# 3
# Rendering depth images

Before describing the proposed rendering method, we present the basic concepts involved in using depth images for 3D warping and novel view synthesis. We start with a review on image acquisition process in Section 3.1. In Section 3.2 we detail depth images as a scene's explicit geometry representation. In Section 3.3 we explain the 3D warping process in the view-dependent texture-mapping (VDTM) technique context. Finally, we conclude in Section 3.4 listing some problems associated to 3D warping and how they can be identified for further dealing.



## 3.1
## Image acquisition process

When a physical device, i.e. an *input camera*, is used to acquire a photograph of a scene, we can simplify the image acquisition process using a pinhole camera model [16]. In this model, the imaging process is a sequence

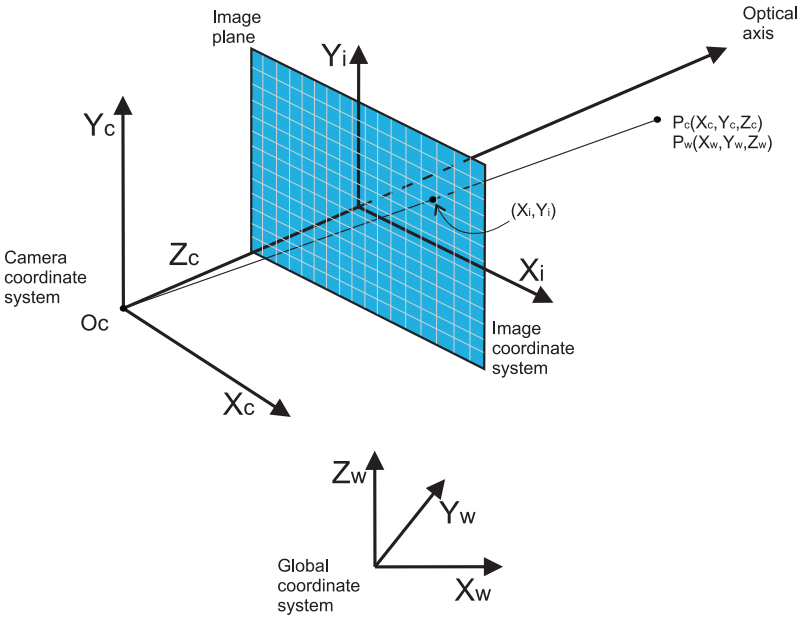


Figure 3.1: Imaging process in the pinhole camera model: image formation is a sequence of transformations between coordinate systems. We ignore here radial distortion.

of transforms between different coordinate systems, as depicted in Figure 3.1.

The *global coordinate system* (or GCS) relates objects in 3D space, differing their relative position and pose. Points $P_w(X_w, Y_w, Z_w, 1)$ in GCS are defined using homogeneous coordinates.

Each *input camera* contains associated *calibration data*, which identifies the camera's pose and intrinsic properties. It consists of two matrices: view matrix $V_{4,4}$ and calibration matrix $K_{3,4}$ [16].

Matrix $\mathbf{V}$ gives the camera's position (given by a translation vector $t$) and pose (given by a rotation matrix $R_{3,3}$) according to GCS:

$$\mathbf{V} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Matrix $\mathbf{K}$ contains camera's intrinsic properties used for perspective projection [16]. It consists of field of view ($f$), aspect ratio ($a$), skew factor ($s$) and image optical center $(x_0, y_0)$:

$$\mathbf{K} = \begin{bmatrix} f & s & x_0 & 0 \\ 0 & af & y_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

So, to express a point $P_w$ relative to the *camera coordinate system* (or CCS), i.e. to obtain $P_c(X_c, Y_c, Z_c, W_c)$, a transform between two orthogonal references in tridimensional space must be done, using matrix $V$::

$$P_c^T = \begin{pmatrix} X_c & Y_c & Z_c & W_c \end{pmatrix}^T = \mathbf{V} \begin{pmatrix} X_w & Y_w & Z_w & 1 \end{pmatrix}^T \tag{3-1}$$

To finally obtain point $P_i(wX_i, wY_i, w)$ in *image coordinate system* (or ICS), a perspective projection is done using calibration matrix $K$:

$$P_i^T = \begin{pmatrix} wX_i & wY_i & w \end{pmatrix}^T = \mathbf{K} \begin{pmatrix} X_c & Y_c & Z_c & W_c \end{pmatrix}^T \tag{3-2}$$

Equation 3-3 summarizes the process of converting a 3D point in global coordinate system into image coordinate system:

$$P_i^T = \begin{pmatrix} wX_i \\ wY_i \\ w \end{pmatrix} = \begin{bmatrix} f & s & x & 0 \\ 0 & af & y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix} \tag{3-3}$$

Figure 3.2: Example of a depth image: color image + dense depth map. Darker pixels in depth map mean greater depth. Courtesy of Zitnick et al [38].

## 3.2 Depth image representation

When a conventional digital camera is used to take a photograph of a scene, only the color information is acquired and stored. Depth can be acquired by special range sensing devices or commercial depth cameras like ZCam [2], or alternatively through algorithms for depth recovery from stereo. A good review on depth estimation techniques is presented by Szeliski [28].

This representation composed of a color image and a dense depth map, with one associated depth value for each pixel in the image, is called *depth image*. An example is shown in Figure 3.2.

Depth stored in the depth map can be relative to any selected coordinate system. Let us assume that depth is written in *global coordinate system*. It means that each pixel $p(X_i, Y_i)$ in a depth map stores an associated $Z_w$ value.

One detail to notice is that depth maps are usually gray-level images, commonly in 8-bit format. For that reason, a linearization of actual depth $Z_w$ into a depth level $d$ is usually applied before storage. Equation 3-4 shows how depth can be linearized to generate a value $d$ in range $[0, 255]$, using the information of minimum and maximum $Z_w$ values in each depth map, $minZ_w$ and $maxZ_w$ respectively.

$$d = 255 \frac{\frac{1}{maxZ_w} - \frac{1}{Z_w}}{\frac{1}{maxZ_w} - \frac{1}{minZ_w}} \tag{3-4}$$

After fetching pixel $p(X_i, Y_i)$ in a depth map, with depth level $d$ in range $[0, 255]$, actual depth $Z_w$ can be retrieved using the inverse of equation 3-4:

$$Z_w = \frac{1}{\frac{d}{255}(\frac{1}{minZ_w} - \frac{1}{maxZ_w}) + \frac{1}{maxZ_w}} \tag{3-5}$$

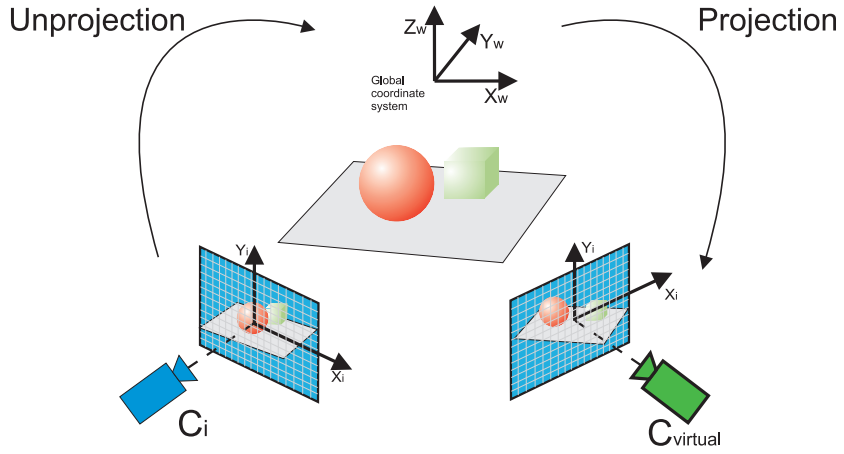


Figure 3.3: 3D warping process. First, 3D mesh generated using depth map from reference camera $C_i$ is unprojected to global coordinate system. Then, mesh is projected into a virtual view using camera $C_{virtual}$ projection matrix.

## 3.3 3D warping

The 3D warping, suggested by McMillan [22], consists in constructing a mesh for a scene with known geometry, and warping that mesh from a reference view into a desired virtual view. Not only is that choice very interesting for its suitability to graphics hardware implementation, but also because a depth map suggests a straighforward regular, dense structure for mesh construction. Besides, the 3D warping technique does not suffer from side effects that impair rendering quality in other methods, as it is the case of blurring in splatting method.

As explained in detail by Debevec et al [10], the technique of view-dependent texture-mapping (VDTM) can take advantage of projective texture-mapping, a built-in feature of graphics hardware. In his framework, each view gives rise to a view-dependent mesh, which is warped to the new viewpoint and colored with the use of projective texture-mapping. Finally those warped views are blended to generate the final image. In this section we focus on how the 3D warping phase is performed.

In our work, we assume that input cameras' calibration data (in the form of matrices $V$ and $K$) and depth images (color + depth) are provided as input. With that information, it is possible to reconstruct the scene's geometry and apply the 3D warping process for view synthesis. This process is depicted in Figure 3.3. Firstly, depth map for reference view $C_i$ is used to generate a regular, triangular 3D mesh with the same resolution as the depth map. We prefer not to downsample the depth map when constructing the 3D mesh so as to avoid blurring effects in objects' boundaries, as reported in [13]. In fact,

with current GPUs memory and bandwidth increases, such a dense mesh does not present a problem for real implementations.

Initially, each mesh's vertice contains values $(X_i, Y_i, d)$. Actual depth $Z_w$ can be retrieved by using equation 3-5. Next step is to retrieve vertice's global coordinates $(X_w, Y_w, Z_w)$, which corresponds to the unprojection step in Figure 3.3.

First, for brevity, we introduce matrix $P$ into equation 3-3, with $P = K \times V$:

$$\begin{pmatrix} wX_i \\ wY_i \\ w \end{pmatrix} = \begin{bmatrix} P_{11} & P_{12} & P_{13} & P_{14} \\ P_{21} & P_{22} & P_{23} & P_{24} \\ P_{31} & P_{32} & P_{33} & P_{34} \end{bmatrix} \begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix} \tag{3-6}$$

That determines the following system of linear equations:

$$\begin{cases} wX_i &= P_{11}X_w + P_{12}Y_w + P_{13}Z_w + P_{14} \\ wY_i &= P_{21}X_w + P_{22}Y_w + P_{23}Z_w + P_{24} \\ w &= P_{31}X_w + P_{32}Y_w + P_{33}Z_w + P_{34} \end{cases}$$

In that system of equations, $X_i$, $Y_i$, $Z_w$ and matrix $P$ are known. So, after some algebraic manipulation, we can solve the system for $X_w$ and $Y_w$:

$$\begin{cases} Y_w &= \frac{X_i(c_1P_{31}-c_2P_{21})+Y_i(c_2P_{11}-c_0P_{31})+c_0P_{21}-c_1P_{11}}{Y_i(P_{31}P_{12}-P_{32}P_{11})+X_i(P_{21}P_{32}-P_{22}P_{31})+P_{11}P_{22}-P_{21}P_{12}} \\ X_w &= \frac{Y_w(P_{12}-P_{32}X_i)+c_0-c_2X_i}{P_{31}X_i-P_{11}} \end{cases} \tag{3-7}$$

where:

$$\begin{cases} c_0 &= Z_wP_{13} + P_{14} \\ c_1 &= Z_wP_{23} + P_{24} \\ c_2 &= Z_wP_{33} + P_{34} \end{cases} \tag{3-8}$$

Finally, the 3D mesh must be projected to the new viewpoint $C_{virtual}$, using its calibration data matrices $V$ and $K$(Section 3.1), using equation 3-3.

Figure 3.4 illustrates the result of the described 3D warping process for an input depth image.

## 3.4 Artifacts inherent to 3D warping

Figure 3.4 shows that the 3D warping process generally succeed in generating high-quality synthesized views, since color is linearly interpolated between the transformed sample locations. However, it assumes continuity between neighboring samples, which is not the case in objects' boundaries. As a result, "rubber sheets" appear between foreground and background objects

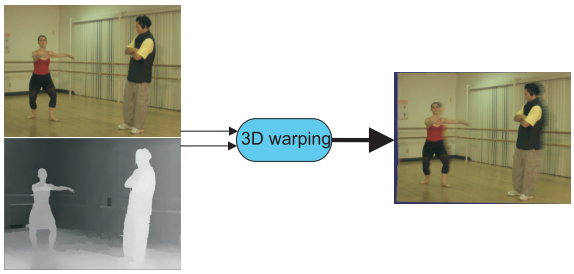


Figure 3.4: 3D warping result. Depth image (pair of images to the left) from a reference camera is projected into a virtual view (right image) using the described 3D warping process. In this case, virtual camera $C_{virtual}$ was placed slightly to the right of reference camera $C_i$.

Figure 3.5: 3D warping artifacts due to discontinuity in depth. The continuity assumption in depth map does not hold at objects' boundaries, which causes undesirable artifacts in the form of stretched triangles in warped view. When depth map (left) is warped to new viewpoint, the region to the right of the ballet dancer reveals occluded areas.

in the warped view. A close-up on this undesirable behavior is shown in Figure 3.5. Those regions with "rubber sheets" coincide with occlusion areas, i.e. regions which have not been captured by reference camera $C_i$ and that become exposed when warping to new viewpoint $C_{virtual}$. To identify them, we label vertices with a method similar to the one described by Zitnick et al [38].

Basically, we test a vertice against its four-neigbors to label it as occluded or not (actually the "occluded" label should be understood as not trustable, meaning that it may belong to an occlusion region). We used a threshold for depth difference to label vertices as occluded (not trustable) or not occluded (belonging to a region with continuous depth). Although a more sophysticated border detection algorithm would yield better results, we found that the proposed method gives good results and is very fast to perform in graphics cards due to local cache.

Figure 3.6 shows a warped view with occlusion regions drawn in black, thank to our labeling approach. The missing information in occlusion regions can be filled with data obtained from other reference cameras, through a compositing technique. Notice in Figure 3.6 that our algorithm labeled some regions in the floor as occluded due to discontinuities in depth above the used

Figure 3.6: 3D warping to a new viewpoint, with occluded regions drawn in black thank to our labeling schema.

threshold.