EUROGRAPHICS 2022 / R. Chaine and M. H. Kim
(Guest Editors)

Volume 41 (2022), Number 2

ZoomShop: Depth-Aware Editing of Photographic Composition (Supplemental Material)

Sean J. Liu[1], Maneesh Agrawala[1], Stephen DiVerdi[2] and Aaron Hertzmann[2]

[1]Stanford University
[2]Adobe Research

Appendix A: Geometric Description and Derivation of $b(z)$

Here, we show how our $b(z)$ camera parameterization can be understood geometrically in terms of non-linear camera models. We first show that a piecewise-linear, continuous choice of $b(z)$ corresponds to a sequence of linear camera models, each applied to different depth ranges, equivalent to the Computational Zoom model proposed by Badki et al. [BGKS17]. We then describe generalizations to non-linear and non-continuous $b(z)$, and what these correspond to geometrically. For each type of these $b(z)$ (i.e., piecewise linear, curved, and discontinuous), we show that Equation 1 holds:

$$u = \frac{x}{b(z)} \tag{1}$$

Figure 1 shows the boundary curve $b(z)$ for linear perspective, piecewise linear, and curved camera models.

A.1 Piecewise Linear Camera Model

We first explore a piecewise linear camera model. In this model, separate linear cameras are applied to each depth range, with continuity constraints between the cameras. We show how the $b(z)$ formulation can be derived from this model. Note that this model is equivalent to Computational Zoom [BGKS17], and this shows how Computational Zoom is a special case of our framework.

Figure 1b shows a piecewise linear camera model. With the camera at the origin, and an image plane at focal depth $f = z_0$, we divide up the scene into a series of *depth zones*, bounded by depth planes $z_1 ... z_N$, where $z_i < z_{i+1}$ for $i \in [0...N-1]$. For each depth plane, the corresponding half-plane width is P_i, which also marks the view boundary at z_i.

Virtual Camera Positions. In the piecewise linear camera model, we can treat each individual depth zone i as a conventional linear perspective projection, with P_{i-1} as the "image plane" of that zone, and z^{c_i} as the position of the virtual camera. See Figure 1b for an illustration.

For each zone i, we can compute the virtual camera position z^{c_i} by extrapolating where the bounds of zone i will hit the z-axis. Take the example in Figure 1b. The line connecting (z_2, P_2) and (z_3, P_3) is defined by:

$$l(z) = P_2 + \frac{P_3 - P_2}{z_3 - z_2}(z - z_2) \tag{2}$$

The line will intersect the z-axis at z^{c_3}:

$$l(z^{c_3}) = P_2 + \frac{P_3 - P_2}{z_3 - z_2}(z^{c_3} - z_2) \Longrightarrow 0 \tag{3}$$

$$z^{c_3} = z_2 - \frac{P_2(z_3 - z_2)}{P_3 - P_2} \tag{4}$$

In general, for any depth zone i, we can compute z^{c_i}:

$$z^{c_i} = z_{i-1} - \frac{P_{i-1}(z_i - z_{i-1})}{P_i - P_{i-1}} \tag{5}$$

To compute the final position of a scene point (x, z) on P_0, we iteratively project (x, z) onto intermediate "image planes" P_{i-1}, P_{i-2}, ..., until the final image plane P_0.

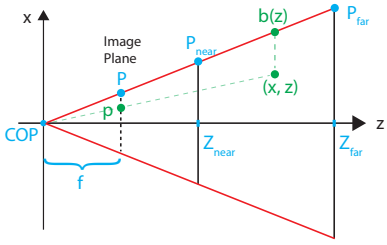
Projection in Closed Form Solution The pseudo-code for projecting a point (x, z) iteratively to $P_{i-1}, P_{i-2}...$ until the image plane P_0 is shown in Figure 2.

In general, given any point (x, z) in depth zone i, we can figure out its projected point onto the previous plane P_{i-1} via $p_x = \frac{f}{z}x$. Offsetting f and z to be with respect to the virtual camera z^{c_i}, the projected position (x', z') onto the previous plane P_{i-1} is:

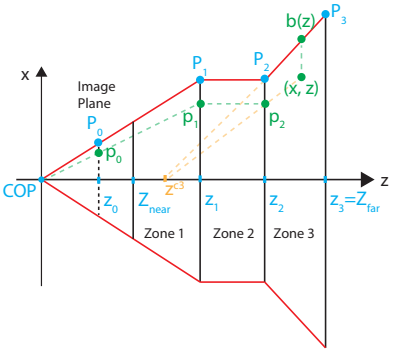
$$x' = \frac{z_{i-1} - z^{c_i}}{z - z^{c_i}}x \tag{6}$$

$$z' = z_{i-1} \tag{7}$$

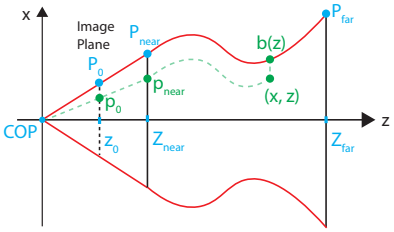
We can convert the pseudo-code iterative projection to closed form. First, we substitute z^{ci} in Equation 6 (also line 9 of the pseudo-code) with Equation 5:



(a) *Linear Perspective Camera Model. P is the half-width of the image plane in world space. Z_{near} and Z_{far} are the z-values of the near and far plane, and f is the focal length of the camera.*



(b) *Piecewise Linear Perspective Camera Model. In this example, the number of depth zones $N = 3$. A scene point (x,z) gets iteratively projected onto planes P_2, P_1, and then P_0 to form the final image. Each depth zone i obeys linear perspective, with P_{i-1} as the virtual image plane and z^{c_i} as the virtual camera. z^{c_3} is shown in yellow.*



(c) *Curved Camera Model. Parameterized by a boundary curve $b(z)$ for $z \in [Z_{near}, Z_{far}]$. A scene point (x,z) is projected onto the near plane $P_{near} = b(Z_{near})$ at (p_{near}, Z_{near}), and then onto the image plane at (p_0, z_0).*

(d) *Comparison of using different camera models, in a 3D scene inspired by Burleigh et al. [BPR18]. In this example, our aim is to scale up the bust while keeping the couches visible. Using linear perspective, the only way is to zoom in crop, which cuts out part of the couch. Using the piecewise linear camera model, we divide the scene into three zones (yellow, green, non-colored) and scale each zone separately. However, this introduces a seam at the zone boundary between the non-colored and green zone (see red circles). Using curved camera rays, we achieve a smoother transition of scale between zones and removes the seams (see green circles). In the last two images, pixels that are not in the original photo are colored in cyan.*

Figure 1: *Camera Models*

$$x' = \frac{z_{i-1} - z^{c_i}}{z - z^{c_i}} x \tag{8}$$

$$= \frac{z_{i-1} - (z_{i-1} - \frac{P_{i-1}(z_i - z_{i-1})}{P_i - P_{i-1}})}{z - (z_{i-1} - \frac{P_{i-1}(z_i - z_{i-1})}{P_i - P_{i-1}})} x \tag{9}$$

$$= \frac{\frac{P_{i-1}(z_i - z_{i-1})}{P_i - P_{i-1}}}{z - z_{i-1} + \frac{P_{i-1}(z_i - z_{i-1})}{P_i - P_{i-1}}} x \tag{10}$$

$$= \frac{P_{i-1}(z_i - z_{i-1})}{(P_i - P_{i-1})(z - z_{i-1}) + P_{i-1}(z_i - z_{i-1})} x \tag{11}$$

$$= \frac{P_{i-1}}{\frac{z - z_{i-1}}{z_i - z_{i-1}}(P_i - P_{i-1}) + P_{i-1}} x \tag{12}$$

$$= \frac{P_{i-1}}{\frac{z - z_{i-1}}{z_i - z_{i-1}} P_i + (1 - \frac{z - z_{i-1}}{z_i - z_{i-1}}) P_{i-1}} x \tag{13}$$

$$= \frac{P_{i-1}}{\frac{z - z_{i-1}}{z_i - z_{i-1}} P_i + \frac{z_i - z}{z_i - z_{i-1}} P_{i-1}} x \tag{14}$$

The denominator $\frac{z - z_{i-1}}{z_i - z_{i-1}} P_i + \frac{z_i - z}{z_i - z_{i-1}} P_{i-1}$ is just a linear interpolation between the zone boundary endpoints at P_{i-1} and P_i. In fact, the denominator is equal to the boundary point $b(z)$ at (x,z).

In other words, we can rewrite Equation 14 and 6 as:

$$x' = \frac{P_{i-1}}{b(z)} x \tag{15}$$

(x', z') is the projection of (x,z) to plane P_{i-1}. Following the pseudo-code, in the next iteration, we project (x', z') to P_{i-2}. Let the projected point onto P_{i-2} be (x'', z''). Then, using Equation 14:

$$x'' = \frac{P_{i-2}}{b(z')} x' \tag{16}$$

$$= \frac{P_{i-2}}{b(z_{i-1})} x' \tag{17}$$

$$= \frac{P_{i-2}}{P_{i-1}} x' \tag{18}$$

$$z'' = z_{i-2} \tag{19}$$

```
# i is zone index of point (x,z).
project(x, z, i):
    # If in zone 1, just project onto P_0
    if i == 1:
        return p_0 = (z_0 / z) x
    else:
        # Compute the projected point (x',z') onto P_{i-1}
        # z^{c_i} is position of zone i's 'ghost' camera
        x' = ((z_{i-1} - z^{c_i}) / (z - z^{c_i})) x  # See Equations 8 and 9
        z' = z_{i-1}
        return project(x', z', i-1)
```

Figure 2: *Piecewise Linear: Pseudo-code for projecting a point (x,z) iteratively onto the image plane P_0.*

where $b(z_{i-1}) = P_{i-1}$ because (x',z') is the projected point on plane P_{i-1}, and the boundary at $z' = z_{i-1}$ is precisely P_{i-1}.

By the same logic, in subsequent iterations, when projecting (x_j,z_j) at plane P_j onto P_{j-1}, the scale factor for x_j is $\frac{P_{j-1}}{b(z_j)} = \frac{P_{j-1}}{P_j}$. Thus, the closed form solution for projecting (x,z) to the image plane P_0 is:

$$p_0 = \frac{P_0}{P_1}\frac{P_1}{P_2}\cdots\frac{P_{i-2}}{P_{i-1}}\frac{P_{i-1}}{b(z)}x \tag{20}$$

$$= \frac{P_0}{b(z)}x \tag{21}$$

where p_0 is the projected (world) coordinate of x onto image plane at z_0.

The normalized image coordinates $u_0 \in [-1,1]$ of the projection is:

$$u_0 = \frac{1}{P_0}\frac{P_0}{b(z)}x \tag{22}$$

$$= \frac{x}{b(z)} \tag{23}$$

where P_0 is the half-width of the image plane in world space.

Another way to interpret this piecewise linear model is in terms of light paths. In a conventional pinhole camera model, light follows a straight line from a scene point to the camera's focal center and intersects the image plane. In the piecewise linear model, a light path from a scene point follows a sequence of straight lines, bending at each depth zone boundary (Figure 1b).

This geometric interpretation of light paths can be extended to the other camera models as well. For all camera models in our framework, the light paths follow the shape of $b(z)$, i.e., for a scene point at position (x_0,y_0,z_0), the light path is a curve given by $f(z) = (\frac{x_0}{b(z_0)}b(z), \frac{y_0}{\lambda b(z_0)}b(z), z)$, where $\lambda = H/W$ is the image aspect ratio.

A.2: Curved Paths

We can generalize the piecewise linear model to a curved model (Figure 1c). In this generalized form, camera paths are no longer piecewise linear, but general curves: each light path is a scaled version of $b(z)$.

Equation 1 still applies to the general view boundary curve. Instead of a finite number of piecewise linear depth zones defined by a piecewise linear $b(z)$, we now have a curved $b(z)$, which is equivalent to an infinite number of piecewise linear depth slices, where each slice is infinitesimally thin.

In this curved model, the nearest depth plane (before the image plane) of the scene is $P_{\text{near}} = b(Z_{\text{near}})$, as shown in Figure 1c. Given any scene point (x,z), the projected point onto P_{near}, is:

$$p_{\text{near}} = \frac{P_{\text{near}}}{b(z)}x \tag{24}$$

Next, the projection from $(p_{\text{near}}, Z_{\text{near}})$ onto the image plane P_0 is just conventional linear perspective:

$$p_0 = \frac{f}{z_{\text{near}}}p_{\text{near}} \tag{25}$$

$$= \frac{z_0}{z_{\text{near}}}p_{\text{near}} \tag{26}$$

$$= \frac{z_0}{z_{\text{near}}}\frac{P_{\text{near}}}{b(z)}x \tag{27}$$

$$= (z_0\frac{P_{\text{near}}}{z_{\text{near}}})\frac{x}{b(z)} \tag{28}$$

$$= P_0\frac{x}{b(z)} \tag{29}$$

where P_0 is the half-width of the image plane in world space. We can convert p_0 from world coordinates to normalized image coordinates $u_x \in [-1,1]$:

$$u_x = \frac{p_0}{P_0} \tag{30}$$

$$= \frac{x}{b(z)} \tag{31}$$

A.3: Discontinuous Paths

We can further generalize the continuous model to a discontinuous one. Figure 3 shows a piecewise discontinuous model, where camera paths follow discontinuous lines. We show that Equation 1 holds even if $b(z)$ is discontinuous.

For piecewise discontinuous paths, we can define two half-widths P_i^+, P_i^- for each depth plane:

$$b(z) = \frac{z - z_{i-1}}{z_i - z_{i-1}}P_i^+ + \frac{z_i - z}{z_i - z_{i-1}}P_{i-1}^-, \quad \text{where } z_{i-1} \le z < z_i \tag{32}$$

When $P_i^+ \neq P_i^-$, a discontinuity occurs in the camera paths through this point. See Figure 3 for an illustration. We can imagine an infinitesimally thin depth zone at each $z_i, 1 \le i \le N$, so:

$$p_0 = \frac{1}{P_0^+}\cdots\frac{P_{i-1}^+}{P_i^-}\frac{P_i^-}{P_i^+}\frac{P_i^+}{b(z)}x \tag{33}$$

which converts to Equation 1 in image coordinates.

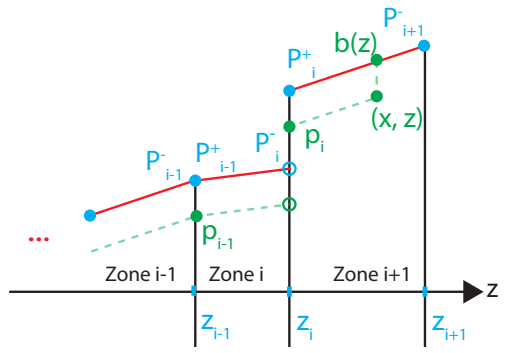


Figure 3: *Illustration of discontinuous piecewise-linear camera paths. We break off each half-width depth plane P_i into two parts, P_i^- and P_i^+. In this example, a discontinuity occurs at z_i because $P_i^- \neq P_i^+$. $b(z)$ is continuous at z_{i-1} because $P_{i-1}^- = P_{i-1}^+$.*

Appendix B: Removing Artifacts

Scaling depth ranges can lead to disocclusions, pixel stretching, or shearing. For example, the teaser figure shows both disoccluded regions (behind the left tree branch) and sheared pixels on the lake in cyan between the yellow and green depth zones. Both of these issues introduce artifacts, which we address with two heuristics.

In the first heuristic, we check the amount of shearing of each pixel in the final image. If a pixel is significantly sheared, it most likely saddles between two different depths that were scaled differently. This may lead to visible artifacts and/or disocclusion. To check for shearing, in the fragment shader, we check the dot product between vectors $\frac{d\vec{u}}{dx} = \begin{bmatrix}\frac{du}{dx} & \frac{dv}{dx}\end{bmatrix}^T$ and $\frac{d\vec{u}}{dy} = \begin{bmatrix}\frac{du}{dy} & \frac{dv}{dy}\end{bmatrix}^T$. If the vectors are orthogonal, then there's no shear. But if the dot product is greater than some threshold τ_{shear}, then we make the pixel transparent:

$$\frac{d\vec{u}}{dx} \cdot \frac{d\vec{u}}{dy} > \tau_{\text{shear}} \tag{34}$$

We use $\tau_{\text{shear}} = 0.53 - 0.9$ in our results.

In the second heuristic, we check for the amount of non-uniform scaling (stretching). A pixel that's stretched significantly in the x-direction, again, likely lies between two different depths that were scaled differently, and thus introduces disocclusion or artifacts. To check for non-uniform scaling, in the fragment shader, we compute the ratio of $||\frac{d\vec{u}}{dx}||$ and $||\frac{d\vec{u}}{dy}||$. A ratio of 1 means uniform scaling; any deviation means the scaling is non-uniform. If the ratio is less than some threshold $\tau_{\text{nonuniform}}$, then we make the pixel transparent:

$$\frac{||\frac{d\vec{u}}{dx}||}{||\frac{d\vec{u}}{dy}||} < \tau_{\text{nonuniform}} \tag{35}$$

We used $\tau_{\text{nonuniform}} = 0.2 - 0.3$ in our results, and also apply the same check for the y-direction, i.e., $||\frac{d\vec{u}}{dy}||/||\frac{d\vec{u}}{dx}||$.

Appendix C: Additional Translation Results

In Figure 4, scaling up the birds pushed the left rock out of view. In addition to translation constraints on the rock, we added an additional constraint on the float to keep it fixed in place. The output shows the rock translated along with some connected water in front. In Figure 5, compressing the depth in the valley pushed the two side rocks partially out of view. Under the shown constraints, ZoomShop smoothly translates the rocks towards the center as well some ground in front.

References

[BGKS17] BADKI, ABHISHEK, GALLO, ORAZIO, KAUTZ, JAN, and SEN, PRADEEP. "Computational Zoom: A Framework for Post-Capture Image Composition". *ACM Trans. Graph.* 36.4 (July 2017). ISSN: 0730-0301. DOI: 10.1145/3072959.3073687 1.

[BPR18] BURLEIGH, ALISTAIR, PEPPERELL, ROBERT, and RUTA, NICOLE. "Natural perspective: Mapping visual space with art and science". *Vision* 2.2 (2018), 21 2.

[Tre20] TREGASKIS, WADE. *Yosemite Valley*. https://www.flickr.com/photos/wadetregaskis/50156486633. Modified and included with permission. Original photo licensed under CC BY-NC 2.0. Feb. 2020 5.

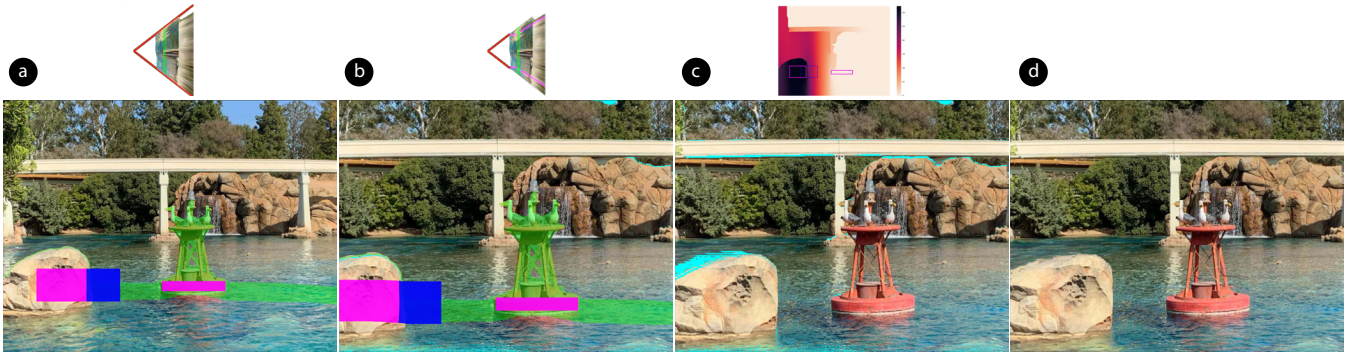


Figure 4: *Birds. Goal: Scale up birds while also keeping the left side rock visible. Magenta and blue rectangles are source-destination pairs which are input to our translation optimization. Each pair of rectangles has the same size; magenta overlays blue rectangles. (a) Original photo. (b) Scaled up birds. (c) ZoomShop output (top: translation map). (d) ZoomShop with inpainting (automatic).*

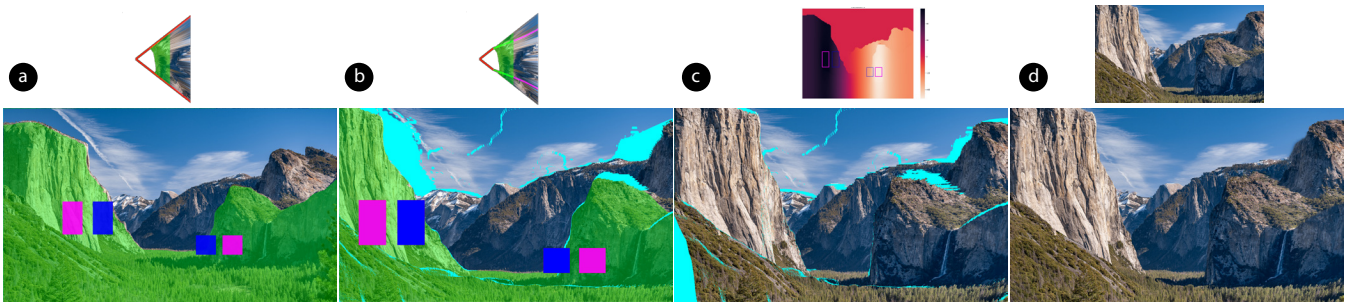


Figure 5: *Yosemite [Tre20]. Goal: Compress depth in valley while keeping two side rocks in view. Magenta and blue rectangles are source-destination pairs which are input to our translation optimization. (a) Original photo. (b) Compressed valley. (c) ZoomShop output (top: translation map). (d) ZoomShop with inpainting (top: automatic, bottom: manual guidance)*