Chapter 4

Cameras and Images

Goal of Study

1. Learn the models of the pinhole camera, intrinsics, extrinsics, and distortion.
2. Learn how to project a spatial point into image planes.
3. Learn the basic image process in OpenCV.

In the previous two lectures, we introduced how to express and optimize the robot's 6 DoF pose and partially explained the meaning of the variables and the equations of motion and observation in SLAM. This chapter will discuss "How robots observe the outside world", which belongs to the observation equation. In the camera-based visual SLAM, the observation mainly refers to the process of image projection.

We have seen a lot of photos in real life. A photo consists of millions of pixels in the computer, recording information about color or brightness. We will see a bundle of light reflected or emitted by an object in the three-dimensional world pass through the camera's optical center and is projected onto the camera's imaging plane. After the camera's light sensor receives the light, it produces a measurement, and we get the pixels, which form the photo we see. Can this process be described by mathematical equations? This lecture will first discuss the camera model, explain how the projection relationship is described, and the internal parameters in this projection process. At the same time, we will also give a brief introduction to the stereo and RGB-D cameras. Then, we introduce the basic operations of 2D images in OpenCV. Finally, an experiment of point cloud stitching is demonstrated to show the meaning of intrinsic and extrinsic parameters.

4.1 Pinhole Camera Models

The process of projecting a 3D point (in meters) to a 2D image plane (in pixels) can be described by a geometric model. Actually, several models describe this, the simplest of which is called the pinhole model. We will start with this pinhole projection. At the same time, due to the presence of the lens on the camera lens, *distortion* is generated during the projection. Therefore, we will use the pinhole model plus a distortion model to describe the entire projection process.

4.1.1 Pinhole Camera Geometry

Most of us have seen the candle projection experiment in the physics class of high school: a lit candle is placed in front of a dark box, and the light of the candle is projected through a small hole in the dark box on the rear plane. Then an inverted candle image is formed on this plane. In this process, the small hole can project a candle in a three-dimensional world onto a two-dimensional imaging plane. For the same reason, we can use this simple model to explain the camera's imaging process, as shown in Figure 4-1.

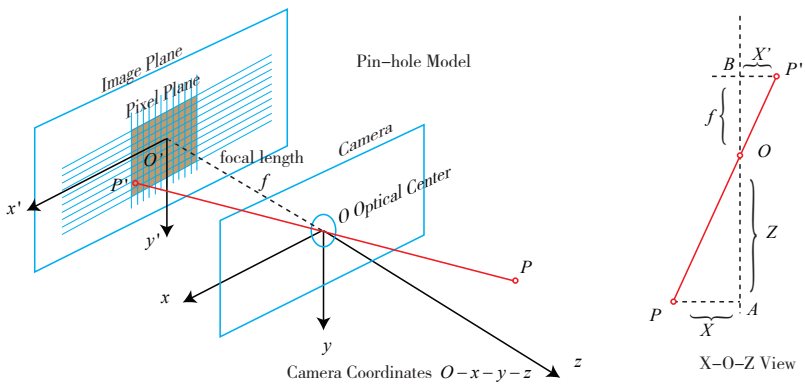

Figure 4-1: Pinhole camera model.

Let's take a look at the simple geometry in this model. Let $O-x-y-z$ be the camera coordinate system. Commonly we put the z axis to the front of the camera, x to the right, and y to the down (so in this figure, we should stand on the left side to see the right side). O is the camera's optical center, which is also the "hole" in the pinhole model. The 3D point P, after being projected through the hole O, falls on the physical imaging plane $O'-x'-y'$ and produces the image point P'. Let the coordinates of P be $[X,Y,Z]^T$, P' is $[X',Y',Z']^T$, and set the physical distance from the imaging plane to camera plane is f (focal length). Then, according to the similarity of the triangles, there are:

$$\frac{Z}{f} = -\frac{X}{X'} = -\frac{Y}{Y'}. \tag{4.1}$$

The negative sign indicates that the image is inverted. However, the image obtained by modern cameras is not inverted (otherwise, the usage of the camera would be very inconvenient). To make the model more realistic, we can equivalently

place the imaging plane symmetrically in front of the camera, as shown by Figure 4-2. This can remove the negative sign in the formula to make it more compact:

$$\frac{Z}{f} = \frac{X}{X'} = \frac{Y}{Y'}. \tag{4.2}$$

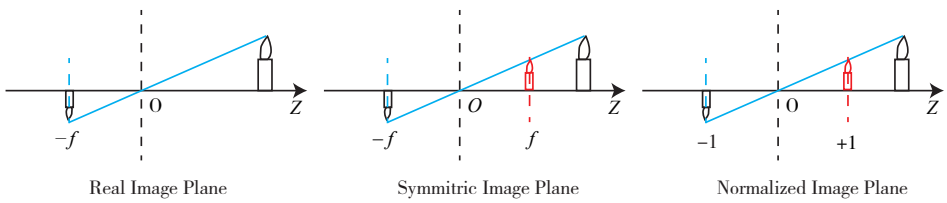

Figure 4-2: The real, symmetric and normalized image plane.

Put X', Y' to the left side:

$$X' = f\frac{X}{Z}, \quad Y' = f\frac{Y}{Z}. \tag{4.3}$$

Readers may ask why we can arbitrarily move the imaging plane to the front? In fact, this is just a mathematical approach to handle the camera projection, and most of the images captured by the camera are not upside-down. The camera's software will flip the picture for you, so what we actually get is the symmetric plane's image. Although the pin-hole image should be inverted from the physical principle since we have pre-processed the picture, it is not wrong to take the symmetric one. Therefore, without causing ambiguity, we often emit the minus symbol in the pin-hole model.

The formula (4.3) describes the spatial relationship between the point P and its image, where the units of all points are meters. For example, a focal length maybe 0.2 meters, and X' be 0.14 meters. However, in the camera, we end up with pixels, where we need to sample and quantize the pixels on the imaging plane. To describe how the sensor converts the perceived light into image pixels, we set a pixel plane $o-u-v$ fixed on the physical imaging plane. Finally, we get *pixel coordinates* of P' in the pixel plane: $[u, v]^T$.

The usual definition of the pixel coordinate system [1] is: the origin o' is in the upper left corner of the image, the u axis is parallel to the x axis, and the v axis is parallel to the y axis. Between the pixel coordinate system and the imaging plane, there is an apparent *zoom* and a *translation of the origin*. We set the pixel coordinates to scale α times on the u axis and β times on v. At the same time, the origin is translated by $[c_x, c_y]^T$. Then, the relationship between the coordinates of P' and the pixel coordinate $[u, v]^T$ is:

$$\begin{cases} u = \alpha X' + c_x \\ v = \beta Y' + c_y \end{cases}. \tag{4.4}$$

Put it into (4.3) and set αf as f_x, βf as f_y:

$$\begin{cases} u = f_x \frac{X}{Z} + c_x \\ v = f_y \frac{Y}{Z} + c_y \end{cases}, \tag{4.5}$$

[1] Or image coordinate system, see section 2 of this lecture.

where f is the focal length in meters, $\alpha, and \beta$ is in pixels/meter, so f_x, f_y and c_x, c_y are in pixels. It would be more compact to write this form as a matrix, but we need to use homogeneous coordinates on the left and non-homogeneous coordinates on the right:

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \frac{1}{Z} \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \triangleq \frac{1}{Z}\mathbf{KP}. \tag{4.6}$$

Let's put Z to the left side as in most books:

$$Z \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \triangleq \mathbf{KP}. \tag{4.7}$$

In this equation, we refer to the matrix composed of the middle quantities as the camera's inner parameter matrix (o intrinsics) $\mathbf{K}$. It is generally assumed that the camera's internal parameters are fixed after manufacturing and will not change during usage. Some camera manufacturers will tell you the intrinsics, and sometimes you need to estimate the internal parameters by yourself, which is called calibration. Because of the maturity of the calibration algorithm (such as the famous Zhang Zhengyou's calibration [25]), it will not be introduced here [2].

There are internal parameters, and naturally, there must be something like "external parameters". In the equation (4.6) , we use the coordinates of P in the camera coordinate system, but in fact, the coordinates of P should be its world coordinates because the camera is moving (we use the symbol $\mathbf{P}_w$). It should be converted to the camera coordinate system based on the current pose of the camera. The camera's pose is described by its rotation matrix $\mathbf{R}$ and the translation vector $\mathbf{t}$. Then there are:

$$Z\mathbf{P}_{uv} = Z \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{K}\left(\mathbf{RP}_w + \mathbf{t}\right) = \mathbf{KTP}_w. \tag{4.8}$$

Note that the latter formula implies a conversion from homogeneous to non-homogeneous coordinates (can you see it?) We use homogeneous coordinates in $\mathbf{TP}$, then convert to non-homogeneous coordinates, and then multiply it by $\mathbf{K}$. It describes the projection relationship of world coordinates to pixel coordinates of P. Among them, the camera's pose $\mathbf{R}, \mathbf{t}$ is also called the camera's *extrinsics*. [3] Compared with the intrinsic, the extrinsics may change with the camera installation and is also the target to be estimated in the SLAM if we only have a camera.

The projection process can also be viewed from another perspective. The formula (4.8) shows that we can convert a world coordinate point to the camera coordinate system first and then remove the last dimension. The depth of the point from the imaging plane of the camera is then removed, which is equivalent to the normalization on the last dimension. In this way, we get the projection of the point P on the camera *normalized plane*:

$$\left(\mathbf{RP}_w + \mathbf{t}\right) = \underbrace{[X, Y, Z]^T}_{\text{Camera Coordinates}} \rightarrow \underbrace{[X/Z, Y/Z, 1]^T}_{\text{Normalized Coordinates}}. \tag{4.9}$$

[2] I'm sure professor Zhang has a copy of this book now.

[3] In robots or autonomous vehicles, extrinsics is often defined as the transform between the camera coordinate system and the robot body coordinate system, describing "where the camera is installed".

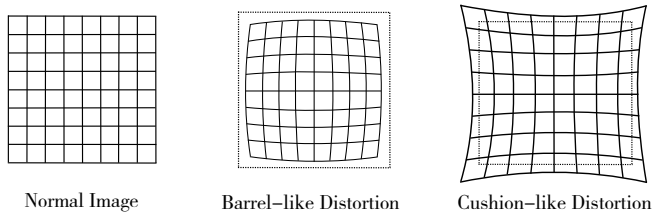

Figure 4-3: The radical distortion.

The *normalized coordinates* can be seen as a point in the $z = 1$ plane in front of the camera [4].This $z = 1$ plane is also called *normalized plane*. We normalize the coordinates and then multiply them with the intrinsic matrix, yielding the pixel coordinates. We can also consider the pixel coordinates $[u, v]^T$ as the result of quantitative measurements on points on the normalized plane. If the camera coordinates are multiplied by any non-zero constant simultaneously, the normalized coordinates are the same, which means that the depth is lost during the projection process. So, in monocular vision, the pixel's depth value cannot be obtained by a single image.

4.1.2 Distortion

In order to get a larger FoV (Field-of-View), we usually add a lens in front of the camera. The addition of the lens has an influence on the propagation of light during imaging: (1) the shape of the lens may affect the propagation way of light, (2) during the mechanical assembly, the lens and the imaging plane are not entirely parallel, which also changes the projected position.

There are some mathematical models to describe the distortion caused by the shape of the lens. In the pinhole model, a straight line keeps straight when projected onto the pixel plane. However, in real photos, the camera lens tends to make a straight line in the real environment become a curve [5]. The closer to the edge of the image, the more obvious this phenomenon is. Since the lenses actually produced are often center-symmetrical, this makes the irregular distortion generally radially symmetrical. They fall into two main categories: *barrel-like distortion* and *cushion-like distortion*, as shown by Figure 4-3.

In barrel distortion, the radius of pixels decreases as the optical axis's distance increases, while the cushion distortion is just the opposite. In both distortions, the line that intersects the center of the image and the optical axis remains the same.

In addition to the shape of the lens, which introduces radial distortion, *tangential distortion* is introduced during assembly of the camera because the lens and the imaging surface cannot be strictly parallel, as shown by Figure 4-4.

To better understand radial and tangential distortion, we describe them in a more rigorous mathematical form. Consider any point on the *normalized plane*, $\mathbf{p}$, whose coordinates are $[x, y]^T$, or $[r, \theta]^T$ in the form of polar coordinates, where r represents the distance between the point $\mathbf{p}$ and the origin of the coordinate system, and θ represents the angle to the horizontal axis. Radial distortion can be seen as a change in the coordinate point along the length, that is, its radius from the

[4]Note that in the calculation, it is necessary to check whether Z is positive because the negative Z can also get the point on the normalized plane by this method. However, the camera does not capture the scene behind the imaging plane.

[5]Yes, it is no longer straight but becomes curved. If it makes an inside curve, it is called barrel-like distortion; otherwise, it is cushion-like distortion if the curve looks outward.

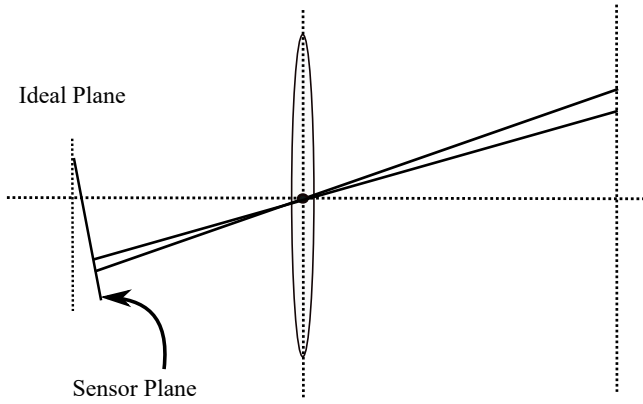

Figure 4-4: Tangential distortion.

origin. Tangential distortion can be seen as a change in the coordinate point along the tangential direction, which means that the horizontal angle has changed. It is generally assumed that these distortions are polynomial, namely:

$$\begin{aligned} x_{\text{distorted}} &= x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \\ y_{\text{distorted}} &= y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \end{aligned}, \tag{4.10}$$

where $[x_{\text{distorted}}, y_{\text{distorted}}]^T$ is the *normalized coordinates* of the point after distortion. On the other hand, for *tangential distortion*, we can use the other two parameters p_1, p_2 to describe it:

$$\begin{aligned} x_{\text{distorted}} &= x + 2p_1 xy + p_2(r^2 + 2x^2) \\ y_{\text{distorted}} &= y + p_1(r^2 + 2y^2) + 2p_2 xy \end{aligned}. \tag{4.11}$$

Putting (4.10) and (4.11) together we get a joint model with 5 distortion coefficients. The complete form is:

$$\begin{cases} x_{\text{distorted}} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + 2p_1 xy + p_2(r^2 + 2x^2) \\ y_{\text{distorted}} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + p_1(r^2 + 2y^2) + 2p_2 xy \end{cases}. \tag{4.12}$$

In the above process of correcting distortion, we used 5 distortion coefficients. In practical applications, you can flexibly choose to number of parameters, for example, only selecting k_1, p_1, p_2, or use k_1, k_2, p_1, p_2, etc.

This section models the camera's imaging process using a pinhole model and described the radial and tangential distortions caused by the lens. Researchers have proposed many other models in the existing imaging system, such as the affine model and perspective model, and there are many different types of distortion. In most visual SLAM systems, pinhole and rad-tan distortion models are sufficient, so we will not describe the other ones.

It is worth mentioning that there are two ways of undistortion (or correction). We can choose to undistort the entire image first, get the corrected image, and then discuss the spatial position of the points on the image. Alternatively, we can also extract some feature points in the distorted image and find its real position through the distortion equation. Both are feasible, but the former seems to be more common in visual SLAM. Therefore, when an image is undistorted, we can directly establish a projection relationship with the pinhole model without considering distortion.

Therefore, in the discussion that follows, we can directly assume that the image has been undistorted.

Finally, let's summarize the imaging process of a monocular camera:

1. First, there is a point P in the world coordinate system, and its world coordinates are $\mathbf{P}_w$.
2. Since the camera is moving, its motion is described by $\mathbf{R}, \mathbf{t}$ or transform matrix $\mathbf{T} \in \mathrm{SE}(3)$. The camera coordinates for P are $\tilde{\mathbf{P}}_c = \mathbf{R}\mathbf{P}_w + \mathbf{t}$.
3. The $\tilde{\mathbf{P}}_c$ components are X, Y, Z, and they are projected onto the normalized plane $Z = 1$ to get the normalized coordinates: $\mathbf{P}_c = [X/Z, Y/Z, 1]^T$. Note that Z may be less than 1, indicating that the point is behind the normalization plane and it should not be projected on the camera plane.
4. If the image is distorted, the coordinates of $\mathbf{P}_c$ after distortion are calculated according to the distortion parameters.
5. Finally, the distorted coordinates of P pass through the intrinsics and we find its pixel coordinates: $\mathbf{P}_{uv} = \mathbf{K}\mathbf{P}_c$.

In summary, we have talked about four coordinates: the world coordinates, the camera coordinates, the normalized coordinates, and the pixel coordinates. Readers should clarify their relationship. They reflect the entire imaging process and will be used in the future.

4.1.3 Stereo Cameras

The pinhole camera model describes the imaging model of a single camera. However, we cannot determine the specific location of a spatial point only by a single pixel. This is because all points on the line from the camera's optical center to the normalized plane can be projected onto that pixel. Only when the depth of P is determined (such as through a binocular or RGB-D camera) can we know exactly its spatial location, as shown in Figure 4-5 .

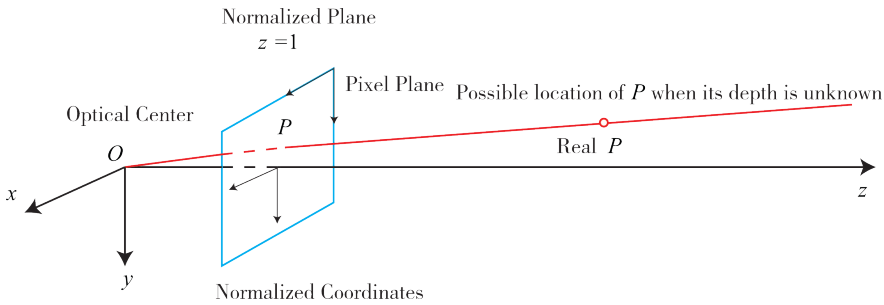

Figure 4-5: The possible location of a single pixel.

There are many ways to measure the pixel distance (or depth). For example, the human eye can judge the object's distance according to the difference (or parallax) of the scene seen by the left and right eyes. The binocular camera principle is also the

same. By simultaneously acquiring the left and right cameras' images and calculating the parallax/disparity between the images, each pixel's depth is estimated. In the following paragraph, we briefly describe the stereo camera's imaging principle (as shown in Figure 4-6).

A binocular camera is generally composed of a left-eye camera and a right-eye camera. Of course, it can also be made up and down, but the mainstream binoculars we've seen are all left and right. Both the left and right cameras are regarded as simple pinhole cameras. They are synchronized and placed horizontally, meaning that both cameras' centers are on the same x axis. The distance between the two centers is called *baseline* (denoted as b), which is an important parameter.

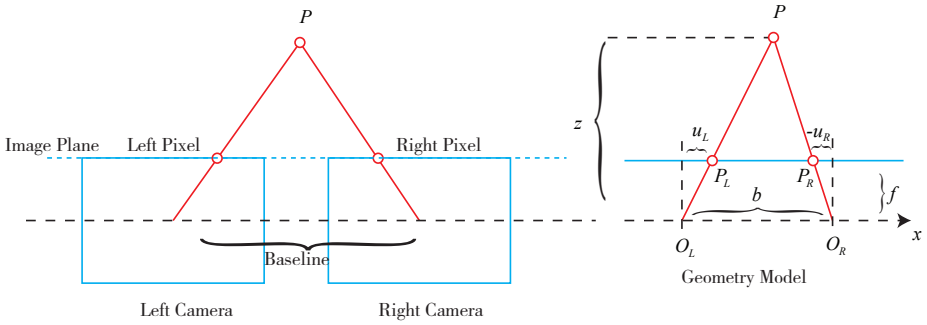

Figure 4-6: Geometry model of stereo cameras from upside down view. The O_L, O_R are left and right optical centers. f is the focal length, u_L and u_R are pixel coordinates of the same point along the x axis. Note that u_R should be a negative value in this figure, so the physical distance should be $-u_R$.

Now consider a 3D point P, projected into the left-eye and the right-eye, written as P_L, P_R. Due to the presence of the camera baseline, these two imaging positions are different. Ideally, since the left and right cameras are only shifted on the x axis, the image of P also differs only on the x axis (corresponding to the u axis of the image). Take the left pixel coordinate as u_L and the right coordinate as u_R. The geometric relationship is shown on the right of Figure 4-6. According to the similarity relationship between $\triangle PP_LP_R$ and $\triangle PO_LO_R$, there are:

$$\frac{z-f}{z} = \frac{b-u_L+u_R}{b}. \tag{4.13}$$

Rearrange it, and we have:

$$z = \frac{fb}{d}, \quad d \triangleq u_L - u_R, \tag{4.14}$$

where d is defined as the difference between the left and right figures' horizontal coordinates and is called disparity or parallax. Based on the parallax, we can estimate the distance between a pixel and the camera. Parallax is inversely proportional to distance: the larger the parallax is, the closer the distance is [6]. Simultaneously, since the parallax is at least one pixel, there is a theoretical maximum value for the binocular depth, which is determined by fb. To see the far-away things, we need a

[6] Readers can simulate it with your own eyes.

larger stereo camera; conversely, small binocular devices can only measure very close distances. By analogy, when the human eye looks at a very distant object (such as a very distant airplane), it is usually impossible to determine its distance accurately.

Although the depth's formula is simple, the real calculation of d itself is more complicated. We need to know precisely where a pixel of the left-eye image appears in the right-eye image (that is, the corresponding relationship). This also belongs to the kind of task that is "easy for humans but difficult for computers". When we want to calculate each pixel's depth in an image, the calculation amount and accuracy will become a problem, and the parallax can be calculated only in the place where the image texture is rich. Due to the calculation amount, binocular depth estimation still needs GPU or FPGA to make the distance calculation run in real-time. This will be mentioned in lecture 11.

4.1.4 RGB-D Cameras

Compared to the binocular camera's way of calculating depth, the RGB-D camera's approach is more "active": it can actively measure each pixel's depth. The current RGB-D cameras can be divided into two categories according to their principle (see Figure 4-7):

1. The first kind of RGB-D sensor uses structured infrared light to measure pixel distance. Many of the old RGB-D sensors are belong to this kind, for example, the Kinect 1st generation, Project Tango 1st generation, Intel RealSense, etc.

2. The second kind measures pixel distance using the *time-of-flight (ToF)*. Examples are Kinect 2 and some existing ToF sensors in cellphones.

Regardless of the type, the RGB-D camera needs to emit a light beam (usually infrared light) to the target object. In the structured light principle, the camera calculates the distance between the object and itself based on the returned structured light pattern. In the ToF principle, the camera emits a light pulse to the target and then determines the distance according to the beam's time of flight. The ToF principle is very similar to the laser sensor, except that the laser obtains the distance by scanning point by point (or line by line). The ToF camera can obtain the entire image's pixel depth, which is also the RGB-D camera's main advantage. So, if you take apart an RGB-D camera, you will usually find that there will be at least one transmitter and one receiver in addition to the ordinary camera.

After measuring the depth, the RGB-D camera usually completes the pairing between the depth and color map pixels according to each camera's position at the time of production. It outputs a pixel-to-pixel corresponding color image and depth image. We can read the color information and distance information at the same image position, calculate the 3D camera coordinates of the pixels, and generate a point cloud. RGB-D data can be processed either at the image level or the point cloud level. The second experiment of this lecture will demonstrate the point cloud construction of RGB-D cameras.

The RGB-D camera can measure the distance of each pixel in real-time. However, due to this measurement of transmitting and receiving, its range of use is limited. RGB-D cameras that use infrared light for depth measurement are susceptible to interference from infrared light emitted by daylight or other sensors, so they cannot be used outdoors. Without modulation, multiple RGB-D cameras can interfere with each other. These points' positions cannot be measured for transparent