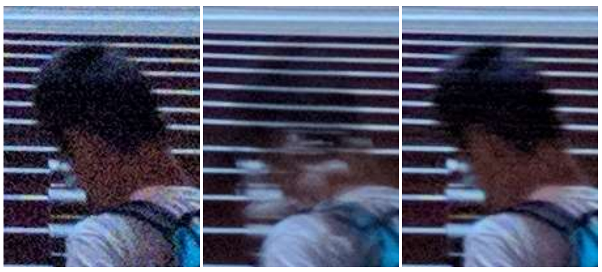

(a) *Reference frame* **(b)** *Temporal mean with alignment* **(c)** *Robust merge*

Figure 9: *Example of subtle ghosting artifacts produced by our merge algorithm due to large motion. Note the horizontal structure from the benches visible through the moving person's head in the robust merged result (c), which is not visible in the reference frame (a). The temporal mean with alignment is included to demonstrate the scale of motion and degree of alignment failure (b).*

Artifacts We have observed several classes of artifacts resulting from this system. First, this filter tends to fail to suppress noise around strong high contrast features, as shown in figure 8. This is a result of high contrast features having a non-sparse representation in the spatial DFT domain, reducing the effectiveness of spatial denoising.

Second, because our shrinkage function never fully rejects a poorly aligned tile, mild ghosting artifacts can sometimes occur, as shown in figure 9. In our experience, these ghosting artifacts are subtle, and very often are difficult to distinguish from motion blur.

Finally, our filter can occasionally produce ringing artifacts typically associated with frequency-domain filters. While ringing is largely mitigated by our windowing approach, in challenging situations classic Gibbs phenomenon can be visible, particularly after being amplified by sharpening and other steps in our finishing pipeline. Ringing is most frequently visible in the neighborhood of poorly-aligned clipped highlights, which exhibit high spatio-temporal contrast. In our experience, ringing has a negligible visual effect for most scenes.

6 Finishing

Aligning and merging the captured Bayer raw frames produces a single raw image with higher bit depth and SNR. In practice our input is 10-bit raw and we merge to 14 bits to preserve the precision gained from merging. This image must now undergo correction, demosaicking, and tone mapping–operations that would normally be performed by an ISP, but in our case is implemented are software and include the key additional step of dynamic range compression. In order of application, these operations are:

1. **Black-level subtraction** deducts an offset from all pixels, so that pixels receiving no light become zero. We obtain this offset from optically shielded pixels on the sensor.
2. **Lens shading correction** brightens the corners of the image to compensate for lens vignetting and corrects for spatially varying color due to light striking the sensor at an oblique angle. These corrections are performed using a low-resolution RGGB image supplied by the ISP.
3. **White balancing** linearly scales the four (RGGB) channels so that grays in the scene map to grays in the image. These scale factors are supplied by the ISP.
4. **Demosaicking** converts the image from a Bayer raw image to a full-resolution linear RGB image with 12 bits per pixel. We use a combination of techniques from Gunturk et al. [2005], including edge directed interpolation with weighted averaging, constant-hue based interpolation, and second order gradients as correction terms.
5. **Chroma denoising** to reduce red and green splotches in dark areas of low-light images. For this we use an approximate bilateral filter, implemented using a sparse 3x3 tap non-linear kernel applied in two passes in YUV.
6. **Color correction** converts the image from sensor RGB to linear sRGB using a 3x3 matrix supplied by the ISP.
7. **Dynamic range compression** See description below.
8. **Dehazing** reduces the effect of veiling glare by applying a global tone curve that pushes low pixel values even lower while preserving midtones and highlights. Specifically, we allow up to 0.1% of pixels to be clamped to zero, but only adjust pixels below 7% of the white level.
9. **Global tone adjustment**, to increase contrast and apply sRGB gamma correction, by concatenating an S-shaped contrast-enhancing tone curve with the standard sRGB color component transfer function.
10. **Chromatic aberration correction** to hide lateral and longitudinal chromatic aberration. We do not assume a lens model, but instead look for pixels along high-contrast edges, and replace their chroma from nearby pixels less likely to be affected by chromatic aberration.
11. **Sharpening** using unsharp masking, implemented using a sum of Gaussian kernels constructed from a 3-level convolution pyramid [Farbman et al. 2011].
12. **Hue-specific color adjustments** to make blue skies and vegetation look more appealing, implemented by shifting bluish cyans and purples towards light blue, and increasing the saturation of blues and greens generally.
13. **Dithering** to avoid quantization artifacts when reducing from 12 bits per pixel to 8 bits for display, implemented by adding blue noise from a precomputed table.

Dynamic range compression For high dynamic range scenes we use local tone mapping to reduce the contrast between highlights and shadows while preserving local contrast. The tone mapping method we have chosen is a variant of exposure fusion [Mertens et al. 2007]. Given input images that depict the same scene at different brightness levels, exposure fusion uses image pyramids to blend the best-exposed parts of the input images to produce a single output image that looks natural and has fewer badly exposed areas than the inputs.

Exposure fusion is typically applied to images captured using bracketing. In our pipeline we capture multiple frames with constant exposure, not bracketing. To adapt exposure fusion to our pipeline, we derive "synthetic exposures" from our intermediate HDR image by applying gain and gamma correction to it, then fuse these as if they had been captured using bracketing. We perform these extractions in grayscale, and we create only two synthetic exposures–one short and one long. The short exposure tells us how many pixels will blow out, and becomes the overall exposure used during capture, while the ratio between the short and long exposures tells us how much dynamic range compression we are applying. Both values come from our auto-exposure algorithm.

Fusing grayscale instead of color images, and using only two synthetic exposures, reduces computation and memory requirements. It also allows us to simplify the per-pixel blend-weights compared to those in the work by Mertens et al. [2007]. In particular, we use a fixed weighting function of luma that favors moderately bright pixels.