Figure 2.1: The Lan scene. (a) Double image artifact when sampling rate is not enough for lightfield rendering. (b) Rendered result at the same viewpoint obtained by our lightfield rendering system augmented with depth, after double image elimination. (c) Only 3 bits are used to encode the depth for improving the rendering quality. It is not an accurate geometric reconstruction, but it provides 'just enough geometry' information for rendering anti-aliased images.
Figure 2.5: Spectral analysis of the epipolar plane image helps eliminating double image. (a) When the depth range is large and sampling rate is small, the replica may overlap with each other and cause aliasing. (b) The depth range in one patch is less than the whole image, and the patch can be reconstructed without aliasing. (c) The range of the frequency components in a patch is smaller, and this patch is more tolerant to depth error.
Figure 2.7: The NETFERT sceae. (a) Rendering with constant depth at nose. Double images at the neck and shoulder are evident. (b) Rendering with constant depth at neck. Now, double images of the nose are evident. (c) Rendering result with our lightfield system augmented with the recovered depth information. (d) The recovered depth map uses 4 bits to encode the depth. The depth does not correspond to accurate geometry; however, it produces satisfactory rendering result. Patch size is $16 \times 16$. (Note that the images shown here are cropped; the original image size is $512 \times 512$.)
Figure 2.8: The PLANT scene. (a) Rendering with constant depth at the background. Double images on the flowers are observed. (b) Rendering result with our modified lightfield, using the recovered depth patches.
Figure 3.1: An example of rendering with pop-up light fields. Rendering using the 5x5 Tsukuba light field data set is shown in the top left. Aliasing is clearly visible near front objects in the bottom left image because the input light field is sparse. The top row shows that the pop-up light field splits the scene gradually into 4 coherent layers, and achieves anti-aliased rendering as shown in the bottom right image.
Figure 3.2: A light field (with images $I_1$ and $I_2$) can be represented by a set of coherent layers ($L_1$ and $L_2$). A coherent layer is a collection of layered images in the light field. For instance, $L_1$ is represented by layered image $R_1^1$ (from $I_1$) and $R_2^1$ (from $I_2$). Each layered image has an alpha matte associated with its boundary. Part of the scene corresponding to each layer (e.g., $L_1$) is simply modeled as a plane (e.g., $R_1$).
Figure 3.3: Illustration of major steps in coherence matting. (a) The user specifies an approximate segmentation. (b) An uncertain region is added in between foreground and background. (c) A background mosaic is constructed from multiple undersegmented background images. (d) A coherent foreground layer is then constructed using coherent matting.
Figure 3.5: Comparison between video matting and coherence matting. (a) is a small window on one frame in the Plaza data (Figure 3.12). (b) and (c) are two alpha epipolar plane images (β-EPI) corresponding to the red line in (a), using the algorithm of video matting and coherence matting respectively. (d) and (e) are the alpha curves of two adjacent columns, which are marked as blue and red lines in (b) and (c). (d) corresponds to video matting, and shows a large jump at \( i = 13 \), which causes an accidental transparency within the face. (e) corresponds to coherence matting, which provides a more reasonable result.
Figure 3.7: The UI for Pop-up light field construction.
Figure 3.9: Local geometry
Figure 3.10: (a) The background mosaic operator uses the polygon lasso operator to segment the layer into regions. (b) The resulting background mosaic fills in many missing pixels in (a). Although (b) still has many missing pixels, it is enough for coherence matting of the foreground.
Figure 3.12: Result of pop-up light field rendering of the *Plaza* sequence rendered from a novel viewpoint (in the position midway between the 11th and 12th frames). The input consists of only 16 images. We have used 16 layers to model the pop-up light field.
Figure 3.13: Results on *Pokémon* 9 × 9: comparison of conventional light field rendering and pop-up light field rendering.

(a) (b) (c)

Figure 3.14: Results on sparse images taken with unstructured camera positions. (a) The global planar surface is set as a frontal-parallel plane in this view, (b) rendering result from another view with the global plane, (c) rendering result from the same view of (b) with local geometry.

Figure 3.15: Comparison of results on *furry rabbit*, with video matting (middle image) and with coherence matting (right image). The alpha matte from coherent matting is smoother than that from video matting in the rendering image. This effect can be more clearly observed in the accompanying video.
Figure 4.1: Lazy Snapping is an interactive image cutout system, consisting of two steps: a quick object marking step and a simple boundary editing step. In (b), only 2 (yellow) lines are drawn to indicate the foreground, and another (blue) line to indicate the background. All these lines are far away from the true object boundary. In (c), an accurate boundary can be obtained by simply clicking and dragging a few polygon vertices in the zoomed-in view. In (d), the cut out is composed on another Van Gogh painting.
Figure 4.2: Graph cut formulation for Object Marking. The graph cut algorithm is defined on $\mathcal{F}$, $\mathcal{B}$, and $\mathcal{U}$. All these nodes participate in the optimization process and are assigned a unique label, either foreground or background.
Figure 4.3: Our new graph cut algorithm works on the graph whose nodes are small regions from watershed segmentation.
Figure 4.4: Graph cut formulation for boundary editing. Only pixels in $\mathcal{F}$, $\mathcal{B}$, or $\mathcal{U}$ are considered in optimization. The polygon location is encoded as an energy term to guide the optimization to snap to user inputs.
Figure 4.6: The polygon soft constraint can override edge locations at low contrast regions. (a) Using the polygon as a soft constraint, the result (dotted line) is very close to the polygon (solid line). (b) Otherwise, the optimization is vulnerable to noise due to weak edges.
Figure 4.7: (a) The original image. (b) With polygon soft constraints, users can select which strong edge to snap. (c) Without polygon soft constraints, the same input polygon may produce erroneous edges because of the inherent edge ambiguity.
Figure 4.8: Hard vertex constraint. (a) A hard vertex modifies the uncertain region. (b) The graph cut segmentation result must pass through this point, while optimizing for other regions.
Figure 4.9: Local polygon editing: Once the vertices (green squares) are moved to new position, a local trimap is computed. (1) Uncertain region is the band of active vertices (orange band). (2) Foreground/Background region is computed using previous graph cut segmentation result in the band of inactive (light green band), and using new polygon (orange squares) for other regions (white regions).
Figure 4.10: Images used in usability study. The four images in the first row (A, B, C, and D) are for the first task. And the other four (E, F, G, and H) are for the second task.
Figure 4.11: (a) and (b) illustrate the average time of cutout process across fourteen subjects and eight images respectively. We normalize the time by that of Photoshop for each column, so that all data can be compared together. Moreover, for (c) and (d), we normalized the time and quality by the median of all samples of each image for all subjects, so that the data from different images can be aligned around 100% for comparison. (c) shows Lazy Snapping is consistently faster than Photoshop regardless of whether the user is a novice or expert. (d) plots the number of error pixels to the time for each experiment. Lazy Snapping is clustered at the lower left corner, indicating better quality in less time.
Figure 4.12: More Experiments. The numbers in the brackets denote the number of foreground markers, the number of background markers and the times to adjust polygon vertices, respectively. Each pair of images shows the marking lines for the first step and the final result. The polygons in the boundary editing step are not shown here. Please refer to the accompanying video to view the polygon editing process.