Instant 3D Photography
Implementation Details

Peter Hedman and Johannes Kopf

1 Supplementary Material

1.1 Warping into a central panorama

Once we have computed a 3D pose, and non-rigidly aligned each depth map (see Section 4.2 in the main paper), we have a globally consistent estimate for the 3D location of every pixel in our input images. This makes it easy to warp the input images into a central panoramic image. For each input image, we triangulate its depth map into a grid mesh and render it with the normal rasterization pipeline, letting the depth test select the front surface when multiple points fall onto the same panorama pixel. One problem with this simple approach is the long stretched triangles at depth discontinuities, which might obscure other good content, and we do not want to include them in the stitch. We resolve using the approach described in Section 4.4 in the main paper, i.e. by first estimating where the occlusion boundaries are and removing all triangles that cross these boundaries.

1.2 Color harmonization

As described in the main paper (Section 4.4.3), we convert the images to the channel-decorrelated CIELAB color space, and then process each channel independently. We solve a linear system to compute global affine color-channel adjustments (a scale α and an offset β) for each source image, such that the adjusted color values in the overlapping regions agree as much as possible.

Formally, for each channel we minimize the sum

$$E = \lambda_{pairwise} E_{pairwise} + \lambda_{reg} E_{reg} \tag{1}$$

of a pairwise color alignment term $E_{pairwise}$ (normalized by $\lambda_{pairwise}$) and a regularization term E_{reg} (strength $\lambda_{reg} = 0.33$).

Pairwise term We compute our pairwise term as sum over every pair of overlapping images A, B in the panorama,

$$E_{pairwise} = \sum_{A,B} \sum_{p_i, p_j \in C(A,B)} \|\alpha_A * p_A^i + \beta_A - \alpha_B * p_b^i - \beta_B\|^2, \tag{2}$$

where α_A and α_B are the scale terms for images A and B, β_A and β_B are the offset terms, and $C(A,B)$ is a regularly subsampled set of corresponding pixels between the images A and B. To make the system better conditioned, we do not include over-saturated pixels (luminance $> \tau_{saturated} = 0.98$)

To make sure that the relative strength between the pairwise term and the regularization term does not differ too much, we normalize the pairwise term by the total number of correspondences. In other words,

$$\lambda_{pairwise} = \frac{1}{\sum_{A,B} |C(A,B)|}. \tag{3}$$

Regularization term Our regularization term,

$$E_r = \sum_A \|\alpha_a - 1\|^2 + \|\beta_a\|^2 \tag{4}$$

encourages identity transformations and prevents the trivial solution of scaling everything down to 0.

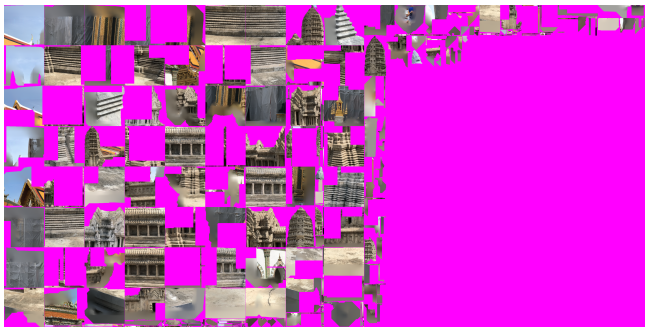

Figure 1: *Texture atlas for the* ANGKOR WAT MINIATURE *scene.*

Over-saturated pixels Since our geometry fusion prevents saturated pixels at all costs, any remaining over-saturated regions in the panorama are likely over-exposed in all of the input images. In this case, applying color harmonization dulls bright highlights. We preserve highlights by computing the final image as a soft blend between the original and the harmonized image, where the threshold above (luminance $> \tau_{saturated}$) ensures that we only use the original image for over-saturated highlights.

1.3 Feathering

We reduce visible seams in the final panorama with feathering, i.e. instead of making a hard choice for the color c_p of a pixel p, we a weighted sum

$$c_p = \frac{\sum w_p^\alpha c_p^\alpha}{\sum w_p^\alpha} \tag{5}$$

of the colors associated with all available labels α.

Seams most often occur at 1) label boundaries 2) input image boundaries. We account for both of these issues by computing the label weights w_p^α as the minimum of:

1. A soft label mask, i.e. a 50×50 box-blurred version of the label mask, and
2. a soft image mask, i.e. a function which linearly decreases from 1 to 0 in regions that are close to image boundaries (within 15% of the input image diagonal).

1.4 Mesh Processing

The meshes generated by the algorithm in Section 4.4 of the main paper use a prohibitively large triangle count, since we have vertices for every pixel in the stitched panorama.

The first step to reducing the size is to decompose the mesh into charts that are not self-overlapping in the sense that they contain multiple vertices that occupy the same pixel position, but at different depth. We build the charts using a flood-fill like algorithm that propagates from a seed vertex as long as certain conditions are satisfied (not self-overlapping, chart size thresholds, etc.) All the charts are packed into a texture atlas (Figure 1).

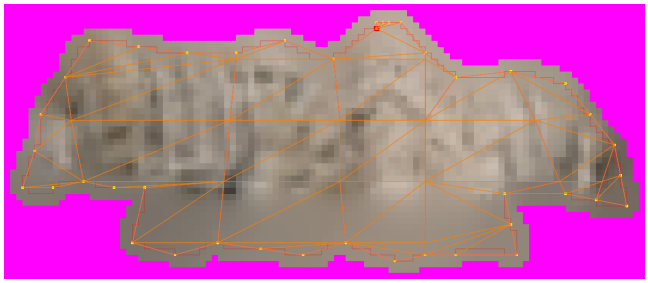

Figure 2: *Mesh simplification is performed in 2D on a per-chart basis.*

Next, we generate simplified meshes directly for each chart as described below. The algorithm below operates in 2D in the texture atlas chart domain. The final mesh is obtained by projecting the vertices into world space according to their image position and depth. We simplify the outline of the chart using the Ramer-Douglas-Peucker algorithm [Ramer 1972], making sure that chart boundaries that are shared between two charts are simplified in the same way. Next, we add vertical "stud" edges that are spaced and subdivided according to the desired tessellation level (the near-vertical edges in Figure 2). Finally, we generate a mesh for the chart by first decomposing it into monotone polygons, and then triangulating each monotone polygon [Berg et al. 2008].

References

BERG, M. D., CHEONG, O., KREVELD, M. V., AND OVERMARS, M. 2008. *Computational Geometry: Algorithms and Applications*, 3rd ed. ed. Springer-Verlag TELOS, Santa Clara, CA, USA.

RAMER, U. 1972. An iterative procedure for the polygonal approximation of plane curves. *Computer Graphics and Image Processing 1*, 3, 244–256.