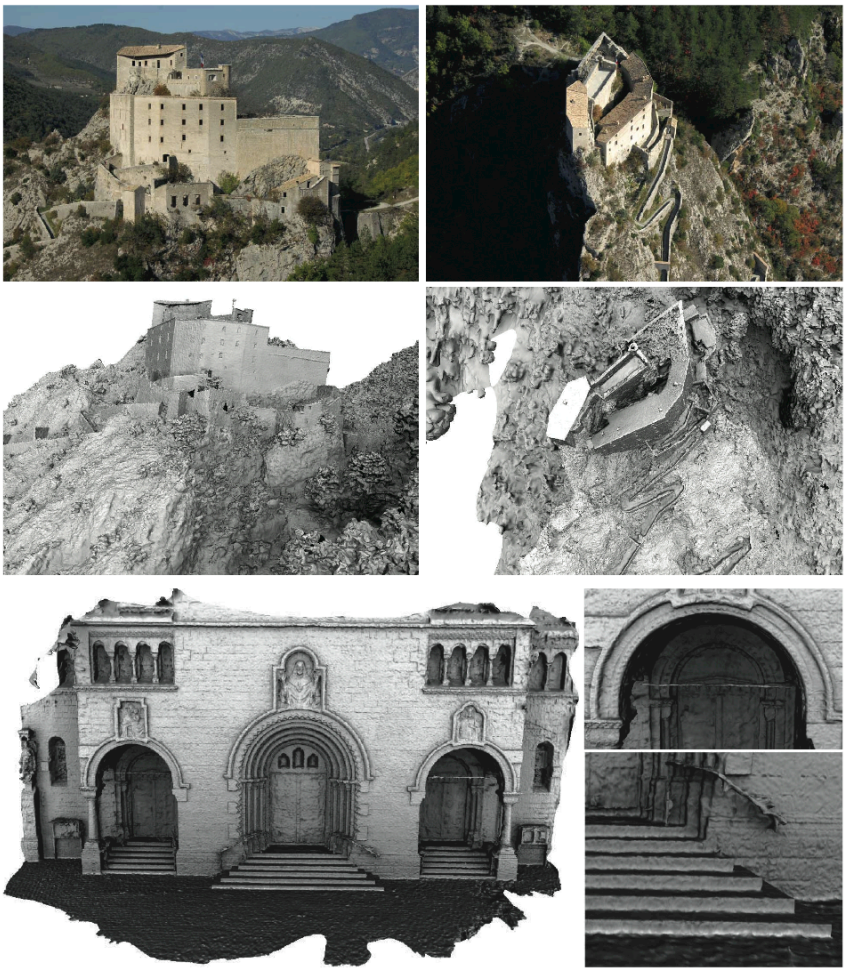


**Figure 3.37:** Top: A reconstruction result by an MVS system by Vu, Labatut, Keriven, and Pons [193]. A helicopter was used to acquire 109 images (3.1 Mpixels). For each column, the top row shows the reconstructed mesh model and the bottom row shows an input image with a similar viewing angle. Bottom: Another reconstruction result by an MVS system by Vu, Labatut,con Keriven and Pons [193]. 25 images of high resolution images ($2048 \times 3072$ pixels) are the input to the system. The dataset (*Herz-Jesu-P25*) is from a MVS benchmark project by Christoph Strecha [176].

# 4

# Multi-view Stereo and Structure Priors

MVS works very well for textured and non-Lambertian surfaces in practice as long as they contain some diffuse reflectance component. However, our world is abundant with textureless and highly non-Lambertian objects that make MVS algorithms fail. Indoor environments are good examples of scenes that contain many such objects, for instance, white walls, transparent glasses, and shiny electric appliances (See Figure 4.1). Fortunately, indoor environments exhibit strong structural regularities such as planarity and orthogonality, which can be exploited to improve reconstruction quality. A recent trend in MVS research is to mix the core MVS principle with some form of structure priors, which essentially acts as a smart interpolator to fill in reconstruction holes with proper geometry or a smart smoother to suppress noise, where photometric consistency is unreliable.

Back in the 90s, Birchfield and Tomasi proposed a reconstruction algorithm, which produces a piecewise affine disparity maps by exploiting the fact that scenes often consist of piecewise planar surfaces [38]. They iterate between the image segmentation into piecewise planar segments and the estimation of disparity affine parameters per segment in a way similar to Expectation Maximization algorithm (See Figure 4.2).

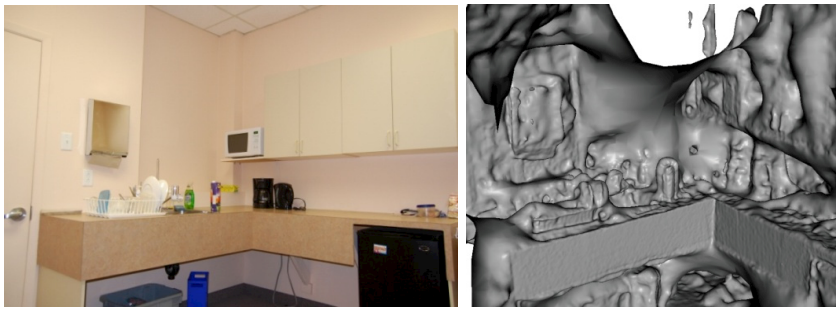


**Figure 4.1:** Standard MVS algorithms fail in reconstructing even a simple scene like this, which is full of homogeneous surfaces. The figure shows the reconstruction result of PMVS software by Furukawa and Ponce [74] followed by Poisson Surface Reconstruction software by Kazhdan [115].

After the work of Birchfield and Tomasi, MVS research rather focused on the reconstruction of high fidelity 3D models by exploring effective photo-consistency evaluation schemes without relying much on priors. After the success of MVS algorithms on near Lambertian textured surfaces (Chapter 3), MVS researchers revisited the use of structure priors. Many such algorithms are based on the 2D MRF formulation as in the depthmap estimation in Section 3.1, while resorting to powerful optimization machineries such as graph-cuts or belief propagation techniques.

In a standard MRF formulation for depthmap estimation, the cost function is a combination of unary terms and pairwise interaction terms, where labels encode depth values. We repeat Equation 3.2 describing the MRF formulation in the following for easy reference.

$$E(\mathbf{k}) = \sum_{p} \Phi(k_p) + \sum_{(p,q)\in\mathcal{N}} \Psi(k_p, k_q). \tag{4.1}$$

$\mathbf{k} = \{k_p\}$ denotes a set of per-pixel labels, which is the variable in this cost function. The first term is the unary term summed over all the pixels, and the second term is the pairwise interaction term summed over all pairs of adjacent pixels $\mathcal{N}$. A convention is to encode data information, that is, photo-consistency information into the unary term, while regularization prior is encoded into the pairwise term. The pairwise

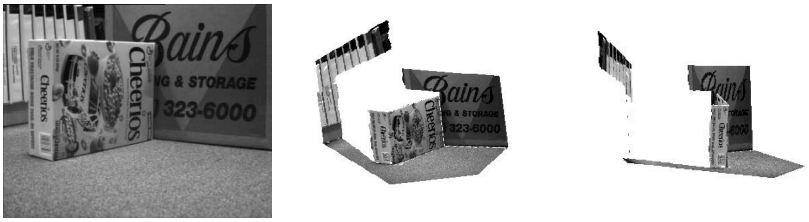



**Figure 4.2:** Birchfield and Tomasi proposed a piecewise affine disparity estimation algorithm back in the 90s. The left is one of the input images, while the middle and the right show the reconstructed 3D model rendered from two different viewpoints. (Figure courtesy of Birchfield et al.)

term usually penalizes the distance of labels, in our case, the amount of discrepancy of depth values at adjacent pixels, which enforces front-parallel surfaces, because the pairwise term becomes zero when all the pixels have the same depth labels.

As described in Section 3.1.5, the second order smoothness prior by Woodford, Torr, Reid, and Fitzgibbon is a natural extension to this, which favors piecewise planar surfaces [196]. However, the introduction of higher order terms makes the energy non-submodular, which resulted in much more complex optimization steps. Olsson, Ulen, and Boykov changed the meaning of labels from depth values to depth values plus surface normal orientations [148] to also enforce piecewise planarity. This approach does not require higher order terms, but does not guarantee submodularity and need complicated optimization steps, too.

In the following sections, we take a close look at MVS algorithms that change the definition of labels to enforce high-level structure priors such as planarity, while still keeping the submodularity to enable the simple application of powerful graph-cuts technique.

## 4.1 Departure from Depthmap to Planemap

The enforcement of piecewise planarity may sound easy but is a challenging one. In order for the graph-cuts algorithm to be applicable, the cost function must be submodular, which prefers different pixels having the same label. However, a planar surface of arbitrary orientation

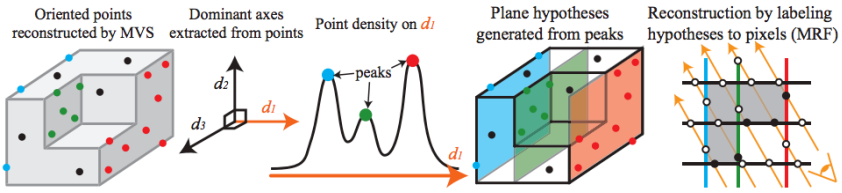



**Figure 4.3:** Manhattan stereo algorithm [67]. Given a set of 3D points reconstructed by a standard MVS algorithm, they estimate dominant axes and extract hypothesis planes along each axis by finding point density peaks. Finally, the MRF formulation is used to assign a hypothesis plane to each pixel.

is realized by pixels with slowly changing but different depth values. Therefore, one idea is to change the definition of labels from depth values to something different so that the favorable configurations (e.g., piecewise planar structure) can be realized by assigning the same label to different pixels.

Furukawa, Curless, Seitz, and Szeliski proposed *Manhattan-world Stereo* algorithm, where labels indicate planes that are extracted from a standard MVS reconstruction [67]. As the name of the algorithm indicates, three dominant orthogonal directions are first computed and only planes along the dominant directions are extracted to define labels. Their processing pipeline is shown in Figure 4.3. Sinha, Steedly, and Szeliski extended the framework and allow planes of arbitrary orientations that are extracted from both reconstructed 3D points and 3D line segments [170]. The MRF formulation is used to estimate a plane ID per pixel in both methods, which is called a planemap as opposed to a depthmap. The definitions of unary and pairwise terms vary across methods as in the case of standard MRF algorithms. Next section explains the MRF formulation of the Manhattan-world Stereo algorithm [67] [1].

[1] The details of other steps in the pipeline such as the plane extraction are refereed to the paper [67].

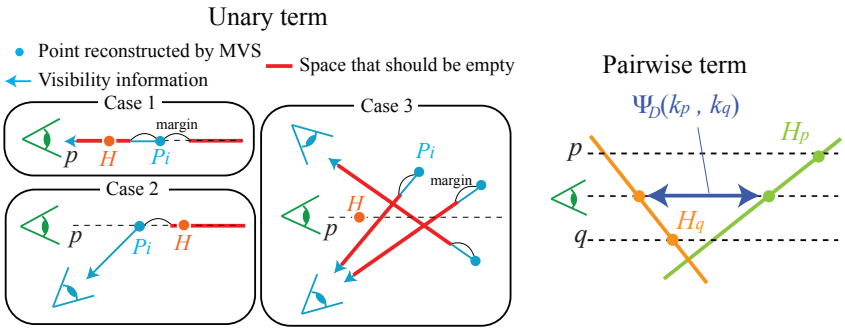



**Figure 4.4:** Data term measures visibility conflicts between a plane hypothesis at a pixel and all the reconstructed points $\{P_i\}$. There are three different cases in which the visibility conflict occurs. The smoothness term in this figure measures the penalty of assigning a hypothesis $H^n$ to pixel $q$, and a hypothesis $H^m$ to pixel $p$.

### MRF Formulation in Manhattan-world Stereo

The input to the algorithm is a set of 3D points $\{P_i\}$ reconstructed by an MVS algorithm, where each point is associated with a set of visible images that are used to reconstruct the point. It is assumed that a set of 3D hypothesis planes are extracted. Given a *reference image*, the problem is to reconstruct a piecewise planar depthmap as a combination of hypothesis planes, in other words, reconstruct a *planemap* by assigning one of the planes to each pixel in the reference image.

### Data Term

The data term $\Phi(k_p)$ measures visibility inconsistency between a plane hypothesis at a pixel and all of the input 3D points (See Figure 4.4). Consider the penalty of assigning a plane to a pixel. Let $H$ denote the 3D point that is to be reconstructed for pixel $p$, when a plane hypothesis is assigned. In other words, $H$ is the intersection between a viewing ray passing through $p$ and the hypothesis plane. The hypothesis plane is considered to be inconsistent with an MVS point $P_i$ under any one of the three following cases, which are also illustrated in Figure 4.4. In the figure, a reference image, for which a planemap is reconstructed, is colored in green.

**Case 1.** If $P_i$ is visible in the reference image (green) and projects at the pixel $p$, the hypothesized point $H$ should not be in front of $P_i$ (since it would occlude it) and should not be behind $P_i$ (since it would be occluded). Therefore, a penalty is imposed on a plane hypothesis, if the corresponding point $H$ is not at the same depth as $P_i$, while allowing some margin. They perform this check for every 3D point $P_i$ that is visible in the reference image and projects near the pixel.

**Case 2.** If $P_i$ is *not* visible in the reference image but still projects at pixel $p$, $H$ should not be behind $P_i$, since it would be occluded. A penalty is imposed on such plane hypotheses, and this check is again performed on all the 3D points.

**Case 3.** Given a 3D point $P_i$, for any of its visible images, the space in front of $P_i$ on the line of sight to the camera center should be empty, which is often referred to as a free-space constraint. A penalty is imposed to a plane hypothesis, whose corresponding point $H$ resides in such free-space. In practice, both $P_i$ and $H$ are projected to the visible image (blue camera at the bottom in the figure), and the depth values are compared to check conflicts.

Now, given a pixel and a hypothesis plane, the penalty of the data term can be defined based on the number of above three types of conflicts. More concretely, let $C(P_i)$ denote the photo-consistency score, in particular NCC measure of point $P_i$, whose visibility information is in conflict with a hypothesis plane assignment $k_p$. The data term is defined as

$$\Phi(k_p) = \min(0.5, \sum_i \max(0, \mathcal{C}(P_i) - 0.7)), \tag{4.2}$$

where the summation is over the point set with visibility conflicts. Note that the penalty is accumulated from points, whose NCC scores are high enough, that is, more than 0.7. A simple truncation of value 0.5 is used for robustness.

### Smoothness term

The smoothness term $\Psi(k_p, k_q)$ enforces spatial consistency and is 0, if the same plane hypothesis is assigned, that is, $h_p = h_q$. Otherwise, they seek a smoothness function that penalizes the amount of depth

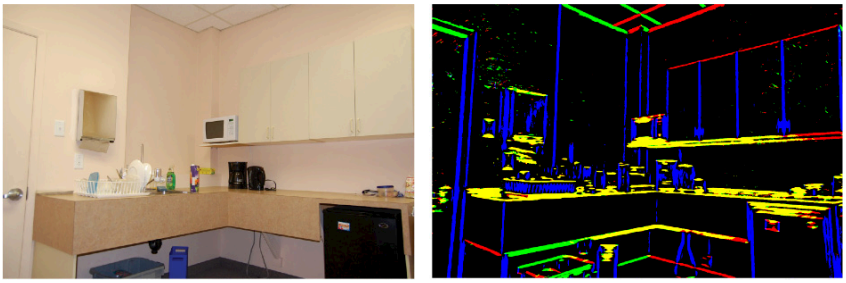


**Figure 4.5:** Input image and extracted dominant lines, used as cues to control the smoothness penalty. The red, green and blue components in the right figure shows the results of edge detection along the three dominant directions, respectively. Note that yellow indicates ambiguity between the red and green directions.

discrepancy. The penalty is down-weighted in the presence of image edges, where depth discontinuity is reasonable. The term is defined by the product of two factors:

$$\Psi(k_p, k_q) = \Psi_D(k_p, k_q)\Psi_E(k_p, k_q) \tag{4.3}$$

The first factor $\Psi_D(k_p, k_q)$ penalizes the amount of depth discrepancy between neighboring pixels, which is illustrated by the blue arrow in the right of Figure 4.4, where the green hypothesis plane is assigned to pixel $p$ and the orange hypothesis plane is assigned to pixel $q$. $H_p$ and $H_q$ are their reconstructed points in this configuration. More specifically, the factor is defined to be the distance between the two planes along the line of sight at the middle point of $p$ and $q$.

The second factor $\Psi_E(k_p, k_q)$ controls the amount of smoothness penalty based on the presence of image intensity edges: the penalty decreases at image intensity edges, which may correspond to a depth discontinuity (See Figure 4.5). Under the assumption of a piecewise planar scene aligned with the orthogonal axes, a depth discontinuity appears as a crease line in an image that passes though one of the three vanishing points, corresponding to the three dominant directions. Therefore, an edge detection filter, which is designed to respond only to dominant lines (See the paper for more details [67]), is used to compute

an edge detected image. $\Psi_E(k_p, k_q)$ is simply defined as follows:

$$\Psi_E(k_p, k_q) = \begin{cases} 0.01 & \text{If } p \text{ or } q \text{ is an edge pixel} \\ 1 & \text{Otherwise} \end{cases} \tag{4.4}$$

Note that the direction(s) of the corresponding dominant line(s) is known for each edge pixel, and it may seem natural to use a more sophisticated scheme to define when $\Psi_E$ should become small. For example, it makes sense to additionally require that the direction of the dominant line at the edge pixel must be perpendicular to the normals of the planes assigned to pixels $p$ and $q$. However, this makes the pairwise term non-submodular, while the above simple solution keeps the optimization tractable and works well in practice.

Given the MRF formulation, the $\alpha$-expansion algorithm is used to minimize the energy and estimate a planemap [44, 45, 121].

### Reconstruction Results

Figure 4.6 shows the reconstruction results of Manhattan-world Stereo algorithm [67] on both indoor and outdoor architectural scenes, which are full of planar surfaces that meet at right angles. Notice the presence of abundant textureless or highly non-Lambertian surfaces such as white walls or shiny metallic surfaces, which are reconstructed successfully. There is also a fair number of curved surfaces, which unfortunately breaks the Manhattan-world assumption, but their algorithm does its best to approximate its shape by piecewise planar surfaces.

Figure 4.7 compares results against a state-of-the-art MVS pipeline without structure priors (PMVS [74] and Poisson Surface Reconstruction software [115, 41, 116]). Manhattan-world Stereo algorithm is, in essence, a depthmap algorithm, and reconstructs a geometry visible only in a single image as opposed to the competing pipeline that uses all the input images to reconstruct an entire scene model. Nonetheless, the competing approach often makes gross errors due to the lack of reliable textures, which are cleanly reconstructed with the use of structure priors.

Reconstruction results of a piecewise planar stereo algorithm by Sinha, Steedly, and Szeliski [170] are presented in Figure 4.8 for various

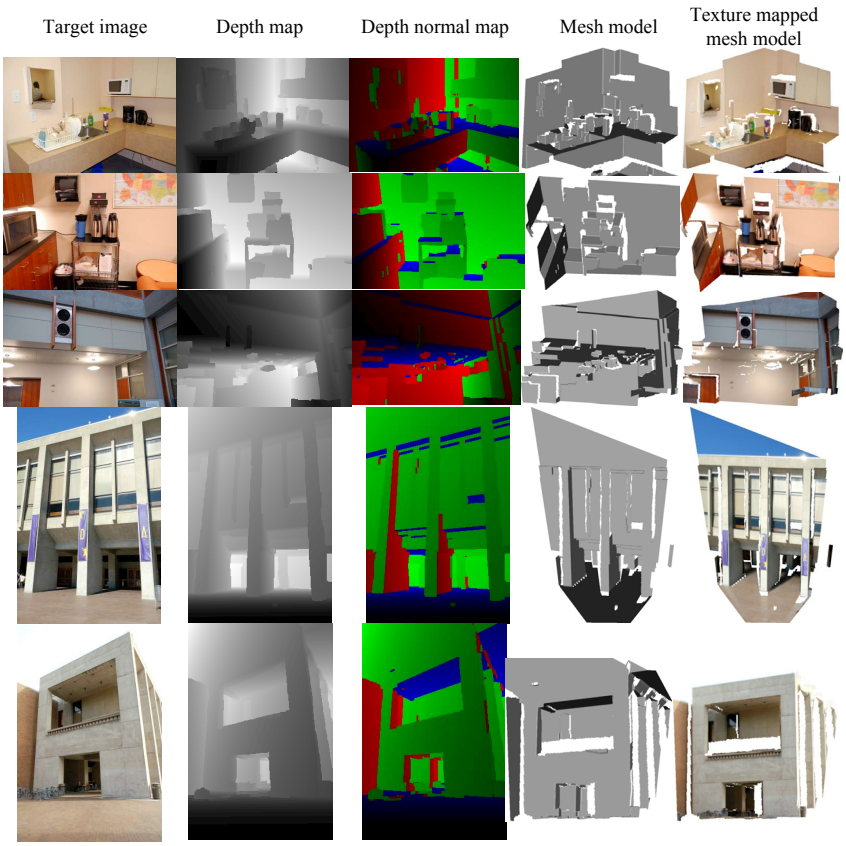



**Figure 4.6:** Reconstruction results of Manhattan-world Stereo algorithm [67]. In the depth normal map view, the three possible surface normals are indicated by the red, green, and blue colors, respectively.

outdoor architectural scenes. Note that the algorithm is able to capture slanted surfaces properly, which is not possible in the Manhattan-world Stereo algorithm.

## 4.2 Departure from Planes to Geometric Primitives

Zebedin, Bauer, Karner, and Bischof developed a system [209] that generates digital elevation models of cities from aerial photographs (See

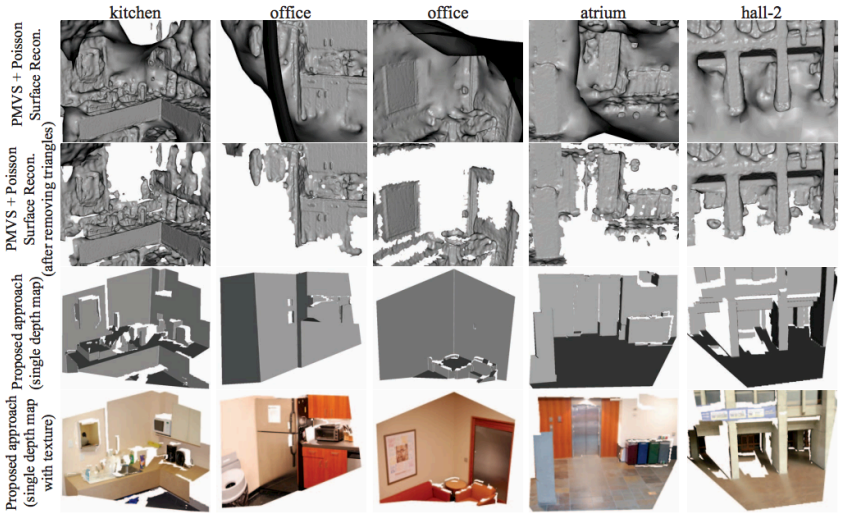



**Figure 4.7:** Comparing Manhattan-world Stereo algorithm [67] with a state-of-the-art MVS pipeline, which is PMVS [74] followed by Poisson Surface Reconstruction software [115].

Figure 4.9) which goes beyond piecewise planar and can handle curved surfaces, in particular, surfaces of revolution in addition to planes of arbitrary orientations.

They estimate an elevation model or a height field representation of cities, which is formulated as a 2D MRF problem in a top down view. Their system is optimized and tested for aerial photographs of cities, and yields very impressive 3D building models (See Figure 4.10). The formulation is very similar to the piecewise planar stereo algorithms in the previous section, where an MRF is used to assign a primitive ID, but with two key differences. The first difference is that the domain is discretized by a grid of much larger rectangular cells (left of Figure 4.9) based on the 3D line segments reconstructed by a multi-view line matching algorithm [32]. A primitive ID is assigned to each cell, which has additional data-aware regularization effects. The second difference is the handling of surfaces of revolution as geometric primitives, which may not be effective for arbitrary scenes, but often arise in out-

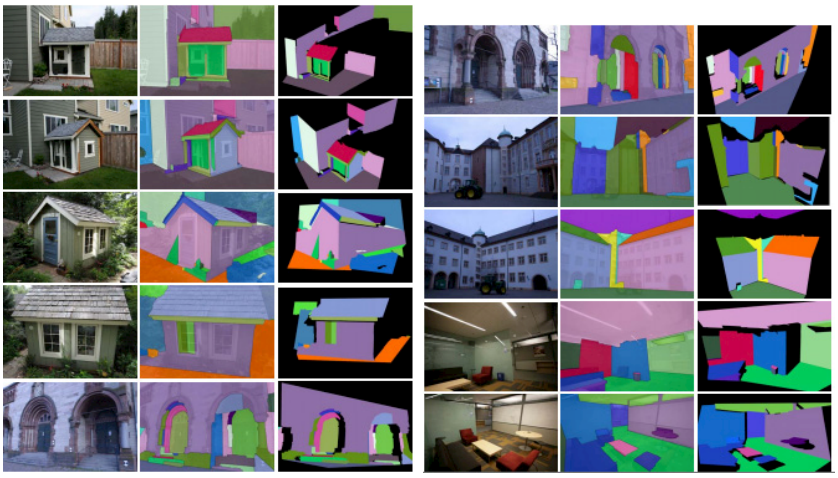


**Figure 4.8:** Reconstruction results by a piecewise planar depthmap algorithm by Sinha, Steedly, and Szeliski [170]. Estimated plane IDs are illustrated by different colors in each of the middle columns, which are also rendered from a different viewpoint in each right column. (Figure courtesy of Sinha et al.)

door buildings such as domes at the top. More reconstruction results are shown in Figure 4.10.

## 4.3 Image Classification for Structure Priors

Structure priors through planes or surfaces of revolution are effective in architectural scenes, but pose problems for non-architectural structures such as bushes, trees, and grass, which break these assumptions but are often present in urban environments.

Gallup, Frahm, and Pollefeys employed image classification techniques to label an image into planar and non-planar regions, where structure priors (i.e., piecewise planarity) are enforced only for regions with planar classification [77]. Their algorithm has several other novel factors such as a more sophisticated way of extracting primitive candidates (planes in this case), and a framework to enforce multiple view consistency in labeling structure IDs to multiple images. However, we focus here on their use of image classification techniques and the handling of non-planar structures (See Figure 4.11).

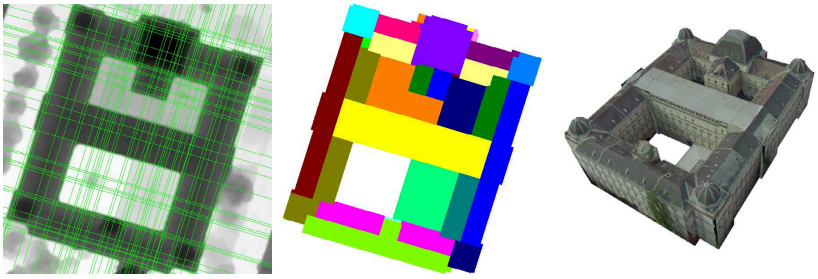


**Figure 4.9:** Left: Reconstructed 3D lines are used to segment the top down view of a scene. (Figure courtesy of Zebedin et al.)

They train a classifier based on the color and texture features to distinguish planar and non-planar surfaces, where both a simple k-nearest neighbor classifier and SVM classifier were tested and produced similar results. More specifically, each image is segmented into a grid of $16 \times 16$ pixel cells, and the following feature vector is computed in each cell. The color component of the feature vector consists of the mean red, green, and blue (RGB color space), mean hue, saturation, value (HSV color space), and the hue histogram with 5 bins. The texture component of the feature vector consists of the statistics of the edge orientation histogram, in particular, entropy, maximum value, and number of modes [125]. An intuition behind the texture cue is that man-made objects tend to have only a few consistent edge orientations. Authors pointed out that the use of superpixels [59, 156] instead of a regular square grid may make more sense, and they indeed experimented with such a superpixel segmentation algorithm, but preferred the regular grid in the end. The main observation was that the photometric consistency and smoothness penalties often provide enough information to identify accurate object boundaries, while the grid ensures the regular size and density of cells, which are preferred. They hand-labeled approximately five thousand segments in five images as planar or non-planar, then trained a classifier. A classifier outputs a probability of a cell to be a planar class, which is used in the MRF formulation to modify the data term as follows.

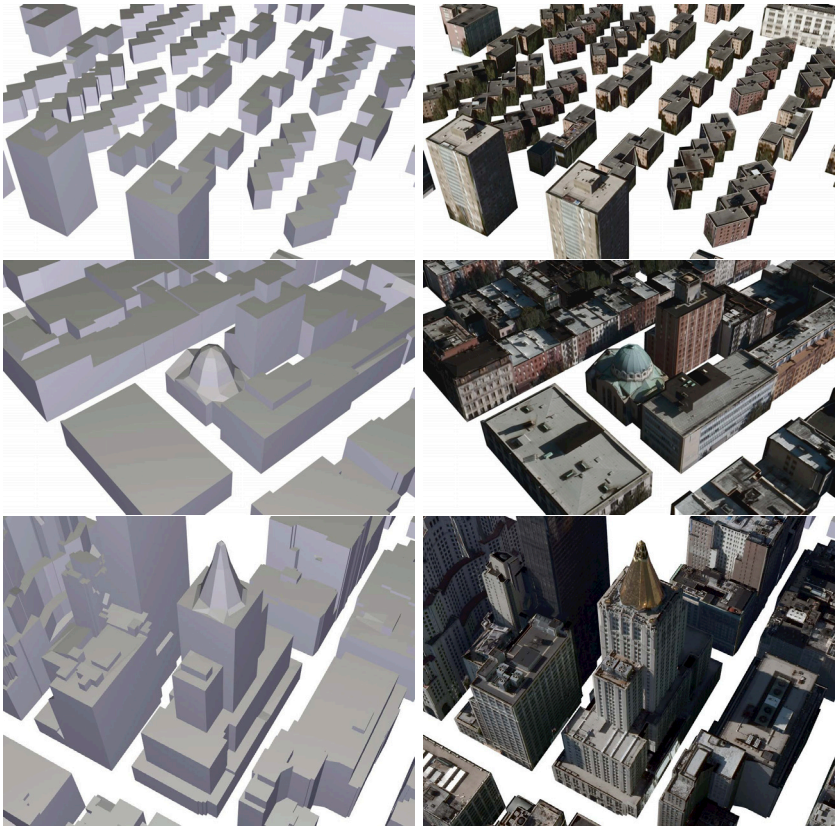


**Figure 4.10:** Extracted geometric primitive IDs are assigned to each 2D cell in a top down view of a city, yielding impressive 3D building models [209]. (Figure courtesy of Zebedin et al.)

In their MRF formulation, the set of labels is the union of plane labels, a non-plane label, and a discard label. A discard label indicates the lack of confidence and corresponds to a reconstruction hole. Plane labels and a non-plane label are obtained by using a depthmap generated by a standard MVS algorithm (i.e., a *raw* depthmap). More concretely, first, plane labels are obtained from raw depthmaps associated with all the input images by using a RANSAC algorithm [61] that fits planes to the raw depthmap. Second, a non-plane label is created to

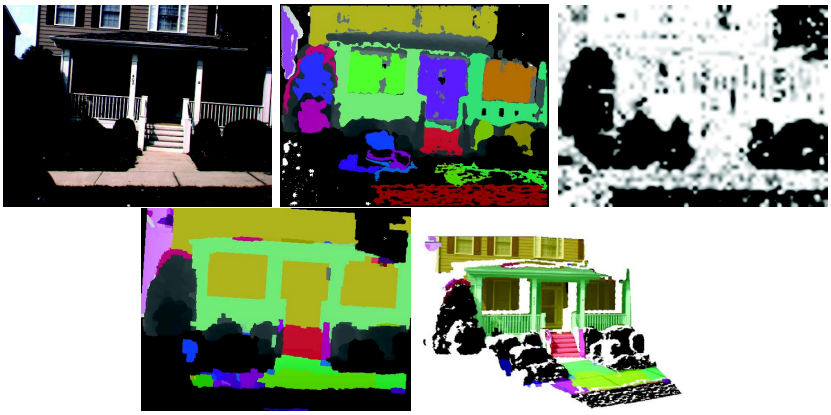


**Figure 4.11:** Gallup, Frahm, and Pollefeys proposed a depthmap reconstruction algorithm, which handles both piecewise planar architectural structures and non-planar non-architectural structures (e.g., trees, brush and grass). An image classification technique is used to identify regions where a structure prior, in particular, planarity is strongly enforced. Top from left to right: A sample input image, plane hypotheses extracted by RANSAC, and the probability of a planar structure. Bottom from left to right: Final estimated labels and the rendering of the reconstructed depthmap from a different viewpoint. (Figure courtesy of Gallup et al.)

simply indicate a non-planar structure that has the same depth value as the raw depthmap. The algorithm employs several techniques beyond the standard RANSAC to improve the quality of planes, mostly by exploiting the fact that the input samples (depth values at pixels) have a spatial structure, that is, laid out on a 2D grid as an image. They also merge planes extracted from different images to obtain a single set of planes for all the input images. Details are referred to the paper [77].

The image classification is used to change the data term for these labels, where $k_p$ is the label to be assigned to pixel $p$: [2]

$$\Phi(k_p) = \begin{cases} \mathcal{C}(k_p) + (1-a) & \text{if } k_p \text{ is a plane label} \\ \mathcal{C}(k_p) + a + \eta_{non\_plane} & \text{if } k_p \text{ is a non-plane label} \\ \eta_{discard} & \text{if } k_p \text{ is a discard label} \end{cases} \tag{4.5}$$

[2]The exact definition of the data term $\Phi(k_p)$ has an upper bound as a hinge loss, but is omitted here for simplicity.

$\mathcal{C}(k_p)$ encodes the photometric consistency information, and $a$ denotes the estimated probability of a pixel to be a planar structure by the classification algorithm. $\eta_{non\_plane}$ and $\eta_{discard}$ are the bias cost terms for the non-plane and discard labels.

The pairwise term $\Psi(k_p, k_q)$ is defined to prefer consistent labeling with some special handling of discard labels. [3]

$$\Psi(k_p, k_q) = \begin{cases} 0 & \text{if } k_p = k_q \\ \Psi_D(k_p, k_q)\Psi_E(k_p, k_q) & \text{if } k_p \text{ and } k_q \text{ are not discard labels} \\ \eta'_{discard}\Psi_E(k_p, k_q) & \text{otherwise} \end{cases} \tag{4.6}$$

First, the penalty is zero when the two labels are the same as in many MRF formulations. Second, when both are not discard labels, that is, when both are either a plane or non-plane label, then, the cost is simply defined to minimize the amount of depth discrepancy, which is rescaled based on the presence of image edges. More concretely, the cost is the product of two factors $\Psi_D(k_p, k_q)$ and $\Psi_E(k_p, k_q)$, which denote the amount of depth discrepancy between the pixels, and the inverse of the presence of image intensity edges, respectively. This formulation is the same as in the Manhattan World Stereo algorithm (4.3), although the exact definitions of the factors are different. Third, in the remaining cases, where at least one label is a discard label, a large bias term $\eta'_{discard}$ replaces $\Psi_D(k_p, k_q)$, which is equivalent to saying that the amount of depth discrepancy for a discard label is considered to be very large.

Their pairwise term satisfies the submodularity conditions and the same $\alpha$-expansion technique is used to solve the problem [44, 45, 121]. Figure 4.11 shows some intermediate results of the algorithm for a typical residential area that has a mix of architectural structure and vegetation. Their algorithm successfully classifies bushes and trees as non-planar structure and retains their shapes. Figure 4.12 compares the method against a standard MVS algorithm, which clearly shows the advantage of the structure prior exploiting the planarity of scenes, while keeping the proper structure for non-planar surfaces. More re-

[3] The algorithm also handles a plane at infinity in a special way, which is omitted here for simplicity.

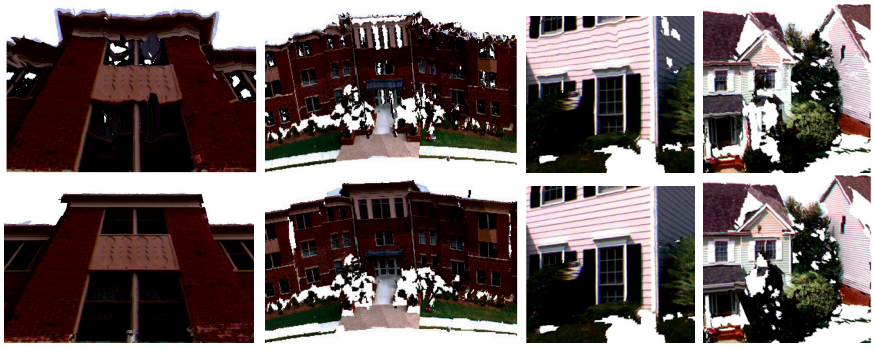


**Figure 4.12:** Comparative evaluations between (Top) a standard MVS algorithm and (Bottom) a planar and non-planar depthmap algorithm by Gallup, Frahm, and Pollefeys [77]. (Figure courtesy of Gallup et al.)

construction examples are shown in Figure 4.13 in a form of texture mapped depthmaps as well as highlighted planes, again for typical urban scenes with a mix of buildings and vegetation.

**Figure 4.13:** Reconstruction results of a depthmap estimation algorithm by Gallup, Frahm, and Pollefeys [77] for two scenes. For each scene, the top shows the final texture mapped depthmaps, and the bottom highlights the estimated plane IDs. (Figure courtesy of Gallup et al.)

# 5

# Software, Best Practices, and Successful Applications

Numerous algorithms have been proposed in MVS, many of which yield high reconstruction accuracy, even comparable to that of laser range sensors [165]. While image acquisition is often a non-trivial step for successful MVS reconstructions, the community has consolidated know-hows and best practices as the field matures. Combined with a suite of effective software, the field has witnessed the development of many successful image-based dense reconstruction systems. This chapter introduces such successful software, best practices, and successful application examples.

## 5.1 Software

"PMVS" by Furukawa and Ponce [74] is probably the first successful open-source MVS software, which has been extensively used by non-experts such as artists and civil engineers, as well as real production companies such as Industrial Light & Magic [9], Weta Digital [20], and Google [8]. Together with "Bundler" by Noah Snavely [173], which solves a Structure-from-Motion (SfM) problem to estimate camera parameters from an image set, and Poisson Surface Reconstruction soft-

ware by Michael Misha Kazhdan [115, 41, 116], which turns an oriented point cloud into a mesh model, people have built an end-to-end fully automated 3D reconstruction pipeline.

More recently, an SfM software "VisualSFM" by Changchang Wu [198], which is a GUI application for SfM, and an MVS software "CMPMVS" by Michal Jancosek and Tomas Pajdla [109] were developed as compelling alternatives.

The Multi-View Environment (MVE) is an implementation of a complete end-to-end pipeline for image-based geometry reconstruction, developed at TU-Darmstadt [12]. It features SfM, MVS, and Surface Reconstruction. The individual steps of the pipeline are available as command line applications, but most features are also available from a user interface.

Open Multiple View Geometry (OpenMVG) [13] is an open-source library for computer-vision scientists, especially targeted to the multiple view geometry community. The library is designed to provide an easy access to the classical problems in multiple view geometry, for example, feature detection/description/matching, feature tracking, and bundle adjustment. The library also includes the two complete SfM pipelines. OpenMVG provides customizable tools, where the community has built data pipeline based on OpenMVG for other multiview geometry software such as PMVS, CMPMVS, and MVE.

## 5.2 Best practices for Image Acquisition

Image acquisition is the first critical step for successful MVS. Here, we summarize best practices and know-hows for successful image acquisition.

**Accuracy of the camera models:** MVS techniques are highly dependent on the accuracy of the camera parameters. Typical reprojection error RMSE should be sub-pixel, ideally smaller than 0.5 pixels. In case reprojection error is large, one possibility is to shrink the input images and modify the corresponding camera parameters, which will reduce the reprojection error proportionally to the shrinkage ratio.

**Image resolution:** One of the reasons that MVS has been so successful is the improvements in image sensors. Consumer-grade cameras

produce high quality and high resolution images. High resolution images bring up details that can be used to uniquely identify a pixel from its neighbors, thus improving the correspondence cue used by MVS algorithms to find similar pixels across multiple images. Note however that by resolution we do not mean just having lots of pixels, the quality of camera lens also matters. Having a very high-res image captured with a poor quality lens will not improve results, and it may actually make them worse, e.g. due to worse results at the SfM stage.

**Image overlap:** For MVS to work correctly, multiple images need to see the same piece of geometry from multiple view points. Although the bare minimum is only two images, at least three images are typically required to observe each 3D point for robustness.

**Baseline:** MVS uses the principle of triangulation to reconstruct 3D geometry from pixel matches across multiple images. This means that the accuracy of the geometry has a direct dependency on the baseline of the triangulation. The more baseline, the more accurate reconstruction. On the other hand, a larger baseline makes it harder to match pixels across images for two reasons: 1) there would be more occlusions, and 2) the appearance of the same pixel would vary more across images, making it harder to match. As a result, there is a compromise between accuracy, robustness, and density. Typical optimal baseline, or viewing angle differences from a 3D point to input camera locations are in the range of 5-15 degrees.

**Number of images:** MVS algorithms are extremely good at exploiting a large number of images together with the view selection, as long as the following two conditions are met: 1) SfM reprojection error is small; and 2) The image set contains high quality images at minimal baselines. In other words, the more images are available, the better MVS works. The Middlebury MVS evaluation results [165] clearly show a relationship between the number of images and the quality. However, if one has to choose between image resolution and number of images, there is no easy decision. MVS algorithms reconstruct more details from higher resolution images, as MVS suffers little from ambiguous matches. On the other hand, high resolution images become an issue for SfM, as the so-called *ratio-test* [136] would reject many feature matches. Therefore,

if good camera calibration is available, in general one should choose image resolution over number of images.

**Image quality:** MVS algorithms compare intensity values across multiple images. Although there exist many algorithms that are robust to illumination variations across the images, the more stable it can be, the better. For example, flash changes shading and shadowing effects in every image, and should not be used for weakly textured surfaces. Similarly, although blur and out-of-focus effects may be artistically pleasant, it degrades much of the detail in MVS reconstructions. Ideally all the images used in MVS should be all-in-focus. This can be achieved by using small apertures and large exposures, within the limits of a particular application.

## 5.3 Successful Applications

The success of MVS technologies has changed the entire shape of the 3D reconstruction industry by replacing laser range sensors, which are costly to build and maintain, with image based solutions. As a variety of successful MVS algorithms presented high quality results in research, industrial applications followed with successful implementations of MVS for real products.

The visual effects industry, for example, relies more and more on image-based dense 3D reconstruction, in particular, for close range 3D scanning such as objects and human faces [37] (See Figure 5.1). For small scale object or scene reconstruction, there exist commercial 3D reconstruction systems by major software companies such as "123D Catch" by Autodesk [31], which can turn a collection of photographs into a 3D model. With the advancement of mobile devices, the software can be fully utilized just by a cell phone or a tablet, and its GUI interface lets anybody enjoy image-based 3D reconstruction capabilities.

Digital mapping is probably the biggest industry, in which MVS plays a crucial role, where their ultimate goal is to digitally map the entire world in 3D (See Figure 5.2). Apple Maps "Flyover" feature provides views of high quality texture mapped city building models. Google Maps "Earth View" also provides stunning 3D views of cities. Microsoft

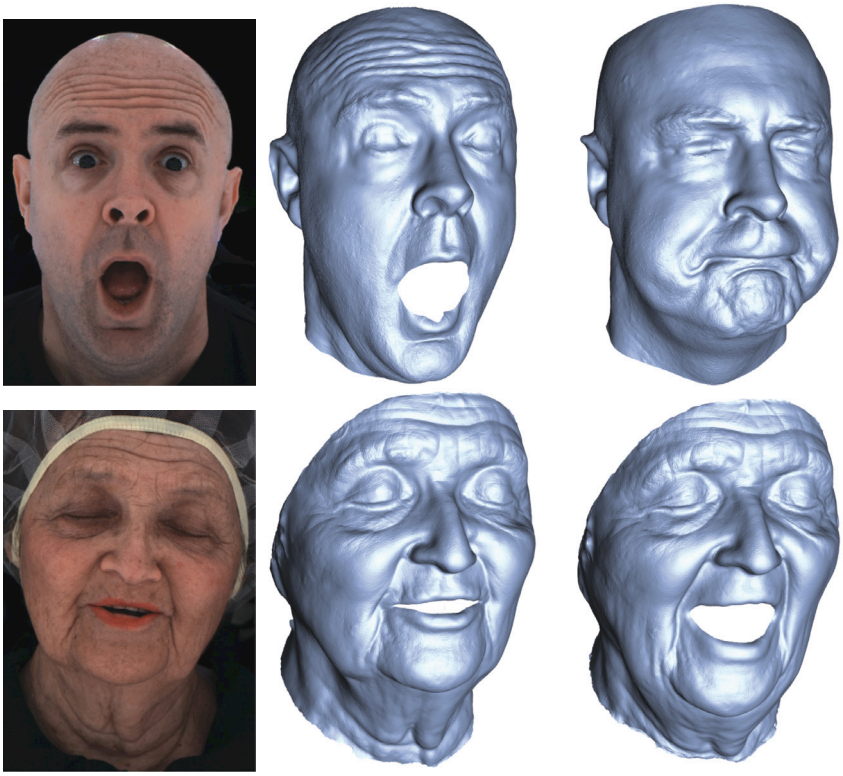


**Figure 5.1:** Passive facial motion capture system developed by Beeler, Hahn, Bradley, Bickel, Beardsley, Gotsman, Summer, and Gross at Disney Research [37]. The left column shows a sample input image of a subject, and the right two columns show reconstructed face mesh models in two different expressions. (Figure courtesy of Beeler et al.)

Bing Map will have a new 3D mode, too. All these maps provide fully interactive exploration and navigation experiences for effective mapping applications. These companies fly planes over the cities to take photographs on a massive scale but in a controlled manner, which are turned into high quality 3D city models.

Online cloud has become the standard storage for photographs. Ever-growing number of photographs are uploaded to online photo sharing websites such as Flickr [7], Panoramio [14], and Picasa [17],

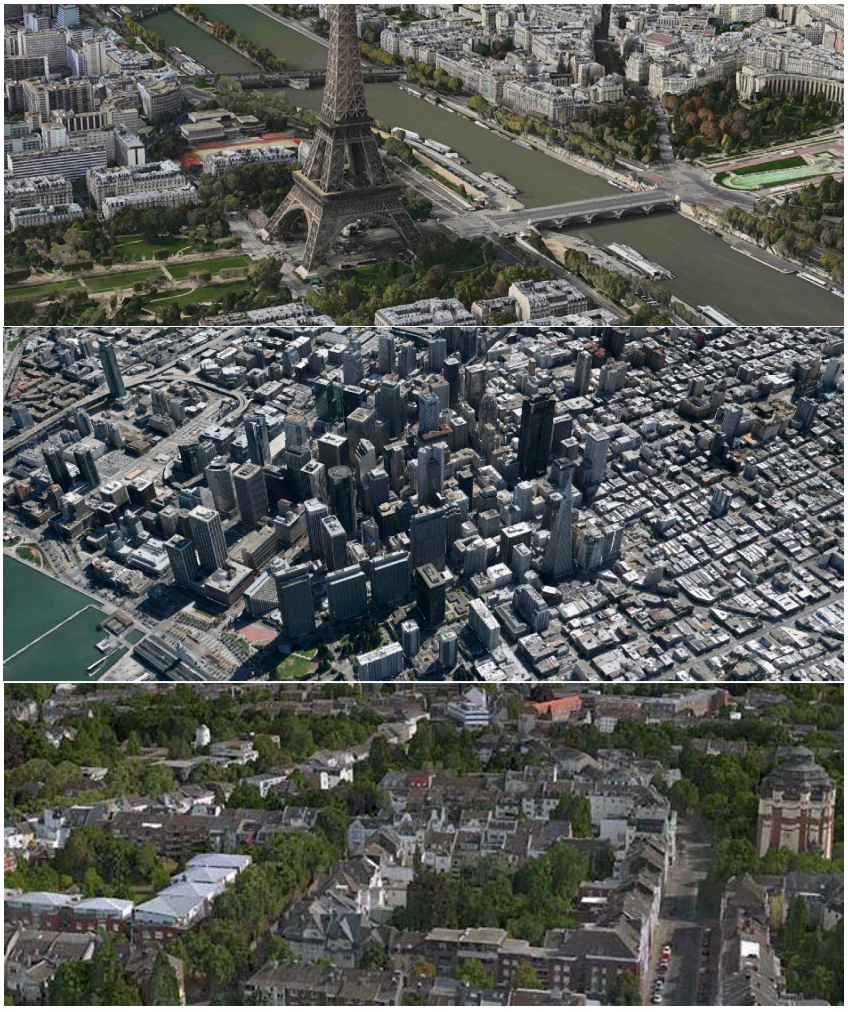


**Figure 5.2:** The use of MVS techniques in digital mapping products. From top to bottom, Apple Maps, Google Maps and Bing Maps.

everyday from all over the world. While one can easily access such images, for example, via a simple keyword search [23, 64], community photo collections pose additional challenges to 3D reconstruction. These online photographs are acquired by different cameras, at dif-


**Figure 5.3:** Google Maps Photo Tours provide with tens of thousands of 3D photo tours of famous landmarks all over the world with fluid 3D transition renderings. The top row shows such landmarks shown as red dots, across the globe, entire Europe, and one neighborhood in Paris, from left to right. The middle row shows the sequence of views in the photo tour for Sacre-Couer near Paris. The bottom row is a screenshot of Google Maps showing a photo tour of St. Peter's Basilica.

ferent times of day, seasons, and weather conditions. However, at the same time, they provides interesting views of the world, in which people do care and take photographs. Microsoft's Photosynth or Google's Photo Tours [127] (See Figure 5.3) harness rich set of community photographs. They provide immersive image-based rendering of famous landmarks all over the world, where MVS technique is used to reconstruct a geometric proxy for compelling image based rendering effects.

MVS researchers have started up successful 3D reconstruction companies. Acute3D [1] designs and markets the Smart3DCapture$^{TM}$system that allows the user to build high quality and resolution 3D models, where the applications range from cartography, architecture, defense, manufacturing, cultural heritage, and etc (See Figure 5.4). Pix4D [18] software converts thousands of aerial images taken by lightweight UAV (unmanned aerial vehicle) or aircraft into geo-referenced 2D orthomosaics and 3D surface models and point clouds. Started as a spin-off of EPFL's (Ecole Polytechnique Federale de Lausanne) Computer Vision Lab in Switzerland. Pix4D is a dynamic and rapidly expanding company (See Figure 5.5).


**Figure 5.4:** Acute3D develops Smart3DCapture$^{TM}$system for high quality 3D reconstructions that enable various applications [1].

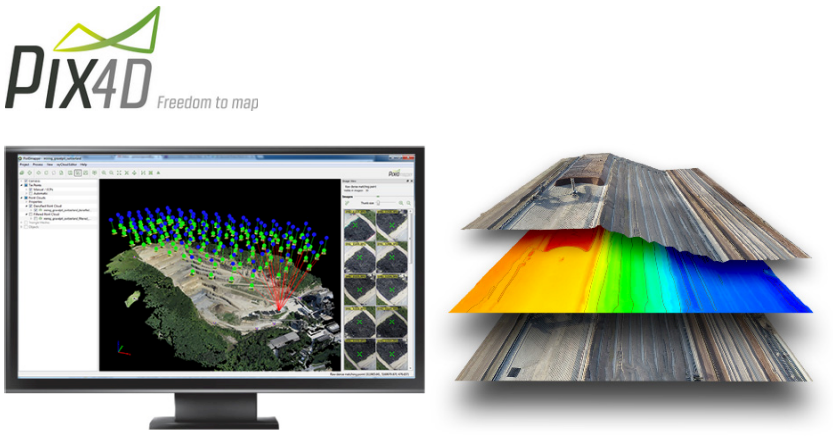



**Figure 5.5:** Pix4D is a company that turns thousands of aerial photographs from UAV into high quality 3D models [18].

# 6

# Limitations and Future Directions

MVS success stories are never-ending, but there are still numerous settings in which the MVS algorithms perform poorly. This chapter discusses the limitations of the current state-of-the-art and unexplored new research directions, then concludes the article.

## 6.1 Limitations

The major failure modes of the current MVS algorithms are classified into the following three types (see Figure 6.1).

**Lack of texture**: Homogeneous surfaces pose challenges to MVS algorithms, which are unfortunately prevalent in indoor environments. The Middlebury multi-view stereo benchmark [165] was a surprise to MVS researchers in that many MVS algorithms successful reconstructed an apparently textureless object "Dino" with high accuracy. It turns out that MVS algorithms are able to pick up very weak and intricate image textures, most of which come from shading and/or shadowing effects (the lighting was fixed during the dataset capture). However, these texture cues are weak and delicate, and images need to have very high quality.

**Figure 6.1:** Despite successes in MVS algorithms, there still exist many failure cases. Standard MVS algorithms cannot handle (left) non-Lambertian surfaces such as metallic or transparent objects, (middle) weakly textured surfaces, and (right) thin structures. (Image source: http://www.ikea.com)

An easy but effective solution is to use a projector(s) during image acquisition, which adds more texture to an object or a scene. When multiple cameras are involved, they need to be synchronized, which adds significant complexity to an acquisition setup, but standard MVS algorithms become applicable. The most successful example in this domain is probably the first version of the Microsoft Kinect Camera, which was originally developed by PrimeSense [19]. Their camera uses an infrared light projection and does not work outdoors in sunlight, but is a very effective 3D reconstruction solution utilizing MVS technology in indoor environments. Pure passive solutions also exist, where MVS and photometric principles [197] are fused to handle textureless surfaces and improve reconstruction qualities [95, 199]. Standard photometric stereo methods directly estimate the surface normal from multiple pixel intensities of the same 3D point, with a static camera to achieve pixel-accurate image registration. In a multi-view setup, the image registration require precise depth estimations, and the above techniques jointly solve MVS and photometric stereo problems.

**Thin structures**: To improve robustness, many MVS algorithms evaluate photo-consistency with a support domain $\Omega$ (see Section 2.1) that can be several pixels wide, instead of a single pixel. This makes it challenging to reconstruct very thin structures such as plants or wires, which could be as wide as only a few pixels in images.

One solution for certain applications is to use a photo-consistency measure that does not require a domain $\Omega$, together with a photo-consistency aggregation step that is edge-aware, see Section 2.1.8. Another option is to acquire a much denser set of photographs on a controlled (e.g., linear) camera motion path. This turns the MVS reconstruction from the problem of pixel matching to the problem of line extraction, where a single pixel in each image is enough to reliably estimate a depth value [117]. This framework also handles arbitrary unstructured 4D light fields. One can go beyond MVS and control the focus and aperture of a camera to take many images from a single viewpoint [89]. This results in many measurements for each pixel location, and the depth of a pixel can be estimated independent of other pixels, which enables the reconstruction of thin structures. Photometric stereo [197] is again effective, where 3D reconstruction can be performed on a pixel basis and has an advantage over thin structures. This technique requires the control of lighting, which significantly limits the application, but enables interesting applications such as real-time motion [91] capture in a lab-setting.

**Non-Lambertian surfaces**: MVS is all about matching image regions with similar or same appearances over multiple images, and hence, most of the algorithms assume Lambertian reflectance. While pure Lambertian surfaces are rare in practice, it is known and empirically verified that MVS algorithms work very well on non-Lambertian surfaces: As long as they contain some diffuse reflectance component, a photometric consistency function is able to identify and ignore images whose non-diffuse effects (e.g., specular highlights) are strong, then utilize the diffuse component in the remaining images. However, many objects in our world are highly non-Lambertian and make MVS algorithms fail.

This could be the most difficult one to overcome. Take specular objects, for example, which are prevalent in man-made environments. There exists work using a calibration object in a scene to infer specular 3D shapes [161]. Shape from specular flow analyzes the optical flow motion of specular surfaces in a video without resorting to a calibration object [188, 51, 22]. Unfortunately these methods produce either sparse or rough 3D shapes. For a general non-Lambertian surface with

complex reflectance properties, directly controlling the light source is one way to make high fidelity 3D reconstruction possible. While a standard photometric stereo principle [197] also assumes Lambertian surfaces, more advanced photometric stereo techniques can handle surfaces with complex reflectance, although these techniques are only applicable in a lab environment (i.e, dark room with a controllable point light source) [96, 28, 27, 107]. Jin, Soatto, and Yezzi proposed pure passive methods that directly model and analyze non-Lambertian effects for MVS matching [112, 113], which is one of the few attempts in MVS to explicitly go beyond Lambertian reflectance model. More recently, Oxholm and Nishino proposed an algorithm to estimate complex shape and reflectance, given natural illumination[150].

## 6.2 Open Problems

MVS researchers "solved" one class of 3D reconstruction problems in Computer Vision. The proposed algorithms are flexible, work very well, and produce 3D models that meet the needs of many practical applications. However, there are still many open problems with practical importance in the 3D reconstruction domain (See Figure 6.2).

**Aerial and ground fusion:** City-scale 3D reconstruction from aerial images has been under way for digital map creation by many IT companies [108, 30, 144]. Ground-based detailed city modeling has been demonstrated via MVS and/or laser range sensors [108, 144]. However, few attempts have been made in fusing both aerial and ground-based 3D models. The first challenge lies in the alignment of the aerial and ground 3D models. Due to the scale and viewpoint changes, standard SfM procedure tends to fail [168]. Even after the alignment, it is not a trivial task to reconstruct a single consistent 3D model that is as large as an entire city and contains details as small as individual objects in every store front [69, 65, 66].

**Inverse CAD:** Another important open problem is the reconstruction of a compact and human-editable 3D model from images, a process dubbed "Inverse CAD". Standard MVS algorithms are optimized to produce dense and accurate 3D models. However, such 3D models

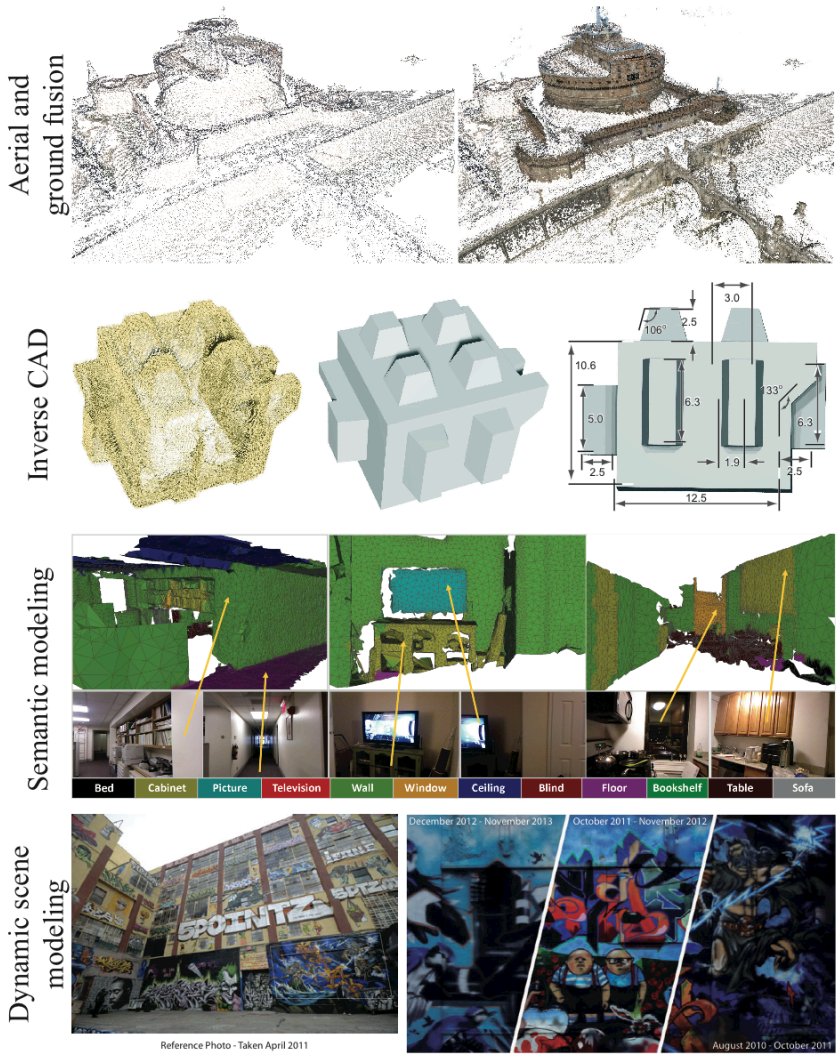



**Figure 6.2:** There are many open problems in the domain of 3D reconstruction, and the figure illustrates several examples. See the text for details. Figures courtesy of Qi et al. [168] (Aerial and ground fusion), Li et al. [133] (Inverse CAD), Valentin et al. [187] (Semantic modeling), and Matzen et al. [140] (Dynamic scene modeling), respectively.

are not suitable for human manipulation or post-editing. Inverse CAD algorithms have been popular in the Computer Graphics community, where the input is a high quality 3D point cloud from laser range sensors [133]. Effective techniques do not exist for MVS point clouds that suffer from noise and large reconstruction holes.

**Semantic modeling:** "Semantic 3D reconstruction" has become a popular research theme in Computer Vision and Robotics. The task is to reconstruct an annotated 3D model, where annotations are chairs, tables, and windows in an indoor scene, for examples [187]. More generally, there is growing interest in fusing recognition, segmentation, and reconstruction problems, the three major paradigms in Computer Vision. Recognition and segmentation have been jointly tackled from an early stage [86, 84]. Segmentation has played an important role in narrow baseline stereo problems [162] in order to deal with occlusions. In a multi-view stereo setting, occlusions are less of an issue, because occluded surfaces are often visible by the "side" cameras. While an approach exists to tackle segmentation in a MVS context [124], it is still an open question how to best tackle recognition, segmentation, and multi-view stereo reconstruction altogether.

**Dynamic scene modeling:** Dynamic scene modeling is yet another open area, where most existing 3D reconstruction methods, including MVS, assume a static rigid scene. Multiple synchronized video streams can be used to recover dense 3D structure and motion information via the same MVS principles [71, 73, 37, 46, 185]. However, such video acquisition setup is only possible in a lab environment. For general scenes, dynamic scene reconstruction becomes substantially more difficult, simply due to the lack of image coverage. The approaches either require restrictive assumptions on the scene geometry [140], or compromise on the reconstruction fidelity [211, 111]. Change detection is a form of dynamic scene modeling, where dynamic foreground objects lie in front of a static and rigid background [186, 211]. These techniques will be a key for autonomous cars and UAVs that drive or fly in real dynamic scenes.

## 6.3 Conclusions

This article provided a tutorial on multi view stereo, from data collection, algorithmic details to successful applications in the industry. MVS has undoubtedly been one of the most successful fields in Computer Vision in the last decade. It enables high quality 3D reconstruction from a handful of images taken by consumer grade cameras. In industry, with the explosion of cell phones and mobile devices equipped with cameras, we are capturing millions of photographs everyday all over the world at an ever growing pace. All these photos are a rich source of input data for MVS, with the ultimate goal of reconstructing the entire world.

## Acknowledgements

We would like to thank Richard Szeliski and Steve Seitz for fruitful discussion and advices at the early stage of the article writing. We would also like to thank all of our colleagues for providing us with images and figures for the article.

# References


[1] Acute3d. http://www.acute3d.com.

[2] Artoolkit. http://sourceforge.net/projects/artoolkit.

[3] Autodesk. http://en.wikipedia.org/wiki/Autodesk.

[4] Boujou. http://www.boujou.com.

[5] Cloud computing. http://en.wikipedia.org/wiki/Cloud_computing.

[6] Cuda: Compute unified device architecture. http://en.wikipedia.org/wiki/CUDA.

[7] Flickr. http://www.flickr.com.

[8] Google inc. http://www.google.com.

[9] Industrial light & magic. http://www.ilm.com.

[10] Kinect. http://en.wikipedia.org/wiki/Kinect.

[11] Lidar. http://en.wikipedia.org/wiki/Lidar.

[12] Multi-view environment. http://www.gris.informatik.tu-darmstadt.de/projects/multiview-environment.

[13] Openmvg (multiple view geometry). https://github.com/openMVG/openMVG.

[14] Panoramio. http://www.panoramio.com.

[15] Photo tours. http://google-latlong.blogspot.com/2012/04/visit-global-landmarks-with-photo-tours.html.

[16] Photosynth. http://photosynth.net.

[17] Picasa. http://picasa.google.com.

[18] Pix4d. http://pix4d.com.

[19] Primesense. http://www.primesense.com.

[20] Weta digital. http://www.wetafx.co.nz.

[21] Andrew Adams, Jongmin Baek, and Myers Abraham Davis. Fast high-dimensional filtering using the permutohedral lattice. In *Eurographics*, 2010.

[22] Y Adato, Y Vasilyev, O Ben-Shahar, and T Zickler. Toward a theory of shape from specular flow. In *IEEE International Conference on Computer Vision*, 2007.

[23] Sameer Agarwal, Yasutaka Furukawa, Noah Snavely, Ian Simon, Brian Curless, Steven M. Seitz, and Richard Szeliski. Building rome in a day. *Communications of the ACM*, 54(10):105–112, October 2011.

[24] Sameer Agarwal, Keir Mierle, and Others. Ceres solver. https://code.google.com/p/ceres-solver/.

[25] Sameer Agarwal, Noah Snavely, Ian Simon, Steven M. Seitz, and Richard Szeliski. Building Rome in a day. In *IEEE International Conference on Computer Vision*, 2009.

[26] A. Alahi, R. Ortiz, and P. Vandergheynst. Freak: Fast retina keypoint. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 510–517, June 2012.

[27] Neil Alldrin and David Kriegman. Toward reconstructing surfaces with arbitrary isotropic reflectance : A stratified photometric stereo approach. In *IEEE International Conference on Computer Vision*, 2007.

[28] Neil Alldrin and David Kriegman. Photometric stereo with non-parametric and spatially-varying reflectance. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[29] Pierre Alliez, David Cohen-Steiner, Yiying Tong, and Mathieu Desbrun. Voronoi-based variational reconstruction of unoriented point sets. In *Symposium on Geometry processing*, volume 7, pages 39–48, 2007.

[30] Apple. Apple maps. http://www.apple.com/ios/maps.

[31] Autodesk. 123d catch. http://www.123dapp.com/catch.

[32] C. Baillard, C. Schmid, A. Zisserman, and A. W. Fitzgibbon. Automatic line matching and 3D reconstruction of buildings from multiple views. In *ISPRS Conference on Automatic Extraction of GIS Objects from Digital Imagery*, pages 69–80, 1999.

[33] Simon Baker, Daniel Scharstein, J. P. Lewis, Stefan Roth, Michael J. Black, and Richard Szeliski. A database and evaluation methodology for optical flow. *International Journal of Computer Vision*, 92(1):1–31, March 2011.

[34] Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan Goldman. Patchmatch: a randomized correspondence algorithm for structural image editing. *ACM Transactions on Graphics*, 28(3):24, 2009.

[35] B.G. Baumgart. *Geometric modeling for computer vision*. PhD thesis, Stanford University, 1974.

[36] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Computer Vision and Image Understanding*, 110(3):346–359, June 2008.

[37] Thabo Beeler, Fabian Hahn, Derek Bradley, Bernd Bickel, Paul Beardsley, Craig Gotsman, Robert W. Sumner, and Markus Gross. High-quality passive facial performance capture using anchor frames. *ACM Transactions on Graphics*, 30:75:1–75:10, August 2011.

[38] S. Birchfield and C. Tomasi. Multiway cut for stereo and motion with slanted surfaces. In *IEEE International Conference on Computer Vision*, 1999.

[39] Stan Birchfield and Carlo Tomasi. A pixel dissimilarity measure that is insensitive to image sampling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20:401–406, 1998.

[40] Michael Bleyer, Christoph Rhemann, and Carsten Rother. Patchmatch stereo-stereo matching with slanted support windows. In *British Machine Vision Conference*, volume 11, pages 1–11, 2011.

[41] Matthew Bolitho, Michael Kazhdan, Randal Burns, and Hugues Hoppe. Multilevel streaming for out-of-core surface reconstruction. In *Symp. Geom. Proc.*, pages 69–78, 2007.

[42] Mario Botsch and Olga Sorkine. On linear variational surface deformation methods. *IEEE Transactions on Visualization and Computer Graphics*, 14(1):213–230, 2008.

[43] Jean-Yves Bouguet. Camera calibration toolbox for matlab. `http://www.vision.caltech.edu/bouguetj/calib_doc/`.

[44] Yuri Boykov and Vladimir Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(9):1124–1137, September 2004.

[45] Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(11):1222–1239, November 2001.

[46] Derek Bradley, Wolfgang Heidrich, Tiberiu Popa, and Alla Sheffer. High resolution passive facial performance capture. *ACM Transactions on Graphics*, 29(4):41, 2010.

[47] Neill D.F. Campbell, George Vogiatzis, Carlos Hernández, and Roberto Cipolla. Automatic 3D object segmentation in multiple views using volumetric graph-cuts. In *British Machine Vision Conference*, volume 1, pages 530–539, 2007.

[48] Neill D.F. Campbell, George Vogiatzis, Carlos Hernández, and Roberto Cipolla. Using multiple hypotheses to improve depth-maps for multi-view stereo. In *10th European Conference on Computer Vision*, volume 5302 of *LNCS*, pages 766–779, 2008.

[49] Neill D.F. Campbell, George Vogiatzis, Carlos Hernández, and Roberto Cipolla. Automatic 3D object segmentation in multiple views using volumetric graph-cuts. *Image and Vision Computing*, 28(1):14 – 25, January 2010.

[50] Neill D.F. Campbell, George Vogiatzis, Carlos Hernández, and Roberto Cipolla. Automatic object segmentation from calibrated images. In *Visual Media Production (CVMP), 2011 Conference for*, pages 126 – 137, Nov. 2011.

[51] Guillermo D. Canas, Yuriy Vasilyev, Yair Adato, Todd Zickler, Steven Gortler, and Ohad Ben-Shahar. A linear formulation of shape from specular flow. In *IEEE International Conference on Computer Vision*, 2009.

[52] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *ACM SIGGRAPH*, 1996.

[53] A.J. Davison, I.D. Reid, N.D. Molton, and O. Stasse. Monoslam: Real-time single camera slam. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):1052–1067, June 2007.

[54] Paul E. Debevec, Camillo J. Taylor, and Jitendra Malik. Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, ACM SIGGRAPH, pages 11–20, New York, NY, USA, 1996. ACM.

[55] Amaël Delaunoy and Emmanuel Prados. Gradient flows for optimizing triangular mesh-based surfaces: Applications to 3d reconstruction problems dealing with visibility. *International Journal of Computer Vision*, 95(2):100–123, November 2011.

[56] Amaël Delaunoy, Emmanuel Prados, P. Gargallo, J.-P. Pons, and P. Sturm. Minimizing the multi-view stereo reprojection error for triangular surface meshes. In *British Machine Vision Conference*, 2008.

[57] Tom Drummond Edward Rosten. Machine learning for high-speed corner detection. In *European Conference on Computer Vision*, pages 430–443. IEEE, 2006.

[58] O.D. Faugeras and R. Keriven. Variational principles, surface evolution, pde's, level-set methods, and the stereo problem. *IEEE Transactions on Image Processing*, 7(3):336–344, 1998.

[59] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2):167–181, 2004.

[60] V. Ferrari, T. Tuytelaars, and L. Van Gool. Simultaneous object recognition and segmentation by image exploration. In *European Conference on Computer Vision*, 2004.

[61] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with application to image analysis and automated cartography. *Commun. Assoc. Comp. Mach.*, 24:381–395, 1981.

[62] A. Fitzgibbon and A. Zisserman. Automatic camera recovery for closed or open image sequences. In *European Conference on Computer Vision*, volume I, pages 311–326, Freiburg, Germany, 1998.

[63] Forssén, Ringaby, and Hedborg. Computer vision on rolling shutter cameras. http://www.cvl.isy.liu.se/education/tutorials/rolling-shutter-tutorial, 2012.

[64] Jan-Michael Frahm, Pierre Fite-Georgel, David Gallup, Tim Johnson, Rahul Raguram, Changchang Wu, Yi-Hung Jen, Enrique Dunn, Brian Clipp, Svetlana Lazebnik, and Marc Pollefeys. Building rome on a cloudless day. In *European Conference on Computer Vision*, pages 368–381, Berlin, Heidelberg, 2010. Springer-Verlag.

[65] Simon Fuhrmann and Michael Goesele. Fusion of depth maps with multiple scales. *ACM Transactions on Graphics*, 30(6):148:1–148:8, December 2011.

[66] Simon Fuhrmann and Michael Goesele. Floating scale surface reconstruction. *ACM Transactions on Graphics*, 33(4):46, 2014.

[67] Yasutaka Furukawa, Brian Curless, Steven M. Seitz, and Richard Szeliski. Manhattan-world stereo. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[68] Yasutaka Furukawa, Brian Curless, Steven M. Seitz, and Richard Szeliski. Reconstructing building interiors from images. In *IEEE International Conference on Computer Vision*, 2009.

[69] Yasutaka Furukawa, Brian Curless, Steven M. Seitz, and Richard Szeliski. Towards Internet-scale multiview stereo. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

[70] Yasutaka Furukawa and Jean Ponce. Carved visual hulls for image-based modeling. *International Journal of Computer Vision*, 81(1):53–67, 2008.

[71] Yasutaka Furukawa and Jean Ponce. Dense 3d motion capture from synchronized video streams. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[72] Yasutaka Furukawa and Jean Ponce. Accurate camera calibration from multi-view stereo and bundle adjustment. *International Journal of Computer Vision*, 84(3):257–268, September 2009.

[73] Yasutaka Furukawa and Jean Ponce. Dense 3d motion capture for human faces. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1674–1681. IEEE, 2009.

[74] Yasutaka Furukawa and Jean Ponce. Accurate, dense, and robust multiview stereopsis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(8):1362–1376, August 2010.

[75] David Gallup, J-M Frahm, Philippos Mordohai, and Marc Pollefeys. Variable baseline/resolution stereo. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2008.

[76] David Gallup, Jan-Michael Frahm, Philippos Mordohai, Qingxiong Yang, and Marc Pollefeys. Real-time plane-sweeping stereo with multiple sweeping directions. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[77] David Gallup, Jan-Michael Frahm, and Marc Pollefeys. Piecewise planar and non-planar stereo for urban scene reconstruction. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

[78] P. Gargallo, E. Prados, and P. Sturm. Minimizing the reprojection error in surface reconstruction from images. In *IEEE International Conference on Computer Vision*, pages 1–8, 2007.

[79] M. Goesele, N. Snavely, B. Curless, H. Hoppe, and S.M. Seitz. Multi-view stereo for community photo collections. In *IEEE International Conference on Computer Vision*, pages 1–8, 2007.

[80] Michael Goesele, Brian Curless, and Steven M. Seitz. Multi-view stereo revisited. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2402–2409, 2006.

[81] Google. Google maps. `http://www.google.com/maps/views/u/0/streetview`.

[82] Brian Gough. *GNU Scientific Library Reference Manual - Third Edition*. Network Theory Ltd., 3rd edition, 2009.

[83] Markus Gross and Hanspeter Pfister. *Point-Based Graphics*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2007.

[84] Saurabh Gupta, Ross Girshick, Pablo Arbeláez, and Jitendra Malik. Learning rich features from rgb-d images for object detection and segmentation. In *European Conference on Computer Vision*, pages 345–360. Springer, 2014.

[85] Martin Habbecke and Leif Kobbelt. A surface-growing approach to multi-view stereo reconstruction. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[86] Bharath Hariharan, Pablo Arbeláez, Ross Girshick, and Jitendra Malik. Simultaneous detection and segmentation. In *European Conference on Computer Vision*, pages 297–312. Springer, 2014.

[87] C. Harris and M. Stephens. A combined corner and edge detector. In *Alvey Vision Conference*, pages 147–151, 1988.

[88] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2004.

[89] Samuel W. Hasinoff and Kiriakos N. Kutulakos. Confocal stereo. *International Journal of Computer Vision*, 81(1):82–104, 2009.

[90] Kaiming He, Jian Sun, and Xiaoou Tang. Guided image filtering. In *European Conference on Computer Vision*, ECCV'10, pages 1–14, Berlin, Heidelberg, 2010.

[91] C. Hernández, G. Vogiatzis, G. J. Brostow, B. Stenger, and R. Cipolla. Non-rigid photometric stereo with colored lights. In *IEEE International Conference on Computer Vision*, 2007.

[92] Carlos Hernández. *Stereo and Silhouette Fusion for 3D Object Modeling from Uncalibrated Images Under Circular Motion*. PhD thesis, Telecom Paris, Paris, France, 2004.

[93] Carlos Hernández and Francis Schmitt. Silhouette and stereo fusion for 3d object modeling. *Computer Vision and Image Understanding*, 96(3):367–392, 2004.

[94] Carlos Hernández, George Vogiatzis, and Roberto Cipolla. Probabilistic visibility for multi-view stereo. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[95] Carlos Hernández, George Vogiatzis, and Roberto Cipolla. Multiview photometric stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(3):548–554, 2008.

[96] Aaron Hertzmann and Steven M. Seitz. Example-based photometric stereo: Shape reconstruction with general, varying brdfs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(8):1254–1264, August 2005.

[97] V.H. Hiep, R. Keriven, P. Labatut, and J.-P. Pons. Towards high-resolution large-scale multi-view stereo. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1430–1437, June 2009.

[98] Heiko Hirschmuller. Accurate and efficient stereo processing by semi-global matching and mutual information. In *IEEE Conference on Computer Vision and Pattern Recognition*, IEEE Conference on Computer Vision and Pattern Recognition, pages 807–814, Washington, DC, USA, 2005. IEEE Computer Society.

[99] Heiko Hirschmuller and Daniel Scharstein. Evaluation of stereo matching costs on images with radiometric differences. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(9):1582–1599, 2009.

[100] Heiko HirschmÃijller. Evaluation of cost functions for stereo matching. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[101] Alexander Hornung, Er Hornung, and Leif Kobbelt. Robust and efficient photo-consistency estimation for volumetric 3d reconstruction. In *European Conference on Computer Vision*, pages 179–190, 2006.

[102] Alexander Hornung and Leif Kobbelt. Hierarchical volumetric multi-view stereo reconstruction of manifold surfaces based on dual graph embedding. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 503–510. IEEE, 2006.

[103] Alexander Hornung and Leif Kobbelt. Robust reconstruction of watertight 3 d models from non-uniformly sampled point clouds without normal information. In *Symposium on geometry processing*, pages 41–50. Citeseer, 2006.

[104] Alexander Hornung and Leif Kobbelt. Interactive pixel-accurate free viewpoint rendering from images with silhouette aware sampling. In *Computer Graphics Forum*, volume 28, pages 2090–2103. Wiley Online Library, 2009.

[105] A. Hosni, C. Rhemann, M. Bleyer, C. Rother, and M. Gelautz. Fast cost-volume filtering for visual correspondence and beyond. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(2):504–511, 2013.

[106] Xiaoyan Hu and P. Mordohai. A quantitative evaluation of confidence measures for stereo vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2121–2133, Nov 2012.

[107] Satoshi Ikehata and Kiyoharu Aizawa. Photometric stereo using constrained bivariate regression for general isotropic surfaces. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[108] Google Inc. Google maps. `http://maps.google.com`.

[109] M. Jancosek and T. Pajdla. Multi-view reconstruction preserving weakly-supported surfaces. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

[110] R.A. Jarvis. A perspective on range finding techniques for computer vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-5(2):122–139, March 1983.

[111] Dinghuang Ji, Enrique Dunn, and Jan-Michael Frahm. 3d reconstruction of dynamic textures in crowd sourced data. In *European Conference on Computer Vision*, 2014.

[112] Hailin Jin, Stefano Soatto, and Anthony J Yezzi. Multi-view stereo beyond lambert. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages I–171. IEEE, 2003.

[113] Hailin Jin, Stefano Soatto, and Anthony J Yezzi. Multi-view stereo reconstruction of dense shape and complex appearance. *International Journal of Computer Vision*, 63(3):175–189, 2005.

[114] Jörg H. Kappes, Bjoern Andres, Fred A. Hamprecht, Christoph Schnörr, Sebastian Nowozin, Dhruv Batra, Sungwoong Kim, Bernhard X. Kausler, Jan Lellmann, Nikos Komodakis, and Carsten Rother. A comparative study of modern inference techniques for discrete energy minimization problem. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2013. (accepted).

[115] M. Kazhdan, M. Bolitho, and H. Hoppe. Poisson surface reconstruction. In *Symp. Geom. Proc.*, 2006.

[116] M. Kazhdan and H. Hoppe. Screened poisson surface reconstruction. *ACM Transactions on Graphics*, 32(3):29:1–29:13, July 2013.

[117] Changil Kim, Henning Zimmer, Yael Pritch, Alexander Sorkine-Hornung, and Markus Gross. Scene reconstruction from high spatio-angular resolution light fields. In *ACM SIGGRAPH*, 2013.

[118] Junhwan Kim, V. Kolmogorov, and R. Zabih. Visual correspondence using energy minimization and mutual information. In *IEEE International Conference on Computer Vision*, volume 2, pages 1033–1040, 2003.

[119] Kalin Kolev and Daniel Cremers. Integration of multiview stereo and silhouettes via convex functionals on convex domains. In *European Conference on Computer Vision*, 2008.

[120] V. Kolmogorov and R. Zabih. Computing visual correspondence with occlusions using graph cuts. In *IEEE International Conference on Computer Vision*, volume 2, pages 508–515 vol.2, 2001.

[121] V. Kolmogorov and R. Zabih. Multi-camera scene reconstruction via graph cuts. In *European Conference on Computer Vision*, volume III, pages 82–96, Copenhagen, Denmark, 2002.

[122] V. Kolmogorov and R. Zabih. What energy functions can be minimized via graph cuts? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(2):147–159, 2004.

[123] Vladimir Kolmogorov. Convergent tree-reweighted message passing for energy minimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(10):1568–1583, 2006.

[124] Adarsh Kowdle, Sudipta N Sinha, and Richard Szeliski. Multiple view object cosegmentation using appearance and stereo cues. In *European Conference on Computer Vision*, pages 789–803. Springer, 2012.

[125] Sanjiv Kumar and Martial Hebert. Discriminative random fields: A discriminative framework for contextual interaction in classification. In *IEEE International Conference on Computer Vision*, pages 1150–1157, 2003.

[126] Akash Kushal and Jean Ponce. A novel approach to modeling 3d objects from stereo views and recognizing them in photographs. In *European Conference on Computer Vision*, 2006.

[127] Avanish Kushal, Ben Self, Yasutaka Furukawa, David Gallup, Carlos Hernández, Brian Curless, and Steven M. Seitz. Photo tours. In *3DImPVT*, 2012.

[128] K.N. Kutulakos and S. M. Seitz. A theory of shape by space carving. *International Journal of Computer Vision*, 38(3):199–218, 2000.

[129] Patrick Labatut, Jean-Philippe Pons, and Renaud Keriven. Efficient multi-view reconstruction of large-scale scenes using interest points, delaunay triangulation and graph cuts. In *IEEE International Conference on Computer Vision*, 2007.

[130] S. Leutenegger, M. Chli, and R.Y. Siegwart. Brisk: Binary robust invariant scalable keypoints. In *IEEE International Conference on Computer Vision*, pages 2548–2555, Nov 2011.

[131] Marc Levoy, Kari Pulli, Brian Curless, Szymon Rusinkiewicz, David Koller, Lucas Pereira, Matt Ginzton, Sean Anderson, James Davis, Jeremy Ginsberg, Jonathan Shade, and Duane Fulk. The digital michelangelo project: 3d scanning of large statues. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '00, pages 131–144, 2000.

[132] Maxime Lhuillier and Long Quan. A quasi-dense approach to surface reconstruction from uncalibrated ima ges. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(3):418–433, 2005.

[133] Yangyan Li, Xiaokun Wu, Yiorgos Chrysathou, Andrei Sharf, Daniel Cohen-Or, and Niloy J Mitra. Globfit: Consistently fitting primitives by discovering global relations. In *ACM Transactions on Graphics*, volume 30, page 52. ACM, 2011.

[134] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, ACM SIGGRAPH, pages 163–169, New York, NY, USA, 1987. ACM.

[135] David G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the International Conference on Computer Vision-Volume 2 - Volume 2*, IEEE International Conference on Computer Vision, pages 1150–, Washington, DC, USA, 1999. IEEE Computer Society.

[136] D.G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[137] Ziyang Ma, Kaiming He, Yichen Wei, Jian Sun, and Enhua Wu. Constant time weighted median filtering for stereo matching and beyond. In *IEEE International Conference on Computer Vision*, 2013.

[138] F. Maes, A. Collignon, D. Vandermeulen, G. Marchal, and P. Suetens. Multimodality image registration by maximization of mutual information. *Medical Imaging, IEEE Transactions on*, 16(2):187–198, 1997.

[139] D. Marr and T. Poggio. A computational theory of human stereo vision. *Proceedings of Royal Society of London*, 204(1156):301–328, 1979.

[140] Kevin Matzen and Noah Snavely. Scene chronology. In *European Conference on Computer Vision*, 2014.

[141] Xing Mei, Xun Sun, Mingcai Zhou, Shaohui Jiao, Haitao Wang, and Xiaopeng Zhang. On building an accurate stereo matching system on graphics hardware. In *ICCV Workshops*, pages 467–474, 2011.

[142] Paul Merrell, Amir Akbarzadeh, Liang Wang, Jan michael Frahm, and Ruigang Yang David NistÃľr. Real-time visibility-based fusion of depth maps. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[143] Microsoft. Bing maps. `http://blogs.bing.com/maps/2011/05/31/a-new-streetside-view/`.

[144] Microsoft. Bing maps. `http://www.bing.com/maps`.

[145] Richard A. Newcombe, S.J. Lovegrove, and A.J. Davison. Dtam: Dense tracking and mapping in real-time. In *IEEE International Conference on Computer Vision*, pages 2320–2327, 2011.

[146] David Nistér and Henrik Stewénius. Scalable recognition with a vocabulary tree. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2161–2168, 2006.

[147] M. Okutomi and T. Kanade. A multiple-baseline stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(4):353–363, April 1993.

[148] Carl Olsson, Johannes Ulen, and Yuri Boykov. In defence of 3d-label stereo. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2013.

[149] G. P. Otto and T. K. W. Chau. 'region-growing' algorithm for matching of terrain images. *Image Vision Computing*, 7(2):83–94, 1989.

[150] Geoffrey Oxholm and Ko Nishino. Multiview shape and reflectance from natural illumination. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2163–2170. IEEE, 2014.

[151] Emanuel Parzen. On estimation of a probability density function and mode. *The Annals of Mathematical Statistics*, 33(3):1065–1076, 1962.

[152] M. Pollefeys, R. Koch, and L. Van Gool. Self-calibration and metric reconstruction in spite of varying and unknown internal camera parameters. *International Journal of Computer Vision*, 32(1):7–25, 1999.

[153] M. Pollefeys, D. Nister, J.M. Frahm, A. Akbarzadeh, P. Mordohai, B. Clipp, C. Engels, D. Gallup, S.J. Kim, P. Merrell, et al. Detailed real-time urban 3d reconstruction from video. *International Journal of Computer Vision*, 78(2):143–167, 2008.

[154] Jean-Philippe Pons, Renaud Keriven, and Olivier Faugeras. Multi-view stereo reconstruction and scene flow estimation with a global image-based matching score. *International Journal of Computer Vision*, 72(2):179–193, April 2007.

[155] Soren Ragsdale. Airplane prop + cmos rolling shutter, 2009.

[156] X. Ren and J. Malik. Learning a classification model for segmentation. In *IEEE International Conference on Computer Vision*, 2003.

[157] Christian Richardt, Douglas Orr, Ian Davies, Antonio Criminisi, and Neil A. Dodgson. Real-time spatiotemporal stereo matching using the dual-cross-bilateral grid. In Kostas Daniilidis, Petros Maragos, and Nikos Paragios, editors, *European Conference on Computer Vision*, volume 6313 of *Lecture Notes in Computer Science*, page 510âĂŞ523, September 2010.

[158] Sébastien Roy and Ingemar J. Cox. A maximum-flow formulation of the n-camera stereo correspondence problem. In *IEEE International Conference on Computer Vision*, 1998.

[159] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb: An efficient alternative to sift or surf. In *IEEE International Conference on Computer Vision*, pages 2564–2571, Nov 2011.

[160] Szymon Rusinkiewicz and Marc Levoy. Qsplat: A multiresolution point rendering system for large meshes. In *ACM SIGGRAPH*, 2000.

[161] S. Savarese, M. Chen, and P. Perona. Local shape from mirror reflections. *International Journal of Computer Vision*, 64(1):31–67, 2005.

[162] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, 47(1/2/3):7–42, 2002.

[163] Daniel Scharstein and Richard Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, 47:7–42, 2002.

[164] Christopher Schroers, Henning Zimmer, Levi Valgaerts, AndrÃľs Bruhn, Oliver Demetz, and Joachim Weickert. Anisotropic range image integration. In *DAGM/OAGM Symposium'12*, pages 73–82, 2012.

[165] Steven M. Seitz, Brian Curless, James Diebel, Daniel Scharstein, and Richard Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 519–528, 2006.

[166] Steven M. Seitz and Charles R. Dyer. Photorealistic scene reconstruction by voxel coloring. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1067–, Washington, DC, USA, 1997. IEEE Computer Society.

[167] Ben Semerjian. A new variational framework for multiview surface reconstruction. In *European Conference on Computer Vision*, pages 719–734. Springer, 2014.

[168] Qi Shan, Changchang Wu, Brian Curless, Yasutaka Furukawa, Carlos Hernandez, and Steven M Seitz. Accurate geo-registration by ground-to-aerial image matching. In *3D Vision (3DV), 2014 2nd International Conference on*, volume 1, pages 525–532. IEEE, 2014.

[169] S.N. Sinha, P. Mordohai, and M. Pollefeys. Multi-view stereo via graph cuts on the dual of an adaptive tetrahedral mesh. In *IEEE International Conference on Computer Vision*, pages 1–8, 2007.

[170] Sudipta Sinha, Drew Steedly, and Richard Szeliski. Piecewise planar stereo for image-based rendering. In *IEEE International Conference on Computer Vision*, 2009.

[171] Sudipta N Sinha and Marc Pollefeys. Multi-view reconstruction using photo-consistency and exact silhouette constraints: A maximum-flow formulation. In *IEEE International Conference on Computer Vision*, 2005.

[172] N. Snavely, S. M. Seitz, and R. Szeliski. Skeletal graphs for efficient structure from motion. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008.

[173] Noah Snavely. Bundler: Structure from motion (sfm) for unordered image collections. `http://www.cs.cornell.edu/~snavely/bundler`.

[174] C. Strecha, R. Fransens, and L. Van Gool. Wide-baseline stereo from multiple views: a probabilistic account. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2006.

[175] C. Strecha, T. Tuytelaars, and L. Van Gool. Dense matching of multiple wide-baseline views. In *IEEE International Conference on Computer Vision*, pages 1194–1201 vol.2, 2003.

[176] C. Strecha, W. Von Hansen, L. Van Gool, P. Fua, and U. Thoennessen. On benchmarking camera calibration and multi-view stereo for high resolution imagery. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, June 2008.

[177] Christoph Strecha, Rik Fransens, and Luc Van Gool. Combined depth and outlier estimation in multi-view stereo. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2394–2401, Washington, DC, USA, 2006. IEEE Computer Society.

[178] Richard Szeliski. *Computer Vision: Algorithms and Applications*. Springer-Verlag New York, Inc., New York, NY, USA, 1st edition, 2010.

[179] Richard Szeliski, Ramin Zabih, Daniel Scharstein, Olga Veksler, Vladimir Kolmogorov, Aseem Agarwala, Marshall Tappen, and Carsten Rother. A comparative study of energy minimization methods for markov random fields with smoothness-based priors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(6):1068–1080, 2008.

[180] P. Tanskanen, K. Kolev, L. Meier, F. Camposeco Paulsen, O. Saurer, and M. Pollefeys. Live metric 3d reconstruction on mobile phones. In *IEEE International Conference on Computer Vision*, 2013.

[181] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *IEEE International Conference on Computer Vision*, pages 839–846, 1998.

[182] Carlo Tomasi. Visual reconstruction: Technical perspective. *Communications of the ACM*, 54(10):104–104, October 2011.

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

[184] R.Y. Tsai. Multiframe image point matching and 3-d surface reconstruction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-5(2):159–174, March 1983.

[185] Ali Osman Ulusoy, Octavian Biris, and Joseph L Mundy. Dynamic probabilistic volumetric models. In *IEEE International Conference on Computer Vision*, pages 505–512. IEEE, 2013.

[186] Ali Osman Ulusoy and Joseph L Mundy. Image-based 4-d reconstruction using 3-d change detection. In *European Conference on Computer Vision*, 2014.

[187] Julien PC Valentin, Sunando Sengupta, Jonathan Warrell, Ali Shahrokni, and Philip HS Torr. Mesh based semantic modelling for indoor and outdoor scenes. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2067–2074. IEEE, 2013.

[188] Yuriy Vasilyev, Todd Zickler, Steven Gortler, and Ohad Ben-Shahar. Shape from specular flow: Is one flow enough? In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2561–2568, 2011.

[189] P. Viola and W.M.I.I.I. Wells. Alignment by maximization of mutual information. In *IEEE International Conference on Computer Vision*, pages 16–23, 1995.

[190] G. Vogiatzis, C. Hernández, P. H S Torr, and R. Cipolla. Multi-view stereo via volumetric graph-cuts and occlusion robust photo-consistency. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(12):2241–2246, 2007.

[191] George Vogiatzis, P.H.S. Torr, and Roberto Cipolla. Multi-view stereo via volumetric graph-cuts. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2005.

[192] George Vogiazis and Carlos Hernández. Automatic camera pose estimation from dot pattern. `http://george-vogiatzis.org/calib/`.

[193] H-H. Vu, P. Labatut, R. Keriven, and J.-P Pons. High accuracy and visibility-consistent dense multi-view stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(5):889–901, May 2012.

[194] Martin J. Wainwright, Tommi S. Jaakkola, and Alan S. Willsky. Map estimation via agreement on trees: message-passing and linear programming. *IEEE Transactions on Information Theory*, 51:2005, 2005.

[195] Sven Wanner and Bastian Goldluecke. Spatial and angular variational super-resolution of 4d light fields. In *European Conference on Computer Vision*, pages 608–621, 2012.

[196] O.J. Woodford, P.H.S. Torr, I. D. Reid, and A. W. Fitzgibbon. Global stereo reconstruction under second order smoothness priors. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[197] Robert J. Woodham. Shape from shading. chapter Photometric Method for Determining Surface Orientation from Multiple Images, pages 513–531. MIT Press, Cambridge, MA, USA, 1989.

[198] Changchang Wu. Visualsfm : A visual structure from motion system. `http://ccwu.me/vsfm`.

[199] Chenglei Wu, Bennett Wilburn, Yasuyuki Matsushita, and Christian Theobalt. High-quality shape from multi-view stereo and shading under general illumination. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 969–976. IEEE, 2011.

[200] Chenyang Xu and Jerry L. Prince. Gradient vector flow: A new external force for snakes. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 66–71. IEEE Computer Society, 1997.

[201] Shuntaro Yamazaki, Srinivasa G. Narasimhan, Simon Baker, and Takeo Kanade. The theory and practice of coplanar shadowgram imaging for acquiring visual hulls of intricate objects. *International Journal of Computer Vision*, 81(3):259–280, 2009.

[202] Ruigang Yang and M. Pollefeys. Multi-resolution real-time stereo on commodity graphics hardware. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages I–211–I–217 vol.1, June 2003.

[203] Ruigang Yang, Marc Pollefeys, Hua Yang, and Greg Welch. A unified approach to real-time, multi-resolution, multi-baseline 2d view synthesis and 3d depth estimation using commodity graphics hardware. *International Journal of Image and Graphics*, 4(04):627–651, 2004.

[204] Jonathan S. Yedidia, William T. Freeman, and Yair Weiss. Generalized belief propagation. In *IN NIPS 13*, pages 689–695. MIT Press, 2000.

[205] Kuk-Jin Yoon and In-So Kweon. Adaptive support-weight approach for correspondence search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(4):650–656, 2006.

[206] Ramin Zabih and John Woodfill. Non-parametric local transforms for computing visual correspondence. In *European Conference on Computer Vision*, pages 151–158, Stockhold, Sweden, 1994.

[207] C. Zach, T. Pock, and H. Bischof. A globally optimal algorithm for robust tv-l 1 range image integration. In *IEEE International Conference on Computer Vision*, 2007.

[208] Andrei Zaharescu, Edmond Boyer, and Radu Horaud. Transformesh: A topology-adaptive mesh-based approach to surface evolution. In *Asian Conference on Computer Vision*, 2007.

[209] L. Zebedin, J. Bauer, K. Karner, and H. Bischof. Fusion of feature- and area-based information for urban buildings modeling from aerial imagery. In *IEEE International Conference on Computer Vision*, pages IV: 873–886, 2008.

[210] Gang Zeng, Sylvain Paris, Long Quan, and Maxime Lhuillier. Surface reconstruction by propagating 3d stereo data in multiple 2d images. In *European Conference on Computer Vision*, 2004.

[211] Enliang Zheng, Ke Wang, Enrique Dunn, and Jan-Michael Frahm. Joint object class sequencing and trajectory triangulation (jost). In *European Conference on Computer Vision*, 2014.