Image-based Rendering

A Brief Review & Study Notes

Ziyang Zhang

Ryerson University
August 22, 2013

Image Based Rendering is a series of techniques to capture, represent, and render a scene with pure images, instead of geometry. This topic has attracted extensive research effort since the beginning of computer vision and computer graphics. Reasons involve the difficulty to build 3D model for real world, the complexity to render and the lack of realism of geometry based models. Image based rendering, instead, is faster, more realistic, and images are easy to capture using cameras.

Serving as a direct study report as well as a study note, this document reviews several important aspects in image based rendering, including the plenoptic modeling, representation of scene at fixed viewpoint, and generating new viewpoints. Mainstream techniques up to the current date for each part are introduced. Some simulations and experiments are done on feature based image stitching. Several possible future works are proposed at the end of this document, as a result of intense literature review.

Key words: Image-based Rendering, Panoramica Imaging, Image Stitching, View Interpolation, Free Viewpoint Generating, CAVE.

Contents

1 Introduction

Image based rendering was proposed in ealy 1990s in the intersection of computer graphics and computer vision, as an alternative solution to *geometry based rendering*. Image based rendering aimed mainly to solve 2 problems, which stand in the way to realistic rendering.

The first problem is that geometry based rendering puts high requirement on computation resources. Rendering of complex scenes was a difficult task since the beginning of computer graphics, which can not be supported by regular computers in those years. Though the capability of graphic hardwares kept increasing within the last several decades, acoording to the famous Moore's Law, the virtual scenes to be rendered also became more and more complex. The expectation on the realism of virtual scenes becomes higher. Therefore high quality geometry based rendering still remains a time consuming process, and perfect realism seems impossible [49].

The second problem is the difficulty of building 3D geometry of the real world. 3D modeling and CAD (Computer Aided Disign) techniques have made some manually designed virtual scenes look quite real. However this requires large amount of manual, artistic as well as technical work, the extreme high cost of which determines that it could only been found in Hollywood's hit movies [27]. On the other hand, automatic 3D modeling is in its early stage. Though some impressive results have been reported [28][23][63], and some commercial softwaqres exist [14] [15]. most of these automatic systems suffers from limited usage. For example, some only work in small range and suit for indoor situation [28][23] [14], while some only suit for outdoor situation [63], and some are semi-automatic and need human interaction [15]. Also, the automatic 3D reconstruction lacks realism, compared to what people see in a real environment.

Image based rendering was proposed in this background. The main idea is that nothing looks more real than a photo taken from the real word. However, rendering means generating displaying image as seen at any given viewpoint and any viewing direction. If lighting and shadowing not considered, for geometry based rendering, since 3D structure is already known for every part of the scene, rendering is just a projection process. When a photo is taken, its viewpoint and angle of view are fixed. So image based rendering means generating new viewpoints from images at known viewpoints. This is generally done by interpolation, morphing and re-projection of images.

Kang [30] gave a comparison of geometry based rendering (referred as *3D model based rendering* in Kang's paper) and image based rendering, as shown in Table 1.

1.1 Plenoptic Modeling of Image Based Rendering

Mcmillan and Bishop [41] proposed to use the *plenoptic function* [2] to generalize the basic problem of Image Based Rendering (IBR).

The name *plenoptic* function [2] was combined from the latin root *plenus*, meaning complete or full, and *optic* pertaining to vision). It was used to define

Table 1: Comparison of geometry based rendering and image based rendering [30]

Geometry based rendering	Image based rendering
Explicit use of 3-D models	Directly uses collection of images
Uses conventional rendering pipeline	Based on interpolation or reprojection
Speed dependent on scene complexity	Speed independent of scene complexity
Relies on hardware accelerator for speed	Relies on processor speed
Requires sophisticated software for realism	Realism depends on input images

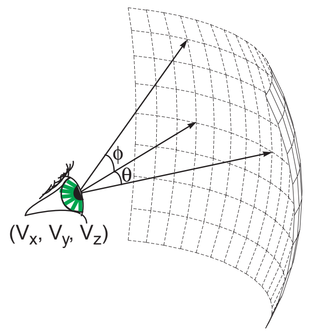


Figure 1: The plenoptic function describes all of the image information visible from a particular viewing position [41].

the pencil of rays visible from any point in space, at any time, and over any range of wavelengths.

As shown in Fig. 1, imagine an observer, for example an idealized eye or a camera, which can be located at any point (V_x, V_y, V_z) in space. From there we can select any of the viewable rays by choosing an azimuth and elevation angle (θ, ϕ) as well as a wavelength λ, and the plenoptic function should give the light intensity $P(V_x, V_y, V_z, \theta, \phi, \lambda)$.

In the case of a dynamic scene, we can additionally choose the time t, thus the plenoptic function becomes:

$$p = P(V_x, V_y, V_z, \theta, \phi, \lambda, t) \tag{1}$$

In computer vision and computer graphics terminology, the plenoptic function can be used as a scene representation. In order to generate a view from a given point in a particular direction, we just need to plug in the values for (V_x, V_y, V_z), (θ, ϕ) and t.

Mcmillan and Bishop [41] gave the following definition to image based rendering:

> We define a complete sample of the plenoptic function as a full spherical map for a given viewpoint and time value, and an incomplete sample as some solid angle subset of this spherical map.

Within this framework we can state the following problem definition for image-based rendering. *Given a set of discrete samples (complete or incomplete) from the plenoptic function, the goal of image-based rendering is to generate a continuous representation of that function.*This problem statement provides for many avenues of exploration, such as how to optimally select sample points and how to best reconstruct a continuous function from these samples.

1.2 Two Sampling Problems

The rest of this review will start from the geometry basis of computer vision, and then discuss some existing techniques to solve two main problem related to image based rendering.

The first problem is that given sample images at fixed viewpoint, but with different directions (θ, ϕ), how to reconstruct the plenoptic function at that certain viewpoint. This problem can be regarded as *panoramic imaging*, which has been well attacked by various methods.

The second problem is that given sample images for any viewing angle, but from different viewpoints, how to reconstruct the plenoptic function at another viewpoints, and how the choosing of sample viewpoints affects the reconstruction.

2 Geometry Basis

Computer vision starts from the acquiring of images. Most images of the real world today are taken with digital cameras. This section introduces the basic model of camera and geometry basis that will be applied to image processing.

2.1 Pinhole Camera model

A camera is a mapping between the 3D world (object space) and a 2D image [22]. Most cameras we are currently using, for example CMOS or CCD digital cameras with *rectilinear lens*, can be modeled as a pinhole projection model, as shown in Fig. 2. *Rectilinear lens* is a photographic lens that yields images where straight features, such as the walls of buildings, appear with straight lines [62].

Though the image plane is on the back side of the pinhole (entrance pupil of the lens, in the case of a real camera), and the image is reversed, we can simplify the model by the geometry shown in Fig. 3, where image plane is in front of the pinhole (the camera center).

Let the camera center be the origin of a Euclidean coordinate system, and consider the plane $Z = f$, which is called the *image plane* or *focal plane*. The distance f is the focal length of the camera lens. Under the pinhole camera model, a point in space with coordinates $\mathbf{X} = (X, Y, Z)^T$ is mapped to the point on the image plane where a line joining the point $\mathbf{X}$ to the camera center

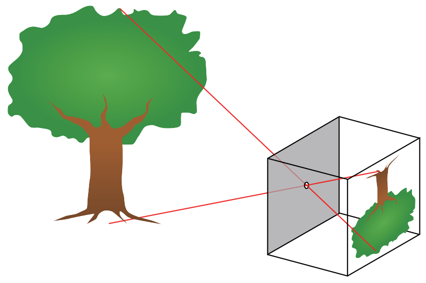


Figure 2: A diagram of a pinhole camera [61].

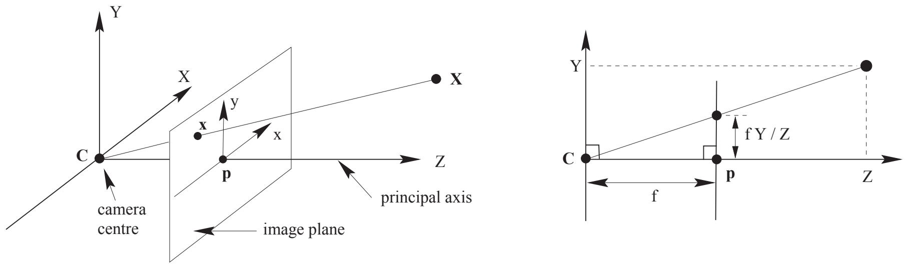


Figure 3: Pinhole camera geometry [22].

meets the image plane. It's easy to find that mapping from world to image coordinates:

$$(X, Y, Z)^T \rightarrow (f\frac{X}{Z}, f\frac{Y}{Z})^T \tag{2}$$

In literature world and image points are usually represented by homogeneous vectors, then the projection can be simply expressed as a linear mapping using matrix multiplication:

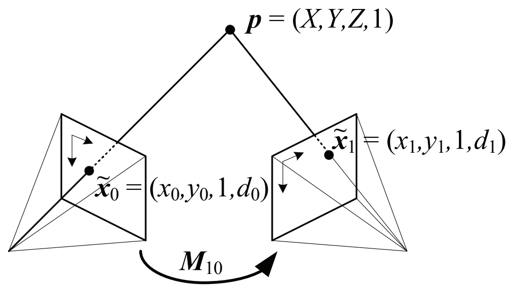
$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \rightarrow \begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{3}$$

or simply:

$$\mathbf{x} = \mathbf{P}\mathbf{X}. \tag{4}$$

The matrix $\mathbf{P}$ here is called the *camera projection matrix* or *camera matrix*, and can be written as $\text{diag}(f, f, 1)[\mathbf{I}|0]$.

The pinhole camera model described above is not suitable for all cases. For example, in the case of microshot, where the object to be photoed is close to the camera, the distance of the image plane to the projection center is no longer the focal length f. More precisely, only when the camera focuses to infinity,



$\boldsymbol{p} = (X,Y,Z,1)$
$\tilde{\boldsymbol{x}}_1 = (x_1,y_1,1,d_1)$
$\tilde{\boldsymbol{x}}_0 = (x_0,y_0,1,d_0)$
$\boldsymbol{M}_{10}$

Figure 4: Two cameras shooting the same scene [54].

the distance of the image plane to the projection center is set to f. However, for most images, the focus point is far away enough compared to camera's focal length, thus can considered as infinity.

Another case is using non-rectilinear lens, or *curvilinear lens*, for example fisheye lens. Curvilinear lens introduce non-linear characteristic into the projection and need a re-mapping process if we want rectilinear image. Though there is no lens that is perfect rectilinear, the distortion of today's camera lenses is reduced to very low level by advanced optics manufacturing technology. Therefore in most cases the simple pinhole camera model is applicable.

Note that the focal length described above is not the same as the *35mm equivalent focal length* generally used in photography. Equivalent focal length is calculated with a *crop factor* in order to keep the same angle of view as the standard 35mm photographic film [59][60].

2.2 Camera Motion Models

Many applications need to to take care of multiple images, for example video coding, object tracking, image stitching (will be discussed later), etc. Therefore it is necessary to find relationship between multiple images. This involves camera motion model, or multi-camera projection.

Consider the case of 2 cameras shooting the same scene, as shown in Fig. 4. The same world point $\mathbf{p} = (X, Y, Z, 1)$ is projected into two images, with $\mathbf{x_0} = (x_0, y_0, 1)$ and $\mathbf{x_1} = (x_1, y_1, 1)$ representing the projection points.

The camera model described previously places the projection center in the origin of world coordinate with its principle axis along Z axis, which is not always true. A $4 * 4$ rotation and translation matrix $\mathbf{E}$ (rigid-body motion with 6 degree of freedom) is used to represent the camera's location and orientation in space. Then the projection can be written as:

$$\mathbf{x}_0 = P_0 E_0 \mathbf{p} \tag{5}$$

and

$$\mathbf{x}_1 = P_1 E_1 \mathbf{p}. \tag{6}$$

Name	Matrix	# D.O.F.	Preserves:	Icon
translation	$\left[\, \boldsymbol{I} \,\middle|\, \boldsymbol{t} \,\right]_{2\times3}$	2	orientation $+\cdots$	□
rigid (Euclidean)	$\left[\, \boldsymbol{R} \,\middle|\, \boldsymbol{t} \,\right]_{2\times3}$	3	lengths $+\cdots$	◇
similarity	$\left[\, s\boldsymbol{R} \,\middle|\, \boldsymbol{t} \,\right]_{2\times3}$	4	angles $+\cdots$	◇
affine	$\left[\, \boldsymbol{A} \,\right]_{2\times3}$	6	parallelism $+\cdots$	▱
projective	$\left[\, \tilde{\boldsymbol{H}} \,\right]_{3\times3}$	8	straight lines	⏢

Table 2: 2D coordinate transromations [54].

where $\mathbf{p}$ is a world point, and $\mathbf{x}_0$, $\mathbf{x}_1$ are projection points on image 0 and image 1 respectively. If the same point $\mathbf{p}$ has its projection on both image, then the two images are related:

$$\mathbf{x}_1 = P_1 E_1 E_0^{-1} P_0^{-1} \mathbf{x}_0 = \mathbf{M}_{10} \mathbf{x}_0 \tag{7}$$

In equation 7, the matrix $\mathbf{M}_{10}$ can be different matrix, representing different camera motions. If the camera undergoes pure rotation around its projection center, the matrix can be simple as a rotation matrix:

$$\mathbf{M}_{10} = P_1 R_1 R_0^{-1} P_0^{-1} = P_1 R_{10} P_0^{-1}. \tag{8}$$

And the mapping between 2 images becomes:

$$\begin{pmatrix} x_1 \\ y_1 \\ 1 \end{pmatrix} \sim \begin{bmatrix} f_1 & & \\ & f_1 & \\ & & 1 \end{bmatrix} \mathbf{R}_{10} \begin{bmatrix} f_0^{-1} & & \\ & f_0^{-1} & \\ & & 1 \end{bmatrix} \begin{pmatrix} x_0 \\ y_0 \\ 1 \end{pmatrix} \tag{9}$$

Table 2 gives several different form of 2D transforms.

In general, $\mathbf{M}_{10}$ can be represented by a $3 * 3$ *homography matrix* with 8 parameters. The registration of 2 images becomes the estimation of these parameters.

Cylindrical and Spherical Coordinates

As a special case of motion model, many images taken for panorama only involve camera *panning*(rotation in horizontal plane). Therefore, warping the images into *cylindrical* coordinates results in pure one dimensional translation among images.

Also, when camera undergoes only panning and *tilting* (rotation in vertical plane), we can warp images into *spherical* coordinates and the image transform will become two dimensional translation.

Fig. 5 shows the projection from a world point $\mathbf{p} = (X, Y, Z)$ onto a cylindrical coordinates (θ, h) and a spherical coordinates (θ, ϕ). The projections can

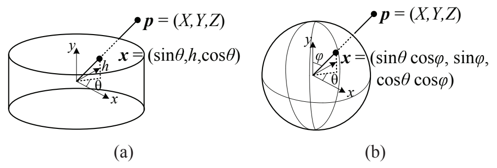

$\boldsymbol{p} = (X,Y,Z)$
$\boldsymbol{x} = (\sin\theta, h, \cos\theta)$
y
x
h
θ
(a)
$\boldsymbol{p} = (X,Y,Z)$
$\boldsymbol{x} = (\sin\theta \cos\varphi, \sin\varphi, \cos\theta \cos\varphi)$
y
x
φ
θ
(b)

Figure 5: Cylindrical and spherical coordinates [54].

be represented as:

$$\begin{aligned} \theta &= \arctan \frac{X}{Z} \\ h &= \frac{Y}{\sqrt{X^2 + Z^2}} \end{aligned} \tag{10}$$

and

$$\begin{aligned} \theta &= \arctan \frac{X}{Z} \\ \phi &= \arctan \frac{Y}{\sqrt{X^2 + Z^2}}. \end{aligned} \tag{11}$$

Once the images are warped into cylindrical or spherical surface, relationship among images becomes pure translation. However, this requires the precise placement of camera, often with a tripod.

3 Panoramic Imaging

As introduced previously, the reconstruction of plenoptic function at a given viewpoint can be regarded as the problem of panoramic imaging. Since the beginning of photography, people have trying different methods to take photos with large viewing angles, including hardware approach and software approach. This section will give a brief introduction to image stitching and several alternative hardware approaches to make panoramic images.

3.1 Image Stitching

Image stitching as a software approach has been under extensive research in the past 2 decades. Now it is easy to create panoramas from multiple images taken separately.

In general, relationship among different images are represented by camera motion models introduced in previous section. First, parameters in the camera matrix or image transform matrix need to be estimated, using directly images pixels or features extracted from image. Then, choose a panorama format that determines the coordinate system, and transform all images to that coordinate. Finally, images that have overlapping points are blended together and sometimes additional steps are needed to remove ghosting and mis-alignment.

3.1.1 Direct(Pixel-based) Alignment

A lot of early image stitching works were directly pixel based, [51][6][54] [55][40][3] and the motion estimation methods used were similar with those in video coding, object tracking, robot navigation etc. Bergen et. al. [6] mentioned that though motion estimation methods can be different when used in different application, they can be generalized in the same framework, except that they use different motion models.

The motion model determines the warping of images. Suppose that we want to align 2 images $I_0(\mathbf{x})$ and $I_0(\mathbf{x})$ together, a motion model $\mathbf{m}$ is used to establish a warping, $\mathbf{x}' = \mathbf{f}(\mathbf{x}; \mathbf{m})$.

The motion model can be a simple translation, for example $\mathbf{x}' = \mathbf{x} + \mathbf{t}$. It also could be a rotation, or any transforms defined in Table 2, with the most complex case a 8-parameter perspective transformation (homography):

$$\mathbf{x}' \sim \mathbf{M}\mathbf{x} = \begin{bmatrix} m_0 & m_1 & m_2 \\ m_3 & m_4 & m_5 \\ m_6 & m_7 & m_8 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \tag{12}$$

where $\sim$ indicates equality up to scale, since $\mathbf{m}$ has only 8 parameters and is invariant to scaling.

First an image is warped by this motion model:

$$\tilde{I}_1(\mathbf{x}) = I_1(\mathbf{f}(\mathbf{x}; \mathbf{m})) \tag{13}$$

The task is then to find a deformation of $\tilde{I}_1(\mathbf{x})$ which brings it into closer registration with $I_0(\mathbf{x})$ and which can also be used to update the parameter $\mathbf{m}$. The warp/register/update loop can then be repeated, until convergence to a desired result.

Error Metrics

The *register* step is usually done by minimizing an error function.

The most widely used error function is the *sum of squared differences* (L2 norm):

$$E_{SSD}(\mathbf{m}) = \sum_i \left[\tilde{I}_1(\mathbf{x}_i) - I_0(\mathbf{x}_i)\right]^2 = \sum_i e_i^2 \tag{14}$$

Other error metrics can also be used, for example, the *sum of absolute differences* (L1 norm):

$$E_{SAD}(\mathbf{m}) = \sum_i |\tilde{I}_1(\mathbf{x}_i) - I_0(\mathbf{x}_i)| = \sum_i |e_i| \tag{15}$$

or negative of normalized cross-correlation:

$$E_{NCC}(\mathbf{m}) = \frac{\sum_i \tilde{I}_1(\mathbf{x}_i) I_0(\mathbf{x}_i)}{\sqrt{\sum_i \tilde{I}_1(\mathbf{x}_i)^2 \sum_i I_0(\mathbf{x}_i)^2}} \tag{16}$$

Iterative Solution

Since the 8-parameter perspective motion model is a general case, which can include pure translation and rotation situations, only the solution to this general case is introduce here. The solution to pure translation and pure rotation can be simpler, but in the same form [54] [51].

A very widely used algorithm to minimize the error function in (14) is the Lucas-Kanade algorithm[40], and many other algorithms were similar with it [51][54]. Baker and Matthew [3] reviewed the derivation of Lucas-Kanade algorithm and how it's related to other similar algorithms. All these algorithms use the *SSD* error metric, and approximate the pixel difference e_i with its first order Taylor expansion, namely the *Gauss-Newton* approximation [6][3].

In the algorithm description below, the terminology in [51] and [55] is used.

Given a current motion estimation $\mathbf{M}$ as in (12), we update the parameters in each iteration using:

$$\mathbf{M} \leftarrow (\mathbf{I} + \mathbf{D})\mathbf{M} \tag{17}$$

where

$$\mathbf{D} = \begin{bmatrix} d_0 & d_1 & d_2 \\ d_3 & d_4 & d_5 \\ d_6 & d_7 & d_8 \end{bmatrix} \tag{18}$$

Resampling image I_1 with the new transformation $(\mathbf{I} + \mathbf{D})\mathbf{M}$ is the same as warping the sampled image $\tilde{I}_1$ by $\mathbf{x}'' \sim (\mathbf{I} + \mathbf{D})\mathbf{x}$, i.e.,

$$\begin{aligned} x'' &= \frac{(1 + d_0)x + d_1 y + d_2}{d_6 x + d_7 y + (1 + d_8)} \\ y'' &= \frac{d_3 x + (1 + d_4)y + d_5}{d_6 x + d_7 y + (1 + d_8)} \end{aligned} \tag{19}$$

The squared error function in (14) is approximated by first order Taylor expansion:

$$\begin{aligned} E(\mathbf{d}) &= \sum_i \left[\tilde{I}_1(\mathbf{x}''_i) - I_0(\mathbf{x}_i)\right]^2 \\ &\approx \sum_i \left[\tilde{I}_1(\mathbf{x}_i) + \nabla \tilde{I}_1(\mathbf{x}_i) \frac{\delta \mathbf{x}''_i}{\delta \mathbf{d}} \mathbf{d} - I_0(\mathbf{x}_i)\right]^2 \\ &= \sum_i \left[\mathbf{g}_i^T \mathbf{J}_i^T \mathbf{d} + e_i\right]^2 \end{aligned} \tag{20}$$

where $e_i = \tilde{I}_1(\mathbf{x}_i) - I_0(\mathbf{x}_i)$ is the current intensity or color error, and $\mathbf{g}_i^T = \nabla \tilde{I}_1(\mathbf{x}_i)$ is the image gradient of $\tilde{I}_1$ at $\mathbf{x}_i$; $\mathbf{d} = (d_0, ..., d_8)$ is the incremental motion parameter vector, and $\mathbf{J}_i = \mathbf{J}_\mathbf{d}(\mathbf{x}_i)$, where

$$\mathbf{J}_\mathbf{d}^T(\mathbf{x}) = \frac{\delta \mathbf{x}''}{\delta \mathbf{d}} = \begin{bmatrix} x & y & 1 & 0 & 0 & 0 & -x^2 & -xy & -x \\ 0 & 0 & 0 & x & y & 1 & -xy & -y^2 & -y \end{bmatrix}^T \tag{21}$$

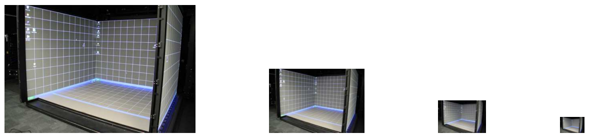


Figure 6: An example of a 4 level Gaussian Pyramid

is the Jacobian matrix of the rewarping coordinate $\mathbf{x}''$ with respect to $\mathbf{d}$.

Minimizing (20) can be solved by various methods. Lucas-Kanade algorithm [40] simply take the derivative of (20) with respect to $\mathbf{d}$ and let this derivative be zero, to get the incremental motion parameter. [51] solve this least-square problem through the *normal equations*

$$\mathbf{A}\mathbf{d} = -\mathbf{b} \tag{22}$$

where

$$\mathbf{A} = \sum_i \mathbf{J}_i \mathbf{g}_i \mathbf{g}_i^T \mathbf{J}_i^T \tag{23}$$

is the *Hessian* matrix, and

$$\mathbf{b} = \sum_i e_i \mathbf{J}_i \mathbf{g}_i \tag{24}$$

is the *accumulated gradient* or *residual.* These linear equations can be easily solved using numerical analysis methods. Note that in practice d_8 is often set to 0, or $\mathbf{A}$ would be singular, because the motion matrix is scale invariant and has 8 parameters.

Translational motion is a special case of the general perspective transformation, where $\mathbf{J}$ is a $2 * 2$ identity matrix because only m_2 and m_5 are used. The translational motion model can be used to construct sylindrical and spherical panorama if we warp images to cylindrical or spherical coordinates using a known focal length, as shown previously.

Hierachical Approach

The iterative solutions to the optimization problem requires the increase in each iteration very small, to achieve good first order Taylor approximation. However, the displacement among images are often too large. It is hard to find an initial estimation. And it is difficult to do full global search under this situation.

Therefore Hierachical approaches are often used. The basic idea is to first search coarsely over large range, and gradually do finer search in higher level. In practise this often involves construction of an image pyramid (for example *Gaussian Pyramid* or *Laplasian Pyramid* [9]). A 4 level Gaussian Pyramid of an image is shown in Fig. 6.

The warp/register/update loop is first performed on lower level images, to find a coarse estimation. This estimation is then forwarded to a higher level as an initial guess, and the warp/register/update loop is performed again in this new level. This process is repeated until the registration is done in the highest level.

3.1.2 Feature based Registration

Direct (pixel-based) alignment uses all the overlapping pixels and thus can achieve high registration accuracy. However, it needs an initialisation, which is typically provided by user input to approximatedly aligh the images, or a fixed image ordering [8]. Though there are hierachical approches that can do coarse alignment in lower level pyramid, the lower level images often lose significant details and can not perform well [54].

An alternative way is to use feature-based alignment. This is to first extract distinctive *features* from each image, to match these features to establish a global correspondence, and to then estimate the geometric transformation between the images. Early feature-based methods were not robust enough that they often got confused in regions that were either too testured or not textured enough. Furthermore establishing correspondences relied on simple cross-correlation between patches surrounding the feature points, which did not work well when images were rotated or had foreshortening due to homographies [54].

Today, the feature detection and matching methods are remarkably robust, and can even be used for known object recognition from widely separated views. For example SIFT features [39] can match images that differ in scale, orientation, and even foreshortening.

Brown and Lowe [38] [8] demonstrated an automatic system to synthesize panoramas. Their methods have already been applied to many commercial softwares [7], for example the Microsoft ICE (Image Composite Editor), the Autopano toolkit, AutoStitch app on iPhone and iPad, etc. The following in this section will introduce their approach.

SIFT Feature Matching

The first step is to extract and match SIFT features between all image pairs. SIFT features are located at scale-space maxima/minima of a difference of Gaussian function. At each feature location, a characteristic scale and orientation is established. This gives a similarity-invariant frame in which to make measurements [8] [34] [39]. An invariant descriptor is then computed in that frame, by accumulating local gradients in orientation histograms. The usage of gradients orientation histogram instead of directly pixels in that frame makes the descriptor more robust to small shifts and rotations, as well as illumination differences [34].

After locating feature points in the images, we need to match a key point in one image to a points in another image. This can be done by finding its nearest neighbor among all the feature points in the other image. The nearest neighbor is defined as the point with minimum Euclidean distance between the descriptor vector. In practise a threshold τ is often used to reject those points, whose distance to the closest neighbor is larger than τ times the distance to the second-closest neighbor.

Fig. 7(c) gives an example of SIFT feature matching. The yellow circles represents the feature points detected, and the cyan lines represents the matches between the 2 images, using a threshold of $\tau = 0.6$. When most of the matchings are correct, there are some mis-matchings existing. This makes it necessary eliminate mis-matchings and use the correct matchings to estimate the transformation.

RANSAC Parameter Estimation

Proposed by Fischler and Bolles [16], the Random Sample Consensus (RANSAC) algorithm is a parameter estimation approach designed to cope with a large proportion of outliers (gross errors) in the input data. Fischler and Bolles [16] mentioned that local feature detectors make 2 types of errors – classification errors and measurement errors. Measurement errors generally follow a normal distribution and can be smoothed out by averaging many data samples. This is why many conventional estimation techniques use as much of the data as possible. However, as already shown in fig. 7, classification errors are gross errors and can largely affect the result if averaged with other samples.

RANSAC is a re-sampling technique that generates candidate solutions by using the minimum number observations required to estimate the model parameters. For example, in order to estimate the 8 parameter transformation matrix (as in (7) and (12)), minimum 4 pairs of matching points are needed.

The RANSAC algorithm is summarized as follows:

```
RANSAC Algorithm
1:  Select randomly the minimum number of points required to determine the
    model parameters.
2:  Solve for the parameters of the model.
3:  Apply the model to all points. Find the inliers, which are those points that
    fit the model with a predefined tolerance $\epsilon$.
4:  If number of inliers is greater than a predefined threshold $t$, re-estimate the
    model parameters using all the inliers and terminate.
5:  Otherwise, repeat step 1 through 4 (maximum of $N$ times).
```

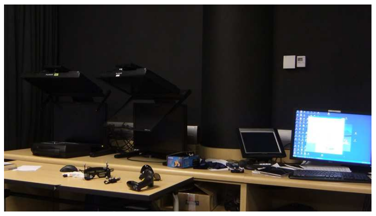
The number of iterations, N is chosen to ensure a probability p that at least one of the random selections does not include an outlier. Suppose in the whole data set, the probability of selecting an inlier is w. And minimum m points are needed to estimate model parameters. Then

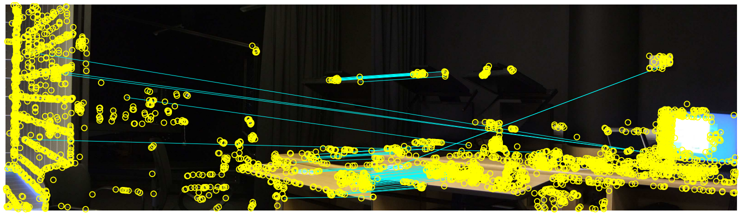
$$1 - p = (1 - w^m)^N. \tag{25}$$

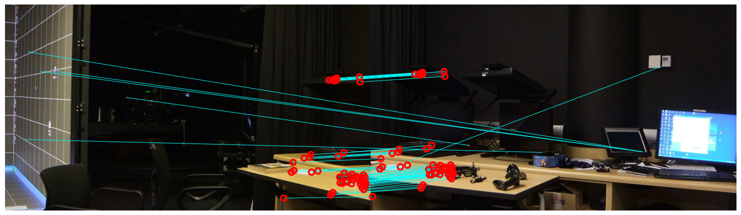
So

$$N = \frac{\log{(1-p)}}{\log{(1-w^m)}}. \tag{26}$$

(a) Original image 1


(b) Original image 2


(c) SIFT feature points (yellow circles) and matching (cyan lines)


(d) RANSAC inliers (red circles)

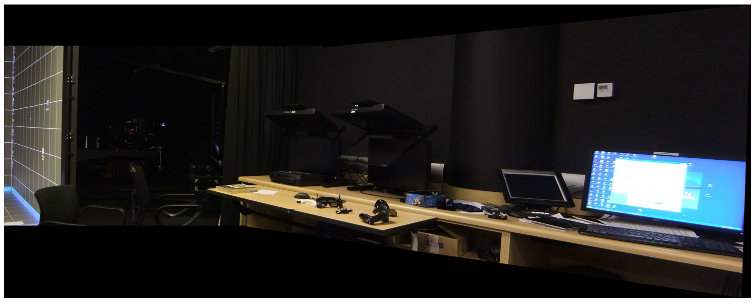

(e) Image aligned according to estimated homography

Figure 7: For the 1600×900 input images, there are 60 feature matches, and 52 inliers after RANSAC. RANSAC are running with $w = 0.5, m = 4$ and $p = 0.999$ in (26). Images are taken with the same exposure setting.

Note that if $w^m \ll 1$, then $N \approx -\log(1-p)w^{-m}$ [16]. Thus if we want $p = 0.99$, and $w^m \ll 1$, then $N \approx 4.6w^{-m}$.

Fig. 7(d) gives an example of 52 selected inliers out of 60 pairs of matched points using SIFT feature.

Estimating Homography From Matching Points

In each iteration of RANSAC algorithm as well as in its ending step, we need to estimate a homography transformation matrix from 4 or more pairs of image points. Suppose we have K pairs of points, $\mathbf{x}_k = (x_k, y_k, 1)$ matching with $\mathbf{x}'_k = (x'_k, y'_k, 1)$, with the model shown in (12), and with $m_8 = 1$, because the matrix is scale invariant. After simple algebra derivation [34], we have

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x'_1x_1 & -x'_1y_1 & -x'_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y'_1x_1 & -y'_1y_1 & -y'_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_K & y_K & 1 & 0 & 0 & 0 & -x'_Kx_K & -x'_Ky_K & -x'_K \\ 0 & 0 & 0 & x_K & y_K & 1 & -y'_Kx_K & -y'_Ky_K & -y'_K \end{bmatrix} \cdot \begin{bmatrix} m_0 \\ m_1 \\ m_2 \\ m_3 \\ m_4 \\ m_5 \\ m_6 \\ m_7 \\ 1 \end{bmatrix} = \mathbf{0}, \quad (27)$$

or simply:

$$\mathbf{Am} = \mathbf{0} \quad (28)$$

If $K = 4$, the linear equation above has one unique solution. When $K > 4$, the matrix $\mathbf{A}$ is over-determined and we can only find an approximate solution. Often the least square criteria is used, to minimize sum of square Euclidean distance error:

$$E_{LS} = \sum_k \|\mathbf{r}_k\|^2 = \|\tilde{\mathbf{x}}'_k(\mathbf{x}_k, \mathbf{m}) - \mathbf{x}'_k\|^2. \quad (29)$$

Under this criteria of minimizing least square error, the estimation of $\mathbf{m}$ can be done from a singular value decomposition of $\mathbf{A}$:

$$\mathbf{A} = \mathbf{UDV}^T = \mathbf{UD} \begin{bmatrix} v_{11} & \cdots & v_{19} \\ \vdots & \ddots & \vdots \\ v_{91} & \cdots & v_{99} \end{bmatrix}^T \quad (30)$$

and use the last column of $\mathbf{V}$ as the estimation:

$$\mathbf{m} = \frac{[v_{19}, v_{29}, \cdots, v_{99}]}{v_{99}} \quad (31)$$

Fig. 7(e) shows the aligned images by transforming the second image using the estimated homography from 52 RANSAC inliers.

3.1.3 Bundle Adjustment

The direct alignment and feature-based registration introduced above discussed how to register one pair of images. In most applications, there are more than a single pair of images to be aligned. We can simply concatenate the pairwise scheme, i.e. add one new image at a time based on one previously added image. However this would cause accumulated errors and disregard overlapping content in multiple images, for example, the end of a panorama have a displacement with the beginning, where in fact they should align together.

Therefore we need to find a *globally consistent* set of alignment parameters that minimize the mis-registration between all pairs of images. This is called *global registration* or *bundle adjustment* [54]. Techniques for this kind of problem have been well developed in photogrammetry literature and have been adjusted to or reinvented in computer vision community [57].

The global registration process can be done by adding new images to the panorama one at a time, aligning the new image with all its overlapping images already in the collection. Another approach is to simultaneously align all the images together using a least square framework to distribute mis-registration errors among all images.

In order to do this, we need to extend the pairwise matching criteria (14) and (29) to a global cost function that involves the parameters for every image. This error function can be generalized as:

$$e = \sum_{i=1} \sum_{j \in \mathcal{I}(i)} \sum_{k \in \mathcal{F}(i,j)} h(\mathbf{r}_{ij}^k) \tag{32}$$

where i is the image number; $\mathcal{I}(i)$ is the set of images matching to image i; $\mathcal{F}(i,j)$ is the set of feature matches between image i and j; and $\mathbf{r}_{ij}^k$ is the residual of the corresponding matched feature points $\mathbf{u}_i^k$ and $\mathbf{u}_j^l$ ($\mathbf{u}_i^k$ denotes the position of kth feature in image i),

$$\mathbf{r}_{ij} = \mathbf{u}_i^k - \mathbf{p}_{ij}^k = \mathbf{u}_i^k - \mathbf{K}_i \mathbf{R}_i \mathbf{R}_j^T \mathbf{R}_j^{-1} \mathbf{u}_j^l. \tag{33}$$

$\mathbf{p}_{ij}^k$ is the projection from image j to image i of the point corresponding to $\mathbf{u}_i^k$.

In (32) the function $h()$ can be Euclidean distance operator, as in [54], and be solved by Gauss-Newton algorithm, or can be a robust error function on distance, as in [8], and be solved by Levenberg-Marquardt algorithm. Tiggs et.al. [57] reviewed various aspects of bundle adjustment, including the choosing of error metrics, motion models and optimization methods.

Sometimes after we have optimized the camera matrix and focal length for each image, we may still find the resulting stitched image blurry or ghosted in some places. This can be caused by unmodeled lens distortion, 3D parallax (failure to rotate the camera around its optical center), small scene motions such as waving tree branches, and large-scale scene motions such as people moving in and out of the scene. Each of these problems can be attacked by some specific approaches. For example, radial distortion of camera lens can be modeled using complex models; 3D parallax can be reduced by doing a full 3D bundle adjustment, which means using a camera motion model including rotation and translation, in stead of pure rotation model generally used. The reader can refer to [54] for related methods to solve these problems.

3.1.4 Compositing and Blending

Once all the images are aligned together, we need to choose a panorama format, which determines what kind of surface the images are projected to, e.g. flat, cylindrical, spherical, etc.

If there are only small number of images, it's more reasonable to choose one image as reference and project all others to this reference coordinates, witch is a flat surface. Thus all the projections are still perspective projections, where straight lines remain straight.

If there are many images to be stitched and they cover a large field of view, for example 360° panoramas, it's better to use cylindrical or spherical coordinates, otherwise there would be significant distortion in the edge area of the result. The environment mapping formats discussed in section 3.4 can be used, e.g. an equirectangular mapping or a cubic mapping. The projection is done in 2 steps. First we need to convert every pixel in the panorama into a ray (from camera center through the image pixel to the 3D world point). Then these rays are mapped back according to the projection equations for selected surface(s).

Another important step of making a good looking panorama is blending.

If the alignments are accurate and images have the same exposure, pixels on different images depicting the same world point will be exactly the same. Then simply use the pixel from any image or averaging them would be enough to have a good result. For example, in the case of Fig. 7, the 2 images are taken with same exposure settings (shutter speed, aperture and ISO), so just averaging the overlapping pixels results in good blending.

However, if the images are exposed differently, the resulting overlapping area will have visible seam or padding like effect, as those in Fig. 8(b). These difference in exposure may be caused by the significant lighting difference in different part of the scene, e.g. parts of the scene are lighted directly by the sun, while other parts are shadows, then keeping the same exposure will result in over-exposure or under-exposure in some images. Another reason may be limitations of camera settings, because many low-end cameras and smartphone cameras have no manual mode and their automatic exposure sometimes reply dramatically to small scene changes.

A simple way to solve this problem is to simply perform a weighted averaging in the overlapping areas [54] [8],

$$I(\mathbf{x}) = \frac{\sum_i I_i(\mathbf{x}) W_i(\mathbf{x})}{\sum_i W_i(\mathbf{x})} \tag{34}$$

where $I_i(\mathbf{x})$ are the re-projected images and $W_i(\mathbf{x})$ the weight.

Generally the weight is chosen so that pixels near the center of an image has higher importance than pixels near edges. For example, $W_i(\mathbf{x}) = w(x)w(y)$, and both $w(x)$ and $w(y)$ are set linearly from 1 in the center to 0 in the edge. Or $W_i(\mathbf{x})$ can be set as the Euclidean distance to the nearest invalid pixel, which is called *distance map*.

Brown and Lowe [38] [8] improved this linear blending by *gain compensation* and *multi-band blending*. They multiply a gain g_i to each image I_i, and minimize the average intensity error in overlapping areas:

$$e = \frac{1}{2} \sum_i \sum_j (g_i \bar{I}_i(\mathcal{R}(i,j)) - g_j \bar{I}_j(\mathcal{R}(j,i))) \tag{35}$$

where $\mathcal{R}(i,j)$ demotes the points on image i that overlap with image j; and $\bar{I}_i(\mathcal{R}(i,j))$ represents the mean intensity of image i on its overlapping area with image j.

Then the blending is done on different frequency bands, in order to blend low frequencies over a large spatial range, and high frequencies over a short range. They applied Gaussian functions with different deviations to filter the image in that certain frequency band. And average images in each band with a band specific weighting function. Average images in different bands are combined together to have the final result.

Fig. 8 gives an example of how gain compensation and multi-band blending help reduce exposure differences.

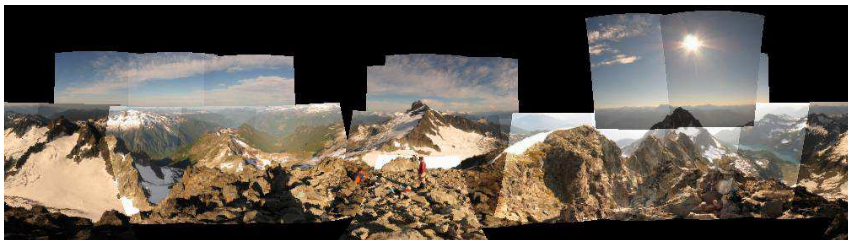
3.2 Omnidirectional camera

To make the panoramic imaging process more convenient, various hardware approaches have been proposed, since the beginning of photography. Benosman and Kang [5] reviewed the history of panoramic imaging hardware, related theory and their applications. A camera that directly produce panorama are often referred as *omnidirectional camera*.

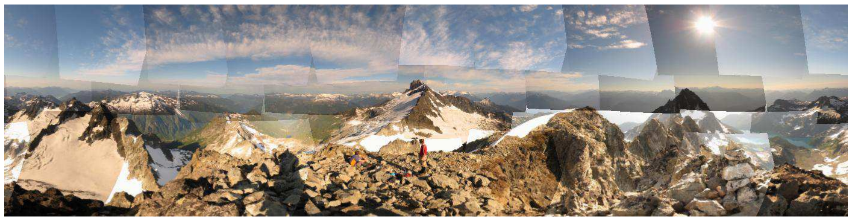
One simple way is to use more cameras to cover a larger field of view. We can arrange multiple cameras on a spherical surface, e.g. the one shown in Fig. 9(c). Then image stitching techniques are applied to align these views together. Note that only one pre-calibration is needed to estimated stitching parameters, since the camera locations are fixed. There are several commercial products available in the market using multi-camera solution, which can take panoramic photo and make panoramic video in real time [26] [1].

Another approach to enlarge field of view is to improve the lens. Lens that has wide viewing angle is usually curvilinear (as opposite to rectilinear), i.e. straight lines appear distorted. Fig. 9(a) shows a fisheye lens manufactured by Nikon, which provide a hemispheerical (180°) field of view.

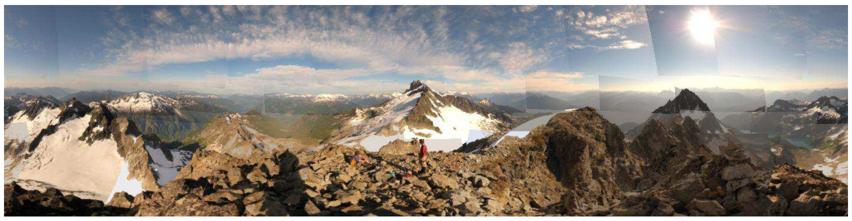
We can also use a convex mirror in front of a regular camera to get large field of view in the back of the camera. This is called a *Catadioptric camera*. And this approch originated from and are most applied in robot navigation applications. Baker and Nayar [4] analyzed the optic geometry and proved that



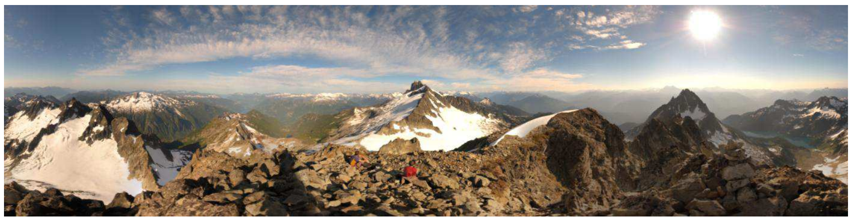
(a) Half of the images registered



(b) Without gain compensation



(c) With gain compensation



(d) With gain compensation and multi-band blending

Figure 8: Example of gain compensation and multi-band blending [8]

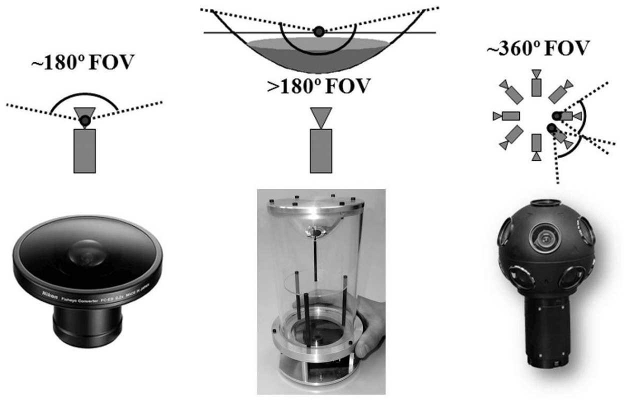


Figure 9: (a) A fisheye lens); (b) catadioptric camera; (c) multi-camera system [48] [26]

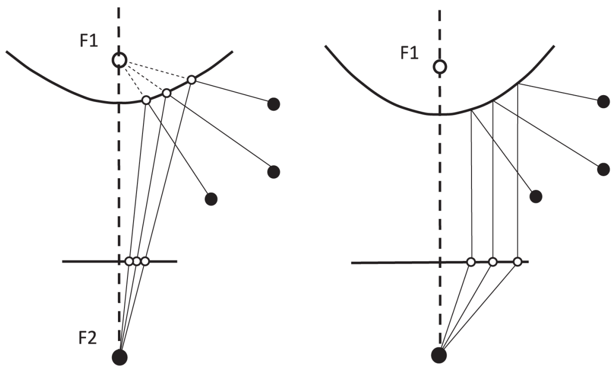


Figure 10: Central catadioptric cameras can be built by (a) hyperbolic mirror and perspective camera; (b) parabolic mirror and orthographic camera [48]

the mirror need to be certain conic surfaces, i.e. hyperbolic, parabolic and elliptical, in order to achieve *single effective viewpoint* property for the system. A vision system is *central* when the optical rays intersect in a single point, which is called *single effective viewpoint* property [4] [48]. This property is important because it enables mapping back to a perspective image. Fig. 9(b) and Fig. 10 shows examples of catadioptric camera geometry.

The images aquired from fisheye or catadioptric cameras can be re-mapped to normal perspective images. The user can refer to [48] for a brief review of how to calibrate these cameras. Fig. 11 shows an example of images from ctadioptric and fisheye cameras.

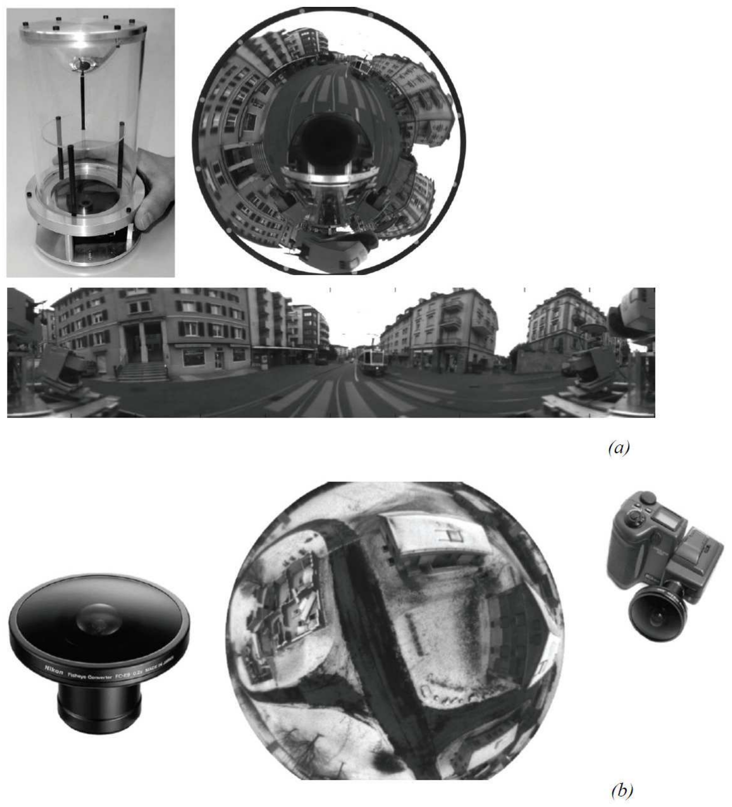


Figure 11: (a) A catadioptric omnidirectional camera using a hyperbolic mirror. The image is typically unwrapped into a cylindrical panorama. The field of view is typically 100 degrees in elevation and 360 degrees in azimuth. (b) Nikon fisheye lens FC-E8. This lens provides a hemispherical (180 degrees) field of view. [48]

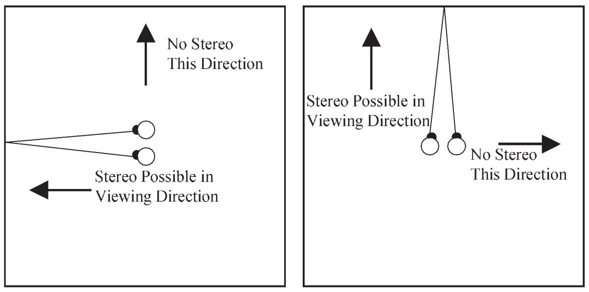


Figure 12: Two single-viewpoint images can only give stereo in one direction. [45]

3.3 Stereo Panorama

Previously introduced panoramic imaging techniques only take care of monoscopic panorama. This section will introduce methods for making stereo panoramas.

In order to create stereoscopic image, we need 2 separate viewpoints for both eyes. Human's perception of depth comes from *disparity*, which is the angular difference in viewing directions of the same point between 2 eyes. However, the arrangement of viewpints determines the viewing direction. As illustrated in Fig. 12, the direction parallel to the *baseline* has no stereo. The baseline is the line passing 2 viewpoints. So creating 2 panoramas at 2 fixed viewpoints can not create omnidirectional stereo.

3.3.1 Stereo Panorama from 2 cameras

One way to make stereo panorama is to rotate 2 cameras simultaneously, just like the movement of 2 eyes when we rotate our head. In this case, the viewpoints lie on a circle, as the case in Fig. 13.

Several works [36] [25] arranged the rotation axis at the optical center of one camera, as shown in Fig. 14. In this way, the panorama at the rotation axis can be stitched together using simple rotation model introduced previously.

However, one can imagine that motion models for the outer camera in Fig. 14 and for both camera in Fig. 13(a) involve translation and rotation at the same time. Actually these cameras' motion is complex so that their images can not be projected to a common surface to form a panorama [45], as in the fixed viewpoint case. Especially when an object is near the camera, the translation of camera position will make that object appear in different size among different images, while the faraway background in similar size. The reader can refer to FIG. 8 in [25] for detailed description. Moreover, stereo vision puts high requirements on the alignment and viewing direction of left/right images.

This problem can be addressed from 2 aspects. Huang and Hung [25] pro-

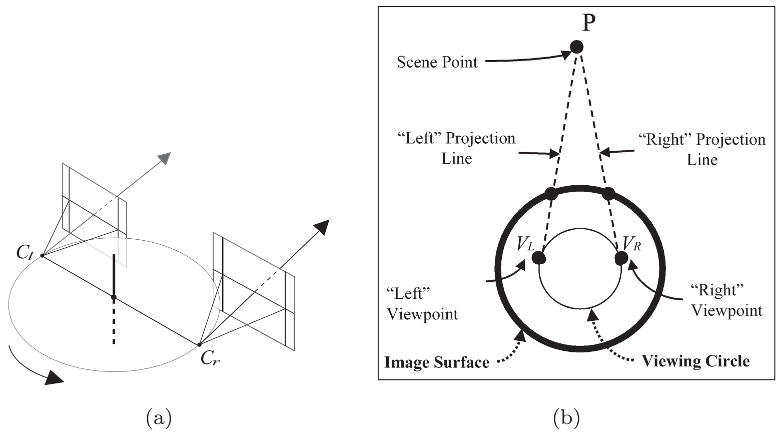


Figure 13: Stereo panorama by rotating 2 cameras. (a)Rotation axis at the center of baseline [29]. (b) Rotation axis slightly behind baseline [45]

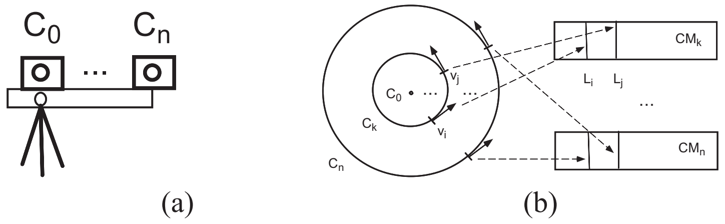


Figure 14: The so called *concentric panorama* [36]. By putting one camera on the rotation axis, the stitching problem at that viewpoint can be simplified.

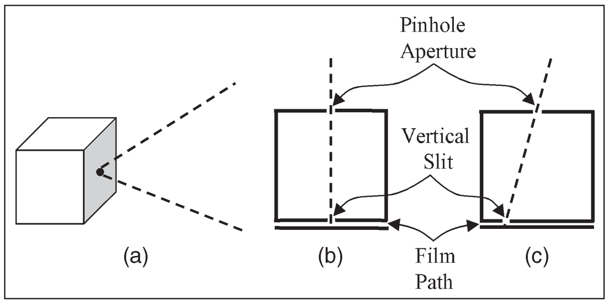


Figure 15: Two models of slit cameras. (a) Side view. (b) and (c) Top view from inside. The locations of aperture and slit are fixed. (b) A vertical slit at the center gives a viewing direction perpendicular to image surface. (c) A vertical slit at the side gives a tilted viewing direction. [45]

posed a disparity warping method to adjust the overlapping areas by locating near objects in images. They claimed their system can generate realistic 360° cylindrical panoramas.

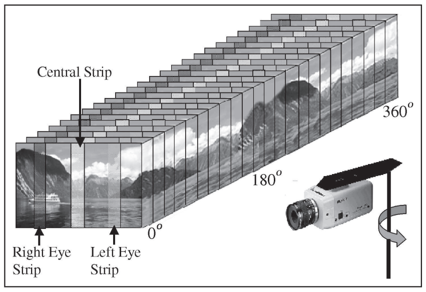
The second approach is to use only a small part in each image, to reduce the inconsistency of different pixels. Generally a *slit* is used to create panorama [36] [29]. A slit is a narrow area that contains few columns (This concept will be illustrated later). As the cameras moves, pixels in a narrow tends to be more consistent, without sudden change in the size of objects. This requires the rotation step to be smaller so that there are enough overlapping among slits.

3.3.2 Stereo Panorama from single camera

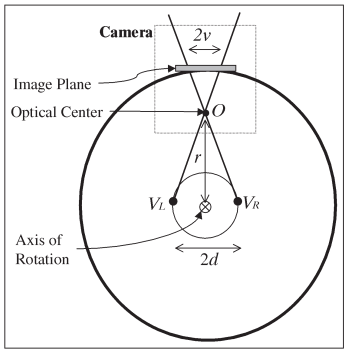
Another popular technique can make stereo panorama with only a single camera, which is modeled as *slit camera*. [45] [11]. As shown in Fig. 15, a slit camera have only column like sensors (an introduction of hardware slit camera can be found in [24], where it's called *rotating sensor-line camera*), or can be simulated by using a narrow slit of pixels from an ordinary camera.

When a slit camera is rotating around an axis behind its optical center, as in the case of Fig. 16, images acquired from the left slit can be regarded as the projections to the right viewpoint denoted as V_R in Fig. 16(b), while the right slit as left viewpoint projection. And both projections lie on the same circular surface.

By combining together narrow strips together, one can get a pair of cylindrical panoramas to form stereo vision. An advantage of this technique is that by selecting slits at different distance from image center, we can control the disparity angle even after the image acquisition.



(a)



(b)

Figure 16: Creating stereo panorama using slit camera model [45].

3.3.3 Remaining Problems

One problem with all the previous described techniques is that they cannot record dynamic scenes, i.e. the scene they are recording must stay the same when the camera(s) rotating. There are approaches to make realtime stereos panoramic images [18] [21], the hardwares of which are shown in Fig. 17. But operating in realtime requires the viewpoint to be fixed. Therefore the baseline of these systems has to be vertical in order to have a full horizontal view. This arrangement can be used in depth estimation, 3D reconstruction and robot navigation, but not useful for visualization, because the baseline of human vision is horizontal. Peleg et.al. [45] proposed the using of spiral mirror or spiral lens together with fixed viewpoints panoramic camera to achieve stereo. However, that remains a theoretical proposal which has never come true.

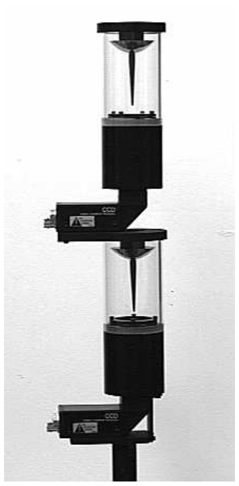
Another disadvantage of previous techniques is that their baselines are always horizontal. When viewing in front of big stereo displays, people may want to tilt their head sometimes. As illustrated in Fig. unknown, when the user's eyes are not on the same level, the cylindrical stereo panoramas can not give correct rendering.

Also, the vertical viewing angle cannot cover a full 180° range for the existing methods.

3.4 Formats of Panorama

The format of a panorama is determined by the projection used to map a full 360° or partial scene onto a 2 dimensional print or screen. This is similar with the term *environment mapping* in computer graphics, which means representing a scene by projecting the environment onto a map [20].

For a full 360° field of view, we need to project the scene onto a closed surface, for example, a cube or a sphere. This leads to two most popular panorama


(a)

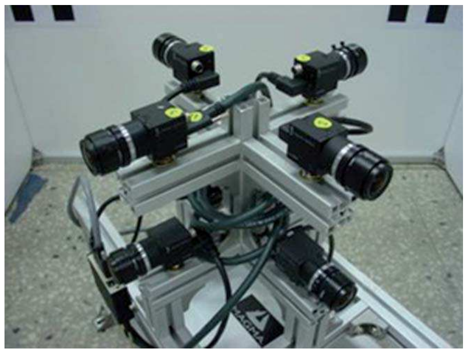

(b)

Figure 17: Realtime stereo panorama systems exist, but their baselines are vertical, in order to keep the viewpoints fixed. These systems are not aimed at visualization, but often at 3D reconstruction and robot navigation. (a) 2 vertically arranged catadioptric cameras [18]. (b) multiple camera system [21].

formats:

Equirectangular

Image stitching process often uses spherical coordinates to make full view panoramas. In order to store and display conveniently on computer screens, the spherical projection is often re-mapped to an rectangular image, which is called *equirectangular* format. Its aspect ratio is $2:1$, with the horizontal coordinate representing the azimuth angle $\theta \in 0^\circ \sim 360^\circ$, and the vertical coordinate representing the elevation angle $\phi \in -90^\circ \sim 90^\circ$. This is the same way that the earth surface is printed in world maps.

Fig. 18 shows an equirectangular panorama of CIM2 lab.

Cubic

The cubic format uses 6 cube faces as the projection surfaces. This often involes a reprojection step from the spherical panorama made by image stitching systems. Fig. 19 shows an example of the scene of CIM2 lab in cubic projection.

"Little planet"

"Little planet" [44] is an unusual format that uses fisheye projection model to map the spherical environment map. The user can refer to [44] and [43] for details.

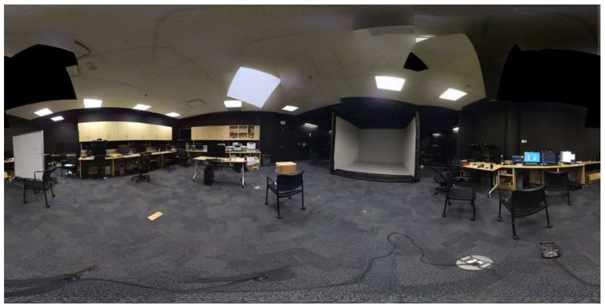


Figure 18: An example of equirectangular panorama.

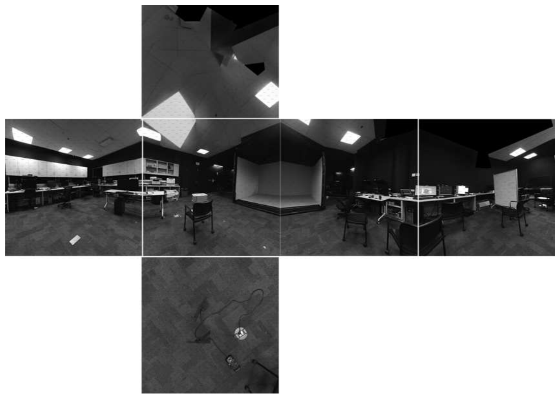


Figure 19: An example of cubic panorama.

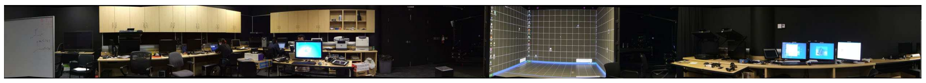


Figure 20: A cylindrical panorama with 360° horizontal and about 70° vertical field of view.

Rectilinear

Rectilinear panorama is just like image taken from an ordinary camera, but with slightly larger field of view. Straight lines stay straight. If field of view is too large the image suffer from unnatural looking distortions in the corners. Therefore the horizontal and vertical field of view are usually limited to about 120° [44].

Cylindrical

Cylindrical is one of the most commonly used panorama formats with partial field of view. It is easy to store cylindrical panorama in 2D image without remapping. Fig. 20 shows a cylindrical panorama of CIM2.

4 Rendering from Images

Rendering means viewing the world from any point in any direction. Once we have made a full view panorama, we can render it on various kind of displays, but the viewpoint is fixed. To create images from new viewpoints, we need to do interpolation.

This section will discuss the rendering of panorama, and image interpolation for the rendering of new viewpoints.

4.1 Single Viewpoint (Rendering Panorama)

Greene [20] argued that projection onto a cube is the best general purpose representation, from computer graphics point of view. In addition, a cubic projection is in 2 dimensional format and is easy to store and process as normal images. And the spherical projection need remapping to equirectangular format to be stored, and mapped back to spherical surface while rendering.

The rendering of both formats on a 2D surface (e.g. a regular screen) involves re-projecting. As shown in fig. 21, the user specifies a viewing direction and angle of view, which determine a viewing cone. Then intersection area of this cone with the panorama is projected to the viewpoint, usually at the center of panorama. This kind of rendering has became so popular that we can find in many commercial products, such as the famous Google StreetView. There are also a number of softwares that can store and render panoramas in a web browser or a video player.

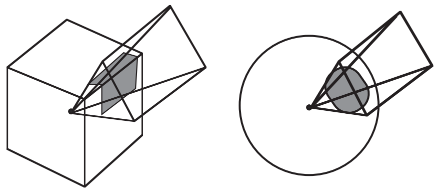


Figure 21: Rendering of cubic and spherical panorama on 2D surface. [12]

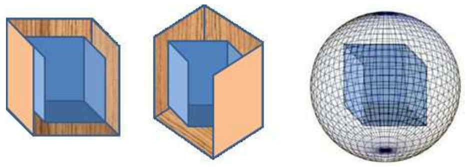


Figure 22: Rendering in CAVE: (a)(b) cubic mapping; (c) spherical mapping [33].

When rendering in a CAVE, spherical format might be a better way than cubic panorama, in terms of the ease of implementation. For example, when the environment map cube does not coincide with with the CAVE surfaces, as the case in Fig. 22(b), re-projection at edges and corners involves complex ray computation, and might also cause distortion. While the re-projection of spherical panorama is more uniformly distributed.

Lee et.al. [33] rendered a stereo panorama image in the CAVE, by applying the image as a texture map inside a sphere surrounding the CAVE, as illustrated in Fig. 22(c). They use the panorama as a more realistic background for a virtual scene.

4.2 Image Interpolation for New Viewpoints

While the sampling and reconstruction of plenoptic function at one fixed viewpoint, i.e. monoscopic panoramic imaging, has been well researched and solved, the reconstruction at arbitrary new viewpoints is still not perfectly solved. However, the state of the art in this area can generate new viewpoints in a predefined limited range, and the result is quite impressing.

Chen's view interpolation [12] [13] is one of the early works trying to attack this problem. He discussed the possibility of a system that collects panoramas in a grid structure to represent the whole scene, as shown in Fig. 23. They implemented a simple view interpolation method to generate image at a new viewpoint using nearby sampled images, and the sampled image should be in same direction. Their interpolation was done by estimating the optical flow between the 2 near by images, and linearly computing a new image coordinate for each pixel. McMillan and Bishop [41] proposed similar method, except that they used cylindrical panoramas instead of single image.

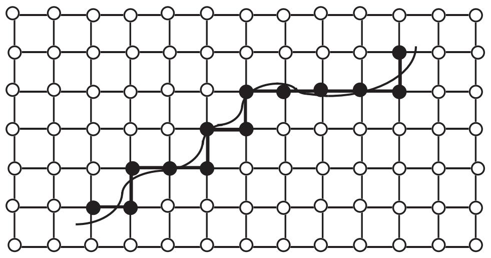


Figure 23: Chen [12] proposed creating a panorama at each node, and generating arbitrary viewpoints and direction.

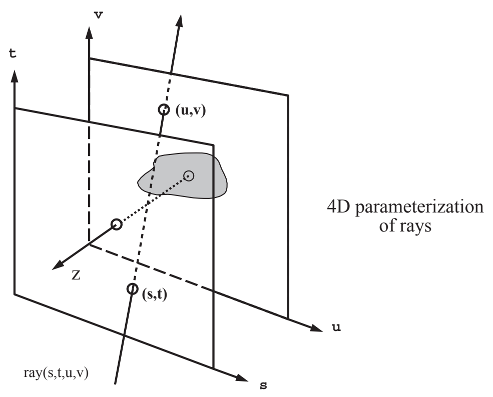


Figure 24: 4D parameterization of rays [19].

4.2.1 Light Field Technology

The concept of light field was introduced by Levoy and Hanrahan [35], and Gortler et al. [19]. The idea of light field is to simplify the original plenoptic function from 5 parameter to 4 parameter, and use this simplified 4 dimensional function to guide the sampling.

Original plenoptic function (as in (1), but without λ and t) uses 5 parameter to describe light ray. However, under a certain constraint called *free space* (an area with no occlusion [35]), the intensity of a light along its path line remains the same. Therefore, one dimension can be removed from plenoptic function. Light field method parameterize a ray in space by 2 intersection points of this ray with 2 known planes, as illustrated in Fig. 24.

In consistency with the ray parameterization, capturing light field of a scene is often done by a camera array lying on a plane. Fig. 25 gives an example of the synthetic view experiment done by Levoy and Hanrahan. After sampling an array of images on the uv plane, light field can be represented using a set of rays connecting the image plane (uv plane) and the object focus plane (st plane). A

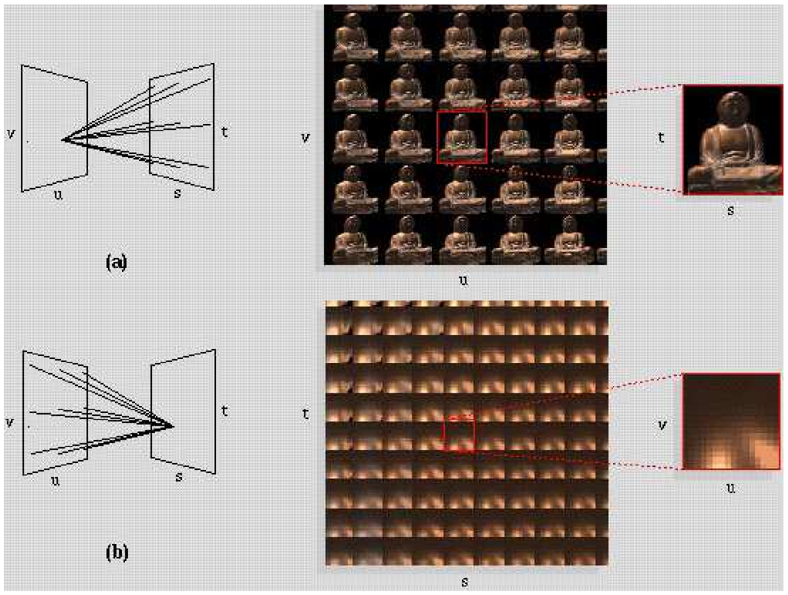


Figure 25: Two visualizations of a light field [35]. (a) Each image in the array represents the rays arriving at one point on the uv plane from all points on the st plane. (b) Each image represents the rays leaving one point on the st plane bound for all points on the uv plane.

new viewpoint on the uv plane can be generated by interpolating nearby light rays.

Fig. 26 shows some arrangement of cameras to capture light field. Large camera arrays can capture a larger range, but it's more expensive and requires accurate camera position and calibration. There are also lens arrays as in Fig. 26 (a) and (b), that can capture a smaller range of light field. Another kind of arrangement is to put a micro-lens array behind the main lens, and in front of the image sensor, as shown in Fig. 27.

Light field technology is similar with another technique called *synthetic aperture* [58] in that they all use multiple cameras or lens arrays to capture multiple samples for later processing. Several commercial light field cameras today [42] take advantage of this technique and feature refocusing to any distance after the image has been taken.

As a side effect of generating new viewpoints, light field cameras can also produce stereo images [31], by generating viewpoints at the location of 2 eyes.

4.2.2 Free Viewpoint TV

Free Viewpoint TV is another system that simplifies the plenoptic function to 4 dimension [56] [17], and has also been in practical usage. Fujii and Tnimo-

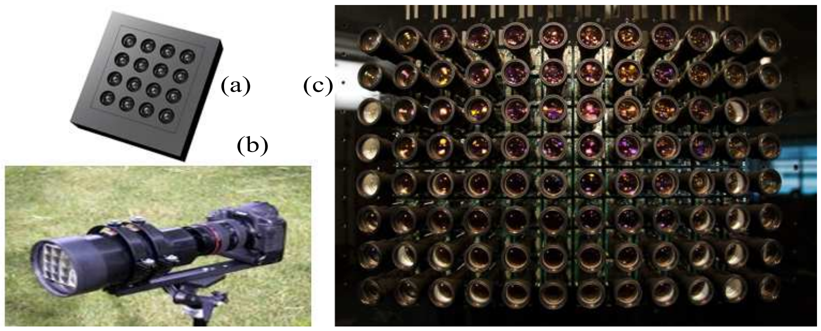


Figure 26: Different kinds of light filed hardware. (a) A lens array for cellphone cameras and webcams. (b) Adobe's plenoptic lens mounted on a DSLR camera. (c) A camera array in Stanford University.

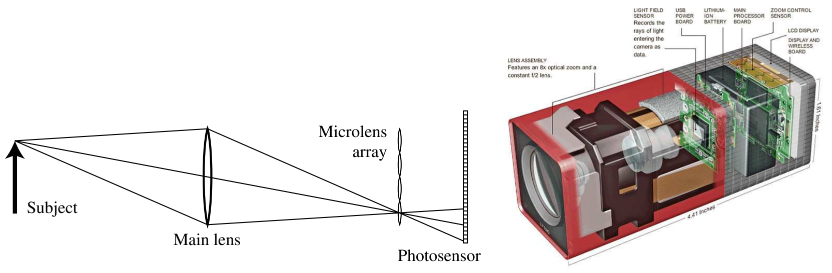

Microlens
array
Subject
Main lens
Photosensor
LIGHT FIELD SENSOR Records the rays of light entering the camera as data.
USB POWER BOARD
LITHIUM-ION BATTERY
MAIN PROCESSOR BOARD
ZOOM CONTROL SENSOR
LCD DISPLAY
DISPLAY AND WIRELESS BOARD
LENS ASSEMBLY Features an 8x optical zoom and a constant f/2 lens.
1.61 Inches
4.41 Inches

Figure 27: Another kind of light field camera that put microlens array behind the main lens [42]. (a) Conceptual schematic. (b) A real product.

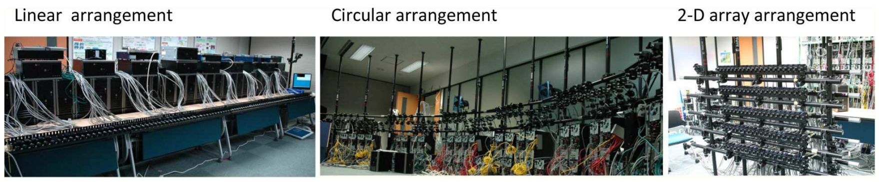


Figure 28: In addition to the planar camera array of light field, FTV also support circular array, because FTV can use a spherical ray parameterization. [56]

to argued that representing a scene in 4D *ray space*, which is the underlying methods in FTV, is a generalization of similar method such as light field. If 4D ray is represented by its intersection with a plane and its direction, which is called *orthogonal ray-space* in [17], it is equivalent to the light field representation. FTV also defines another parameterization, the *spherical ray-space*, that represents the ray by its intersection with a plane tangent to a unit sphere, and perpendicular with the ray.

Therefore, to capture the ray space, FTV can use both planar camera array and cylindrical/spherical camera array. Fig. 28 gives some examples on the arrangement of cameras in ray space capturing.

4.2.3 Applications

Though light field and FTV technology has been developed for practical use and achieves impressing results, their generated new viewpoint is only in one direction does not cover 360° field of view.

However, this kind of interpolation is still useful in various applications. For the rendering of relatively small objects, generally we do not need large viewing angle, only the directions that cover the interested object are needed. Besides, we can arrange the camera array in a circular arrangement, which can cover the whole 360° surrounding of the object. This could be helpful in product advertising that gives a full view of the product. Actually there are already commercial solutions to model small objects from photos taken in a circular manner [14] [15].

For larger scale scenes, for example urban areas and cultural heritage like palaces, we often want a full 360° view, and panorama is needed. However, there is no good solutions to interpolate new viewpoints with full field of view. This is why this kind of image based rendering can only render at sample points, as done by Google StreetView.

4.3 Combine Geometry and Image based Rendering

Chan, Shum and Ng [10] reviewed the development and major issues in Image Based Rendering up to 2007. They classified image-based representations into

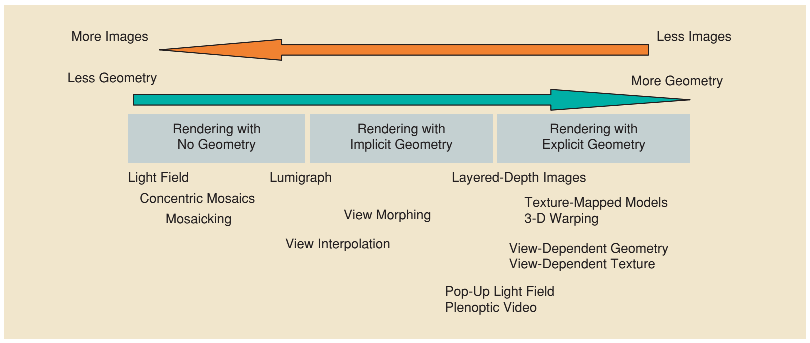


Figure 29: Spectrum of IBR representations [10]

three main categories, as shown in Fig. 29. Light field, and various panoramic imaging techniques, fall into the first category, representation with no geometry.

Moving to the right side of the spectrum in Fig. 29, geometry information starts to take part in the scene representation. For example, in Free Viewpoint TV and Pop-up Light Field [50], depth information is needed to help processing. Therefore they fall into the second category, rendering with implicit geometry.

To learn geometry from images has always been an interested area, since the beginning of computer vision. A number of techniques have been proposed to estimate depth from multiple cameras.

Augmented Reality (AR), which is a popular research topic these years, can also be regarded as a combination of image and geometry based rendering. Because it involves learning camera motion from image, and then render the image together with predefined geometry. However, a new kind of AR can be imagined if we use pure image based representation, instead of geometry based. Chen et.al. [11] proposed a system that renders a virtual object in a panoramic scene. That virtual object, at the same time, is also represented by images, captured by a spherical arranged camera array (they are using one camera in front of a rotatable rig to hold the object, which is equivalent to a spherical camera array).

On the other hand, images are also helping geometry based rendering to become more realistic. Texture mapping has became a basic function of graphic hardware today. In many 3D modeling systems, real images are often segmented and mapped to a 3D model. Also, some pure geometry based rendering applications, especially in 3D gaming industry, are applying image representations for some complex scenes where geometry based rendering suns slowly.

5 Conclusion and Possible Future Work

The literature review of this paper covers several different aspects of image based rendering.

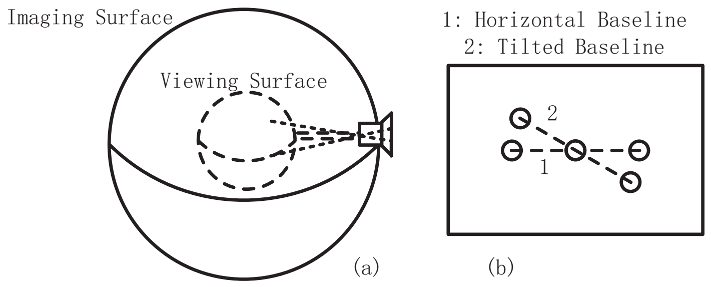


Figure 30: Proposed method for stereo panorama. (a) Rotating a camera on a spherical surface, while the rotation center behind the camera. (b) Use small areas to construct panoramic view for left and right eyes, with different disparity and baseline.

In my own opinion, the goal of rendering and visualization is always the same, i.e. to expose people to scenes that are different from his/her current surroundings, and make people's perception of this scene as realistic as possible. No matter the scene is a totaly virtual one depicting the imagination of artists, or a real one at somewhere far away, no matter what kind of technology it uses, geometry based or image based, the ultimate goal of enriching human perception remains the same. How to collect, store, construct or process data to represent the scene; How to render these data and let people "see", "feel" and even "interact" with the scene. These are 2 fundamental issues.

In the following section, 3 possible ideas are proposed. Due to limited time, these possibilities are only described briefly. The reader will be referred to previous sections for related topics and details.

5.1 Full View Stereo Panorama

Recall that in section 3.3.3, we discussed several problems of stereo panoramic imaging. One of these problem is the lack of full view spherical support, while all existing systems only take cylindrical stereo panorama and use horizontal baseline.

To solve these two problems, we can take an analogy of existing cylindrical method introduced in section 3.3.2, that use only several columns of an image to construct panorama, in order to achieve left and right eye viewpoints.

In the same manner, a spherical rotation scheme is proposed, as shown in Fig. 30. A camera is rotated on a spherical surface, with the rotation center behind the cemera. Similar to the "*slit*" concept, we only use 2 "holes" on the image (as those in Fig. 30(b)), which may only consists of a circle of several pixels, to construct a panorama for each eye. The distance between these 2 holes, together with the rotation radius, determines the disparity. This method will also support tilted baselines, as illustrated in Fig. 30(b).

One difficulty to implement this method is the requirement of accurate lo-

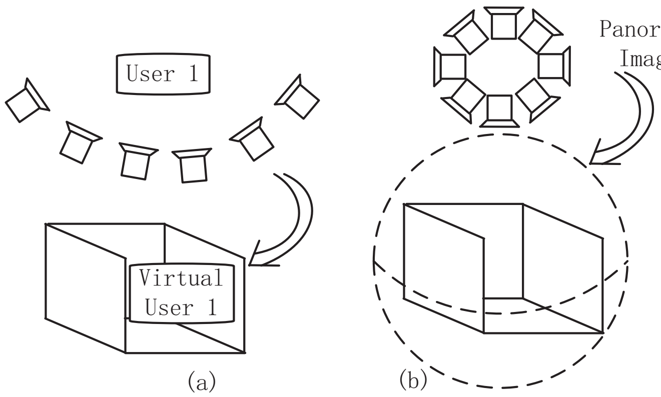


Figure 31: Image based Rendering in CAVE. (a) Application in Telecommunication. (b) Application in Virtual Tour.

cation of camera, so simulation on a synthetic scene should be done first to validate the feasibility. Images should be taken with very small rotation step, to ensure overlapping within the holes.

Another approach would be an analogy of generating stereo viewpoints from a planar light field representation, as introduced in the last paragraph in section 4.2.1. We can use spherical 4D ray-space to represent a scene, and generate new viewpoints according to the eye location for stereo vision. However, the capturing process would be quite similar. For example, we can use a spherical camera array to capture the scene (as illustrated in Fig. 31(b)), which is equivalent to rotating a single camera on a spherical surface.

5.2 Image-based Telecommunication

With the CAVE's huge advantage in immersive visualization, it might be possible to produce a realistic rendering experience using image based representation.

The first possible application would be telecommunication. As shown in Fig. 31(a), a camera array could be used to capture light field or 4D ray-space of a user's upper body. This scene is then send back and rendered to another user in CAVE.

Though various techniques introduced in section 4.2 can be used here, and recent advances in this area have shown good result of generating new viewpoints for rendering [50] [31], a main challenge to this proposal would be the implementation of one such interpolation algorithms. Then segmentation is needed to render the object in front of a possible new scene background.

5.3 Image-based Virtual Tour System

The second possible CAVE application would be a virtual tour system (Fig. 31(b)), like a streetview in CAVE. Since current view interpolation can not support full viewing angle, as discussed previously, such systems generally can only render at sample points. In order to produce better user experience, smaller sample interval should be used.

Monoscopic panoramic imaging is a mature technology. Therefore it's easy to adopt to CAVE. However, to have stereo, first a stereo panoramic imaging technique should be available. Those discussed in section 5.1 might be a reference.

References

[1] 360Heros. www.360heros.com.

[2] E. H. Adelson and J. R. Bergen. The plenoptic function and the elements of early vision. In *Computational models of visual processing*, pages 3–20. The MIT Press, 1991.

[3] S. Baker and I. Matthews. Lucas-kanade 20 years on: A unifying framework. *International Journal of Computer Vision*, 56(3):221–255, 2004.

[4] S. Baker and S. K. Nayar. A theory of single-viewpoint catadioptric image formation. *International Journal of Computer Vision*, 35(2):175–196, 1999.

[5] R. Benosman and S. B. Kang. *Panoramic vision: sensors, theory, and applications*. Springer, 2001.

[6] J. R. Bergen, P. Anandan, K. J. Hanna, and R. Hingorani. Hierarchical model-based motion estimation. In *Computer VisionECCV'92*, pages 237–252. Springer, 1992.

[7] M. Brown. Autostitch. www.cs.bath.ac.uk/brown/autostitch/autostitch.html.

[8] M. Brown and D. G. Lowe. Automatic panoramic image stitching using invariant features. *International Journal of Computer Vision*, 74(1):59–73, 2007.

[9] P. Burt and E. Adelson. The laplacian pyramid as a compact image code. *Communications, IEEE Transactions on*, 31(4):532–540, 1983.

[10] S. Chan, H.-Y. Shum, and K.-T. Ng. Image-based rendering and synthesis. *Signal Processing Magazine, IEEE*, 24(6):22–33, 2007.

[11] C.-W. Chen, L.-W. Chan, Y.-P. Tsai, and Y.-P. Hung. Augmented stereo panoramas. In *Computer Vision–ACCV 2006*, pages 41–49. Springer, 2006.

[12] S. E. Chen. Quicktime vr: An image-based approach to virtual environment navigation. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 29–38. ACM, 1995.

[13] S. E. Chen and L. Williams. View interpolation for image synthesis. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 279–288. ACM, 1993.

[14] Creative Dimension Software Ltd. 3DSOM Image-based 3D modeling. www.3dsom.com.

[15] Eos Systems Inc. PhotoModeler Image-based 3D modeling. www.photomodeler.com.

[16] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[17] T. Fujii and M. Tanimoto. Free viewpoint tv system based on ray-space representation. In *ITCom 2002: The Convergence of Information Technologies and Communications*, pages 175–189. International Society for Optics and Photonics, 2002.

[18] J. Gluckman, S. K. Nayar, and K. J. Thoresz. Real-time omnidirectional and panoramic stereo. In *Proc. of DARPA Image Understanding Workshop*, volume 1, pages 299–303, 1998.

[19] S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M. F. Cohen. The lumigraph. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 43–54. ACM, 1996.

[20] N. Greene. Environment mapping and other applications of world projections. *Computer Graphics and Applications, IEEE*, 6(11):21–29, 1986.

[21] J.-E. Ha and I.-S. Choi. Simple method for calibrating omnidirectional stereo with multiple cameras. *Optical Engineering*, 50(4):043608–043608, 2011.

[22] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[23] A. S. Huang, A. Bachrach, P. Henry, M. Krainin, D. Maturana, D. Fox, and N. Roy. Visual odometry and mapping for autonomous flight using an rgb-d camera. In *Int. Symposium on Robotics Research (ISRR),(Flagstaff, Arizona, USA)*, 2011.

[24] F. Huang and R. Klette. Stereo panorama acquisition and automatic image disparity adjustment for stereoscopic visualization. *Multimedia Tools and Applications*, 47(3):353–377, 2010.

[25] H.-C. Huang and Y.-P. Hung. Panoramic stereo imaging system with automatic disparity warping and seaming. *Graphical Models and Image Processing*, 60(3):196–208, 1998.

[26] Immersive Media. immersivemedia.com.

[27] L. Iwerks. Industrial light & magic: Creating the impossible. Documentary TV Movie. 2010.

[28] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, et al. Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pages 559–568. ACM, 2011.

[29] W. Jiang, M. Okutomi, and S. Sugimoto. Panoramic 3d reconstruction using rotational stereo camera with simple epipolar constraints. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 371–378. IEEE, 2006.

[30] S. B. Kang. Survey of image-based rendering techniques. In *Electronic Imaging'99*, pages 2–16. International Society for Optics and Photonics, 1998.

[31] C. Kim, A. Hornung, S. Heinzle, W. Matusik, and M. Gross. Multi-perspective stereoscopy from light fields. In *ACM Transactions on Graphics (TOG)*, page 190. ACM, 2011.

[32] H. Lee, Y. Tateyama, and T. Ogi. Realistic visual environment for immersive projection display system. In *Virtual Systems and Multimedia (VSMM), 2010 16th International Conference on*, pages 128–132. IEEE, 2010.

[33] H. Lee, Y. Tateyama, and T. Ogi. Image-based stereo background modeling for cave system. In *VR Innovation (ISVRI), 2011 IEEE International Symposium on*, pages 251–254. IEEE, 2011.

[34] b. Leibe. Lecture on computer vision. CV group website at RWTH-AACHEN.

[35] M. Levoy and P. Hanrahan. Light field rendering. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 31–42. ACM, 1996.

[36] Y. Li, H.-Y. Shum, C.-K. Tang, and R. Szeliski. Stereo reconstruction from multiperspective panoramas. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(1):45–62, 2004.

[37] S. Livatino, V. Agerbech, A. Johansen, and B. Johansen. Designing a virtual reality game for the cave. In *Eurographics Italian Chapter Conference*, pages 111–115. Citeseer, 2006.

[38] D. Lowe. Recognizing panoramas. In *Proc. of CVPR*, 2003.

[39] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[40] B. D. Lucas, T. Kanade, et al. An iterative image registration technique with an application to stereo vision. In *IJCAI*, volume 81, pages 674–679, 1981.

[41] L. McMillan and G. Bishop. Plenoptic modeling: An image-based rendering system. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 39–46. ACM, 1995.

[42] R. Ng, M. Levoy, M. Brédif, G. Duval, M. Horowitz, and P. Hanrahan. Light field photography with a hand-held plenoptic camera. *Computer Science Technical Report CSTR*, 2(11), 2005.

[43] PanoTools Wiki. Fisheye projection. wiki.panotools.org/Fisheye.

[44] PanoTools Wiki. Panorama formats. wiki.panotools.org/Panorama_formats.

[45] S. Peleg, M. Ben-Ezra, and Y. Pritch. Omnistereo: Panoramic stereo imaging. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23(3):279–290, 2001.

[46] R. Raskar, M. S. Brown, R. Yang, W.-C. Chen, G. Welch, H. Towles, B. Scales, and H. Fuchs. Multi-projector displays using camera-based registration. In *Visualization'99. Proceedings*, pages 161–522. IEEE, 1999.

[47] R. Raskar, G. Welch, M. Cutts, A. Lake, L. Stesin, and H. Fuchs. The office of the future: A unified approach to image-based modeling and spatially immersive displays. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 179–188. ACM, 1998.

[48] D. Scaramuzza. Omnidirectional camera. GRASP Lab, University of Pennsylvania.

[49] H.-Y. Shum, S.-C. Chan, and S. B. Kang. *Image-based rendering*. Springer, 2007.

[50] H.-Y. Shum, J. Sun, S. Yamazaki, and Y. Li. Pop-up light field, Dec. 15 2009. US Patent 7,633,511.

[51] H.-Y. Shum and R. Szeliski. Systems and experiment paper: Construction of panoramic image mosaics with global and local alignment. *International Journal of Computer Vision*, 36(2):101–130, 2000.

[52] S. N. Sinha, J. Kopf, M. Goesele, D. Scharstein, and R. Szeliski. Image-based rendering for scenes with reflections. *ACM Transactions on Graphics (TOG)*, 31(4):100, 2012.

[53] T. Svoboda, T. Pajdla, and V. Hlaváč. Epipolar geometry for panoramic cameras. In *Computer Vision-ECCV'98*, pages 218–231. Springer, 1998.

[54] R. Szeliski. Image alignment and stitching: A tutorial. *Foundations and Trends® in Computer Graphics and Vision*, 2(1):1–104, 2006.

[55] R. Szeliski and H.-Y. Shum. Creating full view panoramic image mosaic-s and environment maps. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 251–258. ACM Press/Addison-Wesley Publishing Co., 1997.

[56] M. Tanimoto, M. P. Tehrani, T. Fujii, and T. Yendo. Ftv for 3-d spatial communication. *Proceedings of the IEEE*, 100(4):905–917, 2012.

[57] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon. Bundle adjustmenta modern synthesis. In *Vision algorithms: theory and practice*, pages 298–372. Springer, 2000.

[58] V. Vaish and M. Adviser-Levoy. Synthetic aperture imaging using dense camera arrays. *Doctoral Dissertation*, 2007.

[59] Wikipedia. 35mm equivalent focal length.

[60] Wikipedia. Angle of view. en.wikipedia.org/wiki/Angle_of_view.

[61] Wikipedia. Pinhole camera model. en.wikipedia.org/wiki/Pinhole_camera_model.

[62] Wikipedia. Rectilinear lens. en.wikipedia.org/wiki/Rectilinear_lens.

[63] J. Xiao, T. Fang, P. Zhao, M. Lhuillier, and L. Quan. Image-based street-side city modeling. In *ACM Transactions on Graphics (TOG)*, volume 28, page 114. ACM, 2009.