4
Compositing

As shown in the previous chapter, a single reference view generally does not have enough information for synthesis of virtual views free from artifacts, like those caused by occlusions. That is the motivation for using multiple views for completing the missing information.

The original View-Dependent Texture-Mapping method suggested by Debevec et al [10] uses many views to build up a virtual one. Their method computes which polygons are visible in each image, and from which direction. With that information view maps are built for each polygon, and during rendering the three closest viewing directions are chosen and their relative weights for blending computed.

In a simpler but still effective manner, Zitnick et al [38] showed that good rendering results can be achieved using only two reference cameras for compositing, provided the baseline of input cameras is not too wide. With that approach, compositing involves only the trivial determination of which pair of cameras to use for rendering, and the computation and usage of blending weights for each pixel.

In this chapter we explain the process of compositing two reference views for virtual view synthesis. In Section 4.1 we describe how the virtual camera navigation in such a blending system works. In Section 4.2 we explain our blending algorithm, and finally present limitations to our compositing method in Section 4.3.

4.1
Virtual camera navigation

We assume that the cameras setup used to generate the input for our method is similar to the one used by Zitnick et al [38], which is depicted in Figure 4.1. The arrangement of input cameras along a 1D arc leads to a simple way of interpolating cameras: virtual camera can have its movement restricted to the lines linking each pair of adjacent cameras, as shown in Figure 4.2.

When virtual camera navigation follows that restriction, its matrices $K_{virtual}$ and $V_{virtual}$ can be determined by linear interpolation of parameters

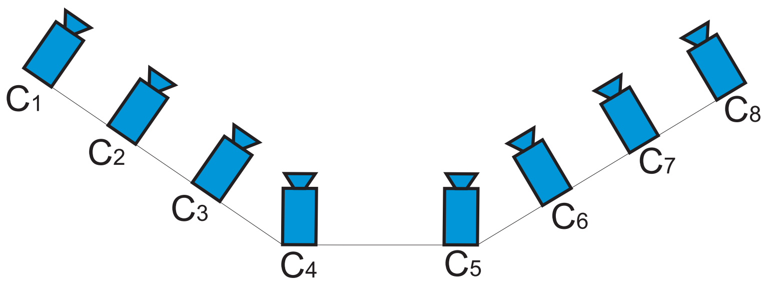


Figure 4.1: Example of cameras setup: input cameras arranged along a 1D arc.

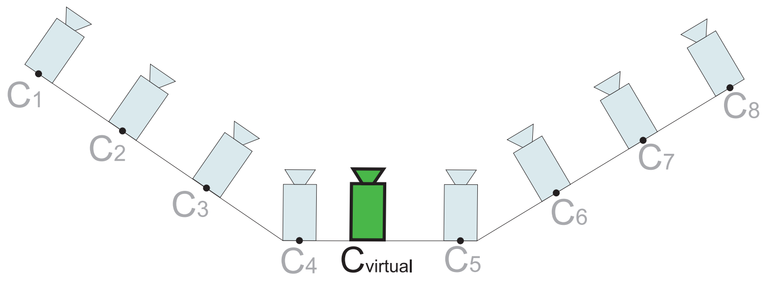


Figure 4.2: Navigation of virtual camera $C_{virtual}$ restricted to the lines linking center of adjacent pair of input cameras.

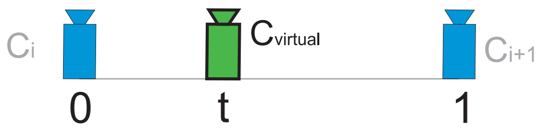


Figure 4.3: Virtual camera's parameters can be determined as a linear interpolation of adjacent pair of cameras.

for adjacent pair of input cameras i and $i+1$, namely C_i and C_{i+1}. This interpolation process is illustrated in Figure 4.3, with t representing the interpolation factor ($0 \leq t \leq 1$). We apply this interpolation process in the proposed method. Virtual camera's calibration matrix $K_{virtual}$ can be determined using calibration matrices K_i and K_{i+1}:

$$K_{virtual} = (1-t)K_i + tK_{i+1} \tag{4-1}$$

A similar approach can be used for determining view matrix $V_{virtual}$, but with the additional previous step of decomposing view matrices V_i and V_{i+1} into eye positions, represented by vectors eye_i and eye_{i+1}, and rotations, represented by quaternions Q_i and Q_{i+1}. The resulting interpolated vector $eye_{virtual}$ and quaternion $Q_{virtual}$ can be converted back into $V_{virtual}$.

Equation 4-2 describes the linear interpolation of eye position, and 4-3 represents the spherical linear interpolation of quaternions [4].

$$eye_{virtual} = (1-t)eye_i + t(eye_{i+1}) \tag{4-2}$$

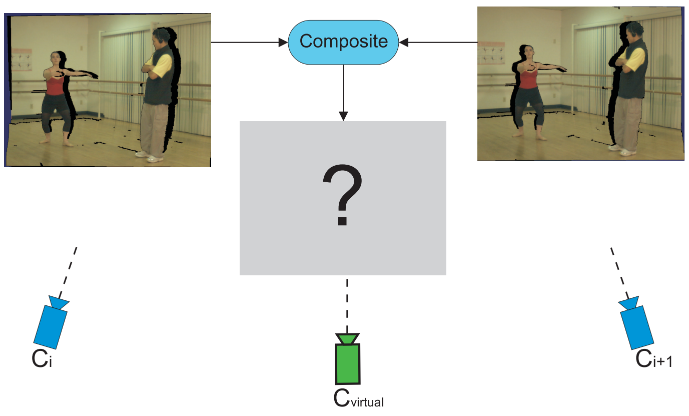


Figure 4.4: Blending algorithm. Two reference views C_i and C_{i+1}, adjacent to virtual viewpoint $C_{virtual}$, are used in composition stage.

$$Q_{virtual} = slerp(t, Q_i, Q_{i+1}) \tag{4-3}$$

Equations for decomposing view matrix V into eye position *eye* and rotation quaternion Q, and building up V from *eye* and Q can be found in [4].

A more sophisticated camera navigation can certainly be employed using similar interpolation principles. For intance, one could use a spline rather than line segments for virtual camera path, which would result in smoother navigation.

Also, the viewpoint could move freely without the constraint of in-between cameras paths, but that would cause sampling issues with extreme zooming-in or zooming-out: holes might appear. We preferred instead to keep the camera movement simple, because it is reasonable to assume that the input cameras arrangement can be planned taking the desirable virtual paths into consideration.

4.2 Blending algorithm

Figure 4.4 illustrates the compositing process. Two reference cameras have their views warped and occlusion areas identified as described in Chapter 3, and are blended to generate a final image.

It is interesting that the compositing algorithm have the following characteristics, which are justified subsequentially:

1. Angular distances between reference cameras and the virtual camera influence weighting.
2. Visibility test per-pixel.
3. Pixels marked as occluded should be treated differently.

The first characteristic follows the suggestion of Buehler et al's Unstructured Lumigraph Rendering [6]: when blending multiple views, angular distances between the desired viewpoint and the reference cameras' positions should be used to produce consistent blending. The closer the viewpoint is to a reference camera C_i (in matter of angular distance), the more that reference camera should affect the final color in the rendered image.

Figure 4.5(a) illustrates that concept. Angular distances θ_i and θ_{i+1} are used to measure the influence of each reference view in the final pixel color. For smooth color interpolation [6], weight for view i can be calculated using a cosine function adapted so that $wAng_i \in [0, 1]$:

$$wAng_i = 0.5(1 + cos\frac{\pi\theta_i}{\theta_i + \theta_{i+1}}) \tag{4-4}$$

Besides, as suggested by Porquet et al [26], a visibility test per pixel should be done. That justifies the second mentioned desired characteristic in the list.

However, so as to compensate for errors in depth estimates and cameras registration, that visibility test should be a soft-Z compare similarly to the method proposed by Zitnick et al [38].

When the difference between depth Z_{w_i} for a pixel p_i in view C_i and depth $Z_{w_{i+1}}$ for the equivalent pixel p_{i+1} in view C_{i+1} is below a threshold value, color values from p_i and p_{i+1} are blended. Otherwise, the color of the pixel closest to virtual camera $C_{virtual}$ is used. That scenario is depicted in Figure 4.5(b): in that case, pixel from C_i (lying in the orange object) is much closer to the virtual camera $C_{virtual}$ that is pixel from C_{i+1} (lying in the farther green illustrated object), and therefore the former's color is used solely to define p_v.

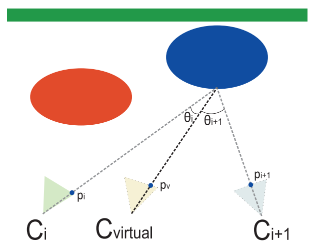
Finally, the third characteristic in the list suggest that pixels marked as occluded be treated differently than others. As already mentioned, those pixels may result in rubber sheets or reveal unsampled areas in the warped image. That need is exemplified in Figure 4.5(c), in which the pixel from C_i is marked as occluded, and in fact that pixel belongs to a rubber sheet in the warped view from C_i. Therefore, p_v gets is color from pixel p_{i+1}, from camera C_{i+1}.

Finally, having calculated weights for each reference camera (normalized weights), the final pixel color $color(p_v)$ is computed:

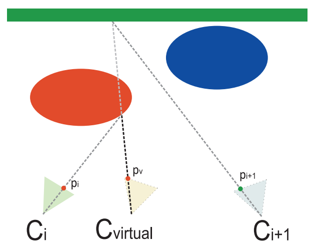
$$color(p_v) = color(p_i)w_i + color(p_{i+1})w_{i+1} \tag{4-5}$$

4.3 Limitations

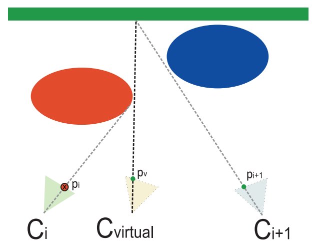
The compositing algorithm described in the previous section smoothly interpolates the contribution from each reference view in a pixel-based ap-



4.5(a): Weights based on angular distances.



4.5(b): Visibility test per pixel.



4.5(c): Pixels marked as occluded ignored.

Figure 4.5: Cases considered in the compositing process.

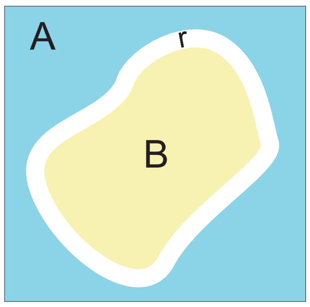


Figure 4.6: Frontier r between regions A and B may become undesirably visible due to the stepped behavior of our compositing algorithm and there exist photometric (e.g. gain) differences in cameras used for capture.

proach. Even though this approach for smooth transition behaves well in most cases, it may cause an undesirable side-effect.

Refer to Figure 4.6. Consider two neighboring regions (group of connected pixels) A and B in the final image. Say A contains pixels which were blended, and B represents an occlusion area, so its pixels have color contribution solely from one reference view. In that situation, the frontier r between A and B may become clearly visible, as a result of photometric differences between reference views. Figure 4.7 exemplifies the appearance of seams. Applying Gaussian blur at the frontiers of A and B would be a simple solution for the problem, but that would also soften the objects boundaries. To solve correctly the problem, an alternative approach should be used for blending seamlessly those areas, like gradient compositing [27].

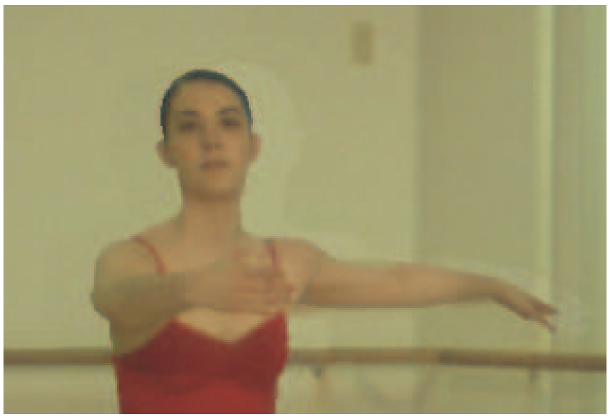


Figure 4.7: Visible seams between occlusion and blended areas, to the right and to the left of woman's head.