Additional Programming Details

1 Data Type

Each pixel i in the input image I has *rgb* colors. So we store them in a 3-vector I_i. Each color is represented using a 64-bit floating-point number scaled to between 0 and 1.

2 Kernel Estimation

In estimating the motion kernel using the energy defined in Equation (12) in the paper, when the input blurred image has relatively high resolution, there may not be enough space to store the entire matrix A in the memory. In this case, we only use part of the pixels in the image $\mathbf{L}$ (in Equation (12)) to estimate the blur kernel. Specifically, based on the observation that edges and textures are most informative for kernel estimation, we only extract the rows in A with the largest standard deviations (in experiments, the top $n\%$ rows are used where the value of n depends on the ratio of the memory size to the input image size) to solve Equation (12). It has been generally found that the more pixels we use to solve (12), the more stable our system is.

3 Kernel Shape Refinement

In each iteration after the kernel is estimated, to reduce the possible noise, we set the values of the kernel elements smaller than a threshold to zeros. Afterwards, the kernel values are normalized such that their sum remains 1.0. In our program, the threshold is set proportional to the maximum value in the kernel and the user can adjust the proportion with the parameter $kCutRatio$.

4 Final Image Refinement

After we complete the whole process to estimate the blur kernel, we perform a final step to suppress the possible ringings in the deblurred image. This step does not help refine the kernel.

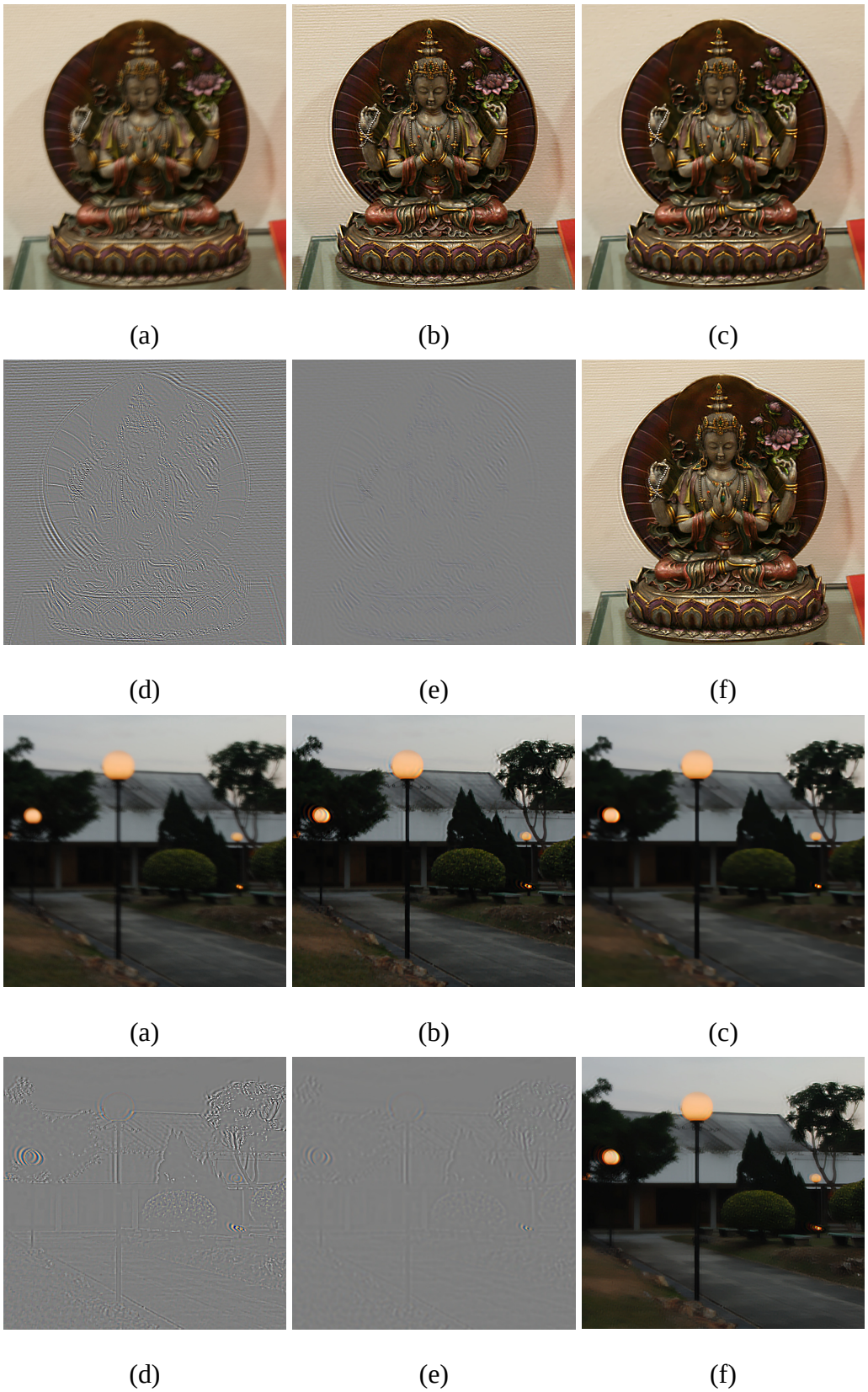

Figure 1: Illustration. (a) The blurred image. (b) The deconvolution result L_2. (c) The devolved image L_1 with large weights. (d) The difference map between (b) and (c). (e) is the bilateral filtered difference map. (f) The final result computed by subtracting (e) from (b).

Referring to Figure 1, two deconvolved images (L_1 and L_2) can be generated after the kernel is estimated. L_1 is obtained by setting weights λ_1 and λ_2 to 0.0016 and 80 respectively, as shown in Figure 1(c). L_2 is produced by using smaller weights ($\lambda_1 = 0.0002$ and $\lambda_2 = 10$) where finer image structures are recovered with stronger ringing artifacts, as shown in Figure 1(b). We then compute a difference map between these two images, as shown in Figure 1(d). It blends ringings with fine local structures. Bilateral filtering is applied to the map to remove small details, as shown in Figure 1(e). Finally, we subtract the ringing map (e) from L_2 to obtain the result in Figure 1(f).

In our program, to allow for higher flexibility, we construct 6 deconvolved images (L_1–L_6) after estimating the kernel. L_6 is with the smallest λ_1 set by the user (with parameter $noiseStr$), and the smallest λ_2 is set to 10. The other five λ_1's are $2\times noiseStr$, $4\times noiseStr$, $8\times noiseStr$, $16\times noiseStr$, and $32\times noiseStr$, respectively. The corresponding λ_2's are 20, 40, 80, 160, and 320. Then we build 5 consecutive ringing maps (R_1–R_5). R_5 is constructed using L_1 and L_6, R_4 is from L_1 and L_5, and so on and so forth. Each R is constructed as described above. Finally, the user can adjust the weights to combine these maps ($weight1$-$weight5$ in our executable). The final result L^* is computed as

$$L^* = L_6 - \sum_{0<i<6} weighti * R_i$$

In most of our experiments, $weight1$-$weight3$ are simply set to zeros.