Burst photography for high dynamic range and low-light imaging on mobile cameras

Supplemental Material

Samuel W. Hasinoff Dillon Sharlet Ryan Geiss Andrew Adams
Jonathan T. Barron Florian Kainz Jiawen Chen Marc Levoy
Google Research

1 Brute-force L1 alignment

At the finest scale of our coarse-to-fine alignment strategy, we require an alignment technique which performs well given large tile sizes but a very small search radius. In this context, techniques such as our previously-described fast subpixel L2 alignment, or even simpler techniques such as phase correlation [Kuglin and Hines 1975] are outperformed by a well-implemented brute-force procedure which minimizes absolute residuals. We use absolute residuals instead of squared residuals because they map well to low level computer architectures. In contrast with squared residuals, absolute residuals require fewer bits for the same input data, enabling higher throughput. On ARM architectures, there is in fact an operation which computes an absolute difference and then accumulates, making brute-force computation of L1 distances very computationally light. This approach scales quadratically with the search radius, making it most appealing in contexts where the search radius is small.

2 Image alignment pyramids

The image pyramids we use for alignment are constructed to balance computational effort and the quality of the alignment results. The main control we have over quality versus computation is the distance in which we search at each pyramid level. Larger search distances allow larger downsampling factors between pyramid levels, which reduces the likelihood of the search getting trapped in a local minimum. Generally, larger search distances are impractical; however, the fast L2 norm search introduced in the paper enables us to use relatively large search areas of ± 4 pixels. However, while the fast L2 norm search is algorithmically efficient, it is still slower than a simpler search of a smaller area. Because of this, we use an L1 search (section 1) of ± 1 pixel for the bottom (high resolution) pyramid levels.

Putting all of this together, a typical alignment pyramid in our pipeline will look like:

- Bottom level of the pyramid, downsampled Bayer to grayscale. This pyramid level uses a ± 1 pixel search using the L1 alignment algorithm, using tiles of size 16×16.
- Because the bottom pyramid level uses a ± 1 pixel search, this pyramid level can be downsampled from the bottom level by a factor of 2. This pyramid level uses a ± 4 pixel search using the fast L2 alignment algorithm, using tiles of size 16×16.
- Because the previous level uses a ± 4 pixel search, this pyramid level can be downsampled by a factor of 4. Again, this pyramid level uses a ± 4 pixel search using the fast L2 alignment algorithm, using tiles of size 16×16.
- The last level is similar to the previous level; it is downsampled by a factor of 4, and uses the same search algorithm. However, at this point, we have downsampled so much that a tile size of 16×16 is effectively very large in the original image. Tiles covering a large effective area in the original image are useful to reduce the impact of noise on the search, and to avoid local minima. However, large tiles begin to fail to be able to approximate scene motion that is not strictly translational. At the same time, tiles that are large enough have largely eliminated the impact of noise on the results. Therefore, we reduce the tile size to 8×8.

3 Fast L2 residual computation

First, let us address the problem of taking two small sub-images, and computing a "distance image" which measures the mis-match between the two sub-images for all possible offsets (translations) of the images. Effectively, this will tell us the relative goodness of all possible translations of the two image, with our assumption being that the translation which minimizes our distance measure is a good estimate of the motion which transforms the first sub-image into the second sub-image.

First, let's begin our derivation with a simplified case. Consider the problem of computing the squared L2 distance between two vectors $\mathbf{a}$ and $\mathbf{b}$:

$$d_2 = \|\mathbf{a} - \mathbf{b}\|_2^2 \tag{1}$$

This distance can be rewritten by simply reorganizing the math:

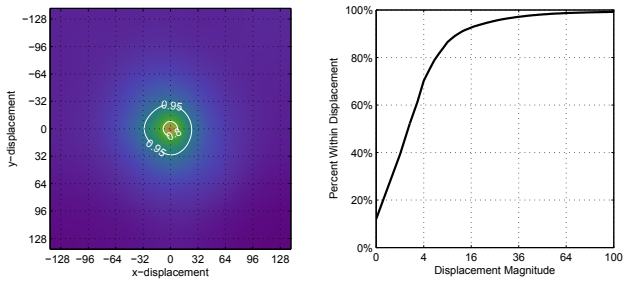
$$\begin{aligned} d_2 &= \|\mathbf{a} - \mathbf{b}\|_2^2 && (2)\\ &= \sum_i (a_i - b_i)^2 && (3)\\ &= \sum_i (a_i - b_i)(a_i - b_i) && (4)\\ &= \sum_i (a_i^2 + b_i^2 - 2a_i b_i) && (5)\\ &= \|\mathbf{a}\|_2^2 + \|\mathbf{b}\|_2^2 - 2\mathbf{a}^{\mathrm{T}}\mathbf{b} && (6) \end{aligned}$$

We see that the squared L2 distance between two vectors decouples into the squared L2 norm of each vector, minus twice the inner product of the two vectors.

Now let us consider an $n \times n$ image template T and a $m \times m$ image I, where $m > n$. That is, we have some large image, and we have an image template that we want to compare to each $n \times n$ sub-image of image. We will assume that our images and templates are grayscale for convenience, though our approach can generalize to color images trivially. More formally, we would like to compute a $(m - n + 1) \times (m - n + 1)$ "distance image" D_2 such that:

$$D_2(u, v) = \sum_{x=0}^{n-1} \sum_{y=0}^{n-1} \left(T(x, y) - I(x + u, y + v)\right)^2 \tag{7}$$

See figure 2 for examples of templates, images, and distance images.



(a) *2D alignment* **(b)** *1D alignment*

Figure 1: *Histogram of displacements found in a random sample of 100 bursts.*

Just as before, this distance calculation can be simplified:

$$D_2(u,v) = \sum_{x=0}^{n-1}\sum_{y=0}^{n-1} T(x,y)^2 \tag{8}$$

$$+ \sum_{x=0}^{n-1}\sum_{y=0}^{n-1} I(x+u,y+v)^2 \tag{9}$$

$$- 2\sum_{x=0}^{n-1}\sum_{y=0}^{n-1} T(x,y)I(x+u,y+v) \tag{10}$$

The first term depends only on T and not at all on u and v, and so it can be computed once and re-used when computing all values of $D_2(u,v)$. The second term can be computed for all values of (u,v) by box filtering $I(x,y)^2$, which can be done efficiently using sliding-window image filtering techniques or (somewhat less efficiently) using integral images. And the third term can also be computed for all values of (u,v) by cross-correlating I and T. Cross-correlation can be expensive to compute naïvely, but can be sped up significantly using fast Fourier transforms. From the convolution theorem, we know that:

$$\mathbf{a} \star \mathbf{b} = \mathcal{F}^{-1}\left\{\mathcal{F}\{\mathbf{a}\}^* \circ \mathcal{F}\{\mathbf{b}\}\right\} \tag{11}$$

where $\mathcal{F}\{\cdot\}$ is the Fourier transform, $\mathcal{F}^{-1}\{\cdot\}$ is the inverse Fourier transform, $\circ$ is the pointwise product (Hadamard product) of two vectors, and $\mathcal{F}\{\mathbf{a}\}^*$ is the conjugate transpose of $\mathcal{F}\{\mathbf{a}\}$. This naturally generalizes from one-dimensional signals to two-dimensional images.

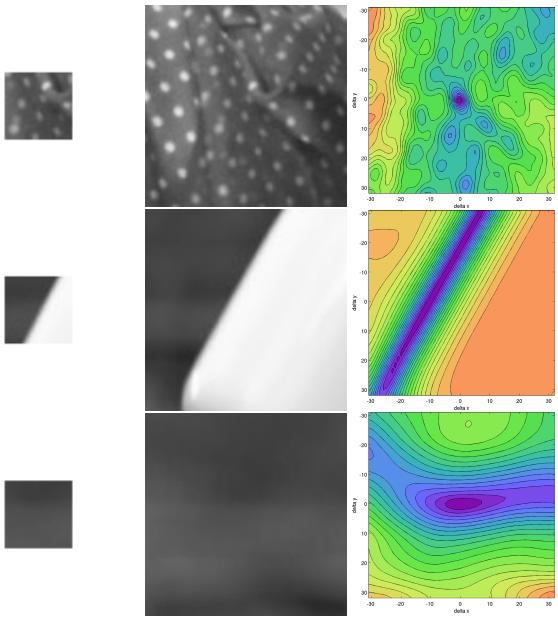
With these three observations, we can rewrite the computation of the distance "image" D_2 for all possible offsets (u,v) as:

$$D_2 = \|T\|_2^2 + \text{box}(I \circ I, n) - 2\left(\mathcal{F}^{-1}\left\{\mathcal{F}\{I\}^* \circ \mathcal{F}\{T\}\right\}\right) \tag{12}$$

where the first term is the sum of the squared elements of T, the second term is the squared elements of image I filtered with a box filter of size $n \times n$ (where the box filter is not normalized), and the third term is $-2\times$ the cross-correlation of I and T, computed efficiently using the fast Fourier transform.

4 Subpixel accurate translation

Given distance image $D_2(u,v)$, we would like to find the single best match between T and I by localizing the minimum of D_2. To produce subpixel-accurate translation estimations, we will use a bivariate quadratic function (a 2D polynomial), fit near the per-pixel minimum of D_2. This approach produces higher-quality translation



(a) *Template* T **(b)** *Image* I **(c)** *Distance* D_2

Figure 2: *Visualization of the distance surfaces (c) produced by computing the squared residual between a small image template (a) and every possible matching sub-image of a larger image (b).*

estimates than the standard approach of fitting two separable functions [Stone et al. 2001], as it jointly estimates the minimum in two dimensions rather than independently estimating each dimensions minimum produces more accurate results in the case when the axes of $D_2(u,v)$ are not isotropic in u and v, which they rarely are in practice (see figure 2c). More formally, we will approximate D_2 as follows:

$$D_2(u,v) \approx \frac{1}{2}[u;v]^{\mathrm{T}}\mathbf{A}[u;v] + \mathbf{b}^{\mathrm{T}}[u;v] + c \tag{13}$$

where $\mathbf{A}$ is a 2×2 positive semi-definite matrix, $\mathbf{b}$ is a 2×1 vector, and c is a scalar. A is assumed to be PSD because we expect the shape of D_2 near the minimum to be an upward-facing quadratic surface, rather than a saddle or a downward-facing surface. Let $(\hat{u},\hat{v})$ be the coordinate of the pixel in D_2 with the smallest distance value. We will consider the 3×3 pixel area around $(\hat{u},\hat{v})$ when fitting our quadratic function:

$$D_2^{sub} = \begin{bmatrix} D_2(\hat{u}-1,\hat{v}-1) & D_2(\hat{u},\hat{v}-1) & D_2(\hat{u}+1,\hat{v}-1) \\ D_2(\hat{u}-1,\hat{v}) & D_2(\hat{u},\hat{v}) & D_2(\hat{u}+1,\hat{v}) \\ D_2(\hat{u}-1,\hat{v}+1) & D_2(\hat{u},\hat{v}+1) & D_2(\hat{u}+1,\hat{v}+1) \end{bmatrix} \tag{14}$$

When fitting our bivariate polynomial, we will weight the pixels nearby the minimum according to a 3×3-sized patch of binomial weights:

$$W = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \tag{15}$$

With D_2^{sub} and W we can set up a least-squares problem with respect to the free parameters in our quadratic approximation $(\mathbf{A},\mathbf{b},c)$ and solve it. Without loss of generality, we will solve for a fit of D_2^{sub} which assumes that the center pixel has a (u,v) coordinate of $(0,0)$, and then shift the sub-pixel position that we will estimate by $(\hat{u},\hat{v})$. To construct our least-squares problem, we must first construct a

matrix $\mathbf{X}$ which contains a second-order polynomial expansion of the 9 (u, v) coordinates in our 3×3 patch:

$$\mathbf{X} = \begin{bmatrix} ½ & 1 & ½ & -1 & -1 & 1 \\ 0 & 0 & ½ & 0 & -1 & 1 \\ ½ & -1 & ½ & 1 & -1 & 1 \\ ½ & 0 & 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ ½ & 0 & 0 & 1 & 0 & 1 \\ ½ & -1 & ½ & -1 & 1 & 1 \\ 0 & 0 & ½ & 0 & 1 & 1 \\ ½ & 1 & ½ & 1 & 1 & 1 \end{bmatrix} \tag{16}$$

We will additionally construct a diagonal "weight" matrix from our (vectorized) 3×3 weight "image" in equation 15, and a RHS vector $\mathbf{y}$ as the vectorized version of D_2^{sub}:

$$\mathbf{W} = \mathrm{diag}(\mathrm{vec}(W)) \tag{17}$$

$$\mathbf{y} = \mathrm{vec}(D_2^{sub}) \tag{18}$$

With these we can construct a least-squares problem which corresponds to fitting our bivariate polynomial:

$$\arg\min_{\boldsymbol{\beta}} \left\| \mathbf{W}^{½}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) \right\|^2 \tag{19}$$

This is a conventional weighted least-squares problem, and so can be solved in a variety of ways. Ideally we would like to avoid repeatedly solving this linear system of equations for each tile in our alignment. This repeated fitting can be sped up significantly by taking advantage of the fact that $\mathbf{X}$ and $\mathbf{W}$ are constant across all tiles. We can therefore rearrange our linear system such that the polynomial fit parameters $\boldsymbol{\beta}$ are a linear function of the image patch $\mathbf{y}$ and a fixed matrix $\mathbf{F}$:

$$\boldsymbol{\beta} = \mathbf{F}\mathbf{y} \tag{20}$$

$$\mathbf{F} = (\mathbf{X}^{\mathrm{T}}\mathbf{W}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{W} \tag{21}$$

From this we can see that we can compute the parameters of the bivariate polynomial by simply taking the inner product of $\mathbf{y}$ (the 3×3 image patch of D_2, vectorized) by each row of a $\mathbf{F}$. This matrix $\mathbf{F}$ is equivalent to a filter bank, where each filter corresponds to some unknown parameter in $(\mathbf{A}, \mathbf{b}, c)$:

$$F_{A_{1,1}} = \begin{bmatrix} 1 & -2 & 1 \\ 2 & -4 & 2 \\ 1 & -2 & 1 \end{bmatrix} /4, \quad F_{A_{2,2}} = \begin{bmatrix} 1 & 2 & 1 \\ -2 & -4 & -2 \\ 1 & 2 & 1 \end{bmatrix} /4$$

$$F_{A_{1,2}} = F_{A_{2,1}} = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix} /4$$

$$F_{b_1} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} /8, \quad F_{b_2} = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} /8$$

$$F_c = \begin{bmatrix} -1 & 2 & -1 \\ 2 & 12 & 2 \\ -1 & 2 & -1 \end{bmatrix} /16 \tag{22}$$

With these filters we can estimate the free parameters of our quadratic approximation by simply taking the inner product of D_2^{sub} with these filters (assuming the error surface and the filter have been vectorized), or equivalently by computing the cross-correlation of D_2^{sub} with these filters:

$$\mathbf{A} = \begin{bmatrix} F_{A_{1,1}} \cdot D_2^{sub} & F_{A_{1,2}} \cdot D_2^{sub} \\ F_{A_{1,2}} \cdot D_2^{sub} & F_{A_{2,2}} \cdot D_2^{sub} \end{bmatrix} \tag{23}$$

$$\mathbf{b} = \begin{bmatrix} F_{b_1} \cdot D_2^{sub} \\ F_{b_2} \cdot D_2^{sub} \end{bmatrix} \tag{24}$$

$$c = F_c \cdot D_2^{sub} \tag{25}$$

This process is similar to the polynomial expansion approach of [Farnebäck 2002]. The constant shift c and its filter F_c are irrelevant for our subpixel minimum localization, but are included here for the sake of completeness.

Depending on the shape of D_2^{sub}, the estimated $\mathbf{A}$ may not be positive semi-definite, contrary to our initial assumptions. To fix this, after estimating the parameters of our quadratic, we first force the diagonal elements of $\mathbf{A}$ to be non-negative:

$$\mathbf{A}_{1,1} \leftarrow \max(0, \mathbf{A}_{1,1}) \tag{26}$$

$$\mathbf{A}_{2,2} \leftarrow \max(0, \mathbf{A}_{2,2}) \tag{27}$$

We then compute the determinant of $\mathbf{A}$:

$$\det(\mathbf{A}) = \mathbf{A}_{1,1}\mathbf{A}_{2,2} - \mathbf{A}_{1,2}^2 \tag{28}$$

if $\det(\mathbf{A}) < 0$, then we set the off-diagonal elements of $\mathbf{A}$ to be zero. These corrections give us a $\mathbf{A}$ which is guaranteed to be positive semi-definite.

With our quadratic approximation, we can now estimate the minimum of that quadratic. Doing so requires that we rewrite our quadratic in a different form by completing the square:

$$\frac{1}{2}\mathbf{x}^{\mathrm{T}}\mathbf{A}\mathbf{x} + \mathbf{b}^{\mathrm{T}}\mathbf{x} + c = \frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^{\mathrm{T}}\mathbf{A}(\mathbf{x} - \boldsymbol{\mu}) + s \tag{29}$$

Given a quadratic defined in the first form, we can convert it into the second form as follows:

$$\boldsymbol{\mu} = -\mathbf{A}^{-1}\mathbf{b} \tag{30}$$

$$s = c - \frac{\boldsymbol{\mu}^{\mathrm{T}}\mathbf{A}\boldsymbol{\mu}}{2} \tag{31}$$

For our particular bivariate case, this is equivalent to:

$$\boldsymbol{\mu} = -\frac{\left[\mathbf{A}_{2,2}\mathbf{b}_1 - \mathbf{A}_{1,2}\mathbf{b}_2, \quad \mathbf{A}_{1,1}\mathbf{b}_2 - \mathbf{A}_{1,2}\mathbf{b}_1\right]^{\mathrm{T}}}{\mathbf{A}_{1,1}\mathbf{A}_{2,2} - \mathbf{A}_{1,2}^2} \tag{32}$$

$$s = c - \frac{\mathbf{A}_{1,1}\boldsymbol{\mu}_1^2 + 2\mathbf{A}_{1,2}\boldsymbol{\mu}_1\boldsymbol{\mu}_2 + \mathbf{A}_{2,2}\boldsymbol{\mu}_2^2}{2} \tag{33}$$

Once we have recovered the location of the minimum of the quadratic $\boldsymbol{\mu}$, we can simply take that as the sub-pixel location of the minimum. Note that the fitted surface we produce treats the center pixel of D_2^{sub} as $(0, 0)$, so after fitting we need to add the per-pixel minimum location $(\hat{u}, \hat{v})$ into $\boldsymbol{\mu}$, which gives us the actual location of the minimum in D_2. In the presence of severe noise or very flat images, it is possible for the predicted sub-pixel minimum $\boldsymbol{\mu}$ to be very different from the observed per-pixel minimum $(\hat{u}, \hat{v})$, so in practice if we observe that the two are sufficiently different (more than 1 pixel removed) we set $\boldsymbol{\mu} = [\hat{u}; \hat{v}]$.

5 Example-based auto-exposure

In the following, we elaborate on implementation details for our example-based auto-exposure method. While this treatment should be of interest to someone implementing their own auto-exposure algorithm, our experience is that the quality of the labels and diversity of scenes in our database of about $5{,}000$ scenes dominates these engineering decisions. Moreover, the specifics of the scene descriptors we use is guided heavily by concerns for efficiency.

Exposure labels We label each scene in our auto-exposure database with two exposures, a short exposure for the highlights and a long exposure for the shadows, hand tuned to produce the most pleasing tone mapping result using our variant of exposure fusion [Mertens et al. 2007]. To represent these exposures we use the average pixel brightness of the two corresponding gamma-corrected images that serve as input to exposure fusion. Representing exposure in this way helps decouple our labeling from the absolute scene luminance or the sensitivity of the camera. As a refinement, we compute average pixel brightness differently for the two exposures. For the short exposure, we use the L2 norm (emphasizing the highlights); for the long exposure, we use the L0.5 norm (emphasizing the shadows) instead.

Scene descriptor Given a raw viewfinder frame as input, we compute a scene descriptor and use this to find the most similar scenes in our auto-exposure database. The descriptor we use encodes the basic information necessary for exposure decisions, but also has enough expressive power to help distinguish between categories of scenes. While our descriptor is simpler than typical descriptors used in computer vision for scene recognition, e.g., [Oliva and Torralba 2001], it plays a similar role in our system.

Note that consuming raw frames for auto-exposure produces more useful descriptors than consuming tone mapped ISP-processed frames. For the relatively wide-angle cameras on mobile devices, the pixel values at the corners are usually about 2 f-stops darker than those at the center, due to vignetting. Further, because the green channels are the most sensitive, the red and blue channel values are typically 1 f-stop darker than the green values. To take advantage of this extra dynamic range, our auto-exposure method leaves pixel values unclipped when applying white balance gains, lens shading correction, etc. In a normal imaging pipeline, this approach is unacceptable, as it can lead to false color shifts (often towards pink) in the highlights. In the context of auto-exposure, however, the unclipped signal is useful.

The core feature we use to build our descriptor is the spatially-weighted image brightness distribution, computed on an aggressively downsampled version of the image (25:1). In the course of downsampling, we perform a naïve demosaic by multiplexing the Bayer color planes, averaging the two green channels. For the 12–13 Mpix input we typically handle, this initial downsampling corresponds to a thumbnail-sized 160x120 linear RGB image.

Starting from this downsampled image, we subtract the black level and correct the color using the ISP-suggested white balance gains, lens shading correction, and 3x3 color correction matrix (all without clipping). To capture information at multiple spatial frequencies we make a copy of the downsampled image and apply a low pass filter to the copy. From each of these spatial scales, we downsample further (4:1) and extract two single-channel images: the maximum and the average of the RGB channels. We also compute spatial weights: a fixed weighting to favor the center of the image (3:1, with a radial falloff), and a strong boost (40:1) where faces are detected. In total this processing yields 4 single-channel linear 40x30 images and a corresponding weight map.

Next, we normalize these downsampled images, so that we can match similarly-shaped image brightness distributions across scenes with different luminance. To do this, we take the logarithm of image brightness values, compute the weighted mean of all unclipped values, and subtract this mean.

For efficiency, we implement two optimizations:

- We modify the second (4:1) downsampling operation to output *two* brightness values per pixel. This lets us downsample more aggressively while preserving higher-frequency brightness information. We start by averaging the 4x4 input pixels that contribute to each downsampled pixel, as usual. Then we perform a second pass, dividing the pixels into two groups: those brighter than the average, and those darker. Finally, we compute the average of each group and what fraction of the input samples correspond to each. This "split-pixel" representation lets us produce higher quality descriptors for a given level of downsampling.
- We represent the weighted brightness distribution of each downsampled image using 64 quantiles, rather than a typical histogram. The quantiles contain enough information to describe the scene, but store the information more compactly. For the long exposure case, each quantile in a set represents 1/64th of the weighted pixels. However, in the short exposure case, the top 8 quantiles, representing the highlights, correspond to fewer of the weighted pixels (about 1/512th each). This has the dual effect of adding precision to the highlights and giving them more weight in our distance metric.

Our final descriptor for auto-exposure is a 256-element vector, formed by concatenating the sets of 64 quantiles for each of the 4 downsampled images.

Distance metric To match an input image to our auto-exposure database, we use the L1 distance between descriptors. Because our descriptor is built from quantiles, this corresponds to the earth-mover's distance [Cohen and Guibas 1997] summed over the 4 underlying brightness distributions. In our implementation, we search exhaustively over our database of $5{,}000$ scenes, recording the L1 distance from the input to each labeled example.

Extra attention is required to handle clipped input pixels. Because our labeled examples are built from traditional HDR exposure bracketing, their histograms rarely include clipping, except when bright light sources are visible in the frame. However, our input is a single raw image captured during ISP-controlled viewfinding, so it will include clipped pixels in general. To address this, we track the fraction of pixels clipped (for at least one channel) in the input image and use this to determine which quantiles were contaminated. We then ignore these quantiles when computing the L1 distance.

Blending exposure labels To determine what short and long exposures to use for the input, we blend the exposure labels of the examples in our auto-exposure database, weighted by how closely they match the input. For a given labeled example i, we compute its weight as $\min\left(\max\left(2-\frac{d_i}{\min d_k},0\right),1\right)$, where d_i is the distance between the descriptors for input and the example. This scheme assigns a weight of 1 for the top match, and a weight of 0 for any example whose distance is double that or more. As described in the paper, we also ignore examples whose absolute luminance differs from the current scene by a factor of 8 or more. This helps retain perception of scene brightness, avoiding, for example, unnatural day-for-night renditions.

Target brightness to overall exposure So far, our example-based auto-exposure has provided us with a target pixel brightness (after gamma correction) for each of the short and long exposures. To translate these into overall exposures (the product of exposure time and gain) for the current scene, we use a lightweight simulation of our finishing pipeline. This simulation tells us how adjusting the overall exposure, relative to the parameters used to capture the input frame, influences the average image brightness of the final gamma-corrected image. Because the mapping between overall exposure and scene brightness is smooth and monotonic, we can invert this function with several steps of bisection.

6 Comparison with JPEG burst fusion

In our system, a key design decision is to use raw images as input to our align and merge algorithms, and then finish the raw merged result. Using raw images gives us both increased dynamic range and the ability to model sensor noise simply and accurately. By contrast, most previous burst fusion methods, e.g. [Liu et al. 2014; Dabov et al. 2007; Maggioni et al. 2012], consume JPEG images, which have already been finished by a photographic imaging pipeline.

To compare our system with such JPEG-based methods, we start from a dataset of 30 raw bursts and apply the same raw-to-JPEG finishing pipeline for all methods. For our method, this means running align and merge on raw bursts as usual, but substituting a different finishing pipeline. For JPEG-based methods, this means generating the JPEG input from the raw image bursts using the given finishing pipeline. This experimental approach lets us focus on the performance of the align and merge algorithms, without the confounding effect of the finishing pipeline, which can vary widely in tuning and overall quality across implementations.

Experimental details The 30-burst dataset we use for this evaluation is a subset of our larger dataset of several thousand raw bursts, to be released on publication, and includes the 10 bursts corresponding to figures 3-11 in the main paper. These bursts were captured for their coverage of different types of scenes, levels of motion, and brightness. The bursts were captured with 3 types of cameras, whose raw images are 12–13 Mpix.

For a raw-to-JPEG converter we used dcraw [Coffin 2016], followed by JPEG encoding at quality level 98, which effectively eliminates artifacts due to compression. While the pipeline implemented by dcraw is basic compared to commercial systems like Adobe Camera Raw, its predictability and lack of local tonemapping is an advantage for analysis. Furthermore, the AHD demosaicking method [Hirakawa and Parks 2005] implemented by dcraw works reasonably well in practice and is representative of the algorithms used by mobile ISPs. Color rendition in the results is somewhat compromised, due to limitations of both the DNG format and dcraw's treatment of color metadata, but the effect is uniform across methods. Also note that some bursts are underexposed. This follows from our capture strategy for HDR scenes, together with the conservative global tonemapping applied by dcraw, which sets the white level at the 99th percentile.

We compare our method against several state of the art JPEG-based burst fusion methods from the academic literature: two variants of the burst denoising method proposed by Liu et al. [2014], as well as CV-BM3D [Dabov et al. 2007]. For [Liu et al. 2014], the authors used their implementation to process our dataset, holding settings fixed for all results. For 3 manually selected bursts, the authors brightened the input using a global tonemapping curve, consistent with the approach proposed in [Liu et al. 2014] for handling "extreme low-light" scenes. For CV-BM3D, we ran the authors' Matlab implementation from the BM3D webpage[1]. Because this method does not include a mechanism for automatically setting the key noise level parameter, we ran a grid search over 17 different noise levels and hand-selected the result that visually seemed like the best tradeoff between noise reduction and loss of detail. We also tried comparing against V-BM4D [Maggioni et al. 2012], but the authors' implementation was not able to handle our 12–13 Mpix bursts.

To illustrate the performance of commercially available tools, we also compare against the JPEG-based "Merge to HDR Pro" feature of Adobe Photoshop CC 2015.1.2 [Adobe Inc. 2016] with "ghost removal" enabled, without further tonemapping. Although this Photoshop feature supports merging raw images as well, we found the HDR output unsuitable for input to dcraw because it already partially has photographic processing applied. In our experiments, Photoshop's JPEG-based and raw-based results were qualitatively similar, so we only include the JPEG-based results in this comparison. We also tried the "Photo Merge HDR" feature in Lightroom CC 2015, but we found that when the input images all have the same exposure, this feature has no denoising effect; each pixel in the output is apparently derived from a single input frame.

Summary of burst fusion results We include full-resolution image results for all methods over all 30 bursts in supplemental material, to allow detailed inspection at 1:1 magnification. Here we summarize our high-level findings, and in figures 3–5 present results several illustrative bursts. Crops in these figures are roughly 600×600, so we encourage the reader to zoom in aggressively (300% or more) to appreciate fine pixel-level differences.

- In general, all the methods we evaluated are capable at handling the smooth motion due to camera shake for reasonably bright scenes. With moving subjects, more complicated occlusion relationships, or lower-light scenes, performance begins to degrade.
- We found Photoshop's merging feature to be the most conservative of all methods, only implementing a very limited amount of denoising. Photoshop's most notable artifact is strongly colored ghosts in regions of clipped pixels. It also sometimes produces thin "echoes" at boundaries of heavy motion.
- Both variants of [Liu et al. 2014] show artifacts at motion boundaries, where differing amounts of merging leads to discontinuities in the level of retained noise. Both of these methods occasionally demonstrate ghosting artifacts as well. In certain scenes, we also found that the fast pixel-based variant of [Liu et al. 2014] also shows a significant loss of contrast, possibly due to issues in pyramid blending.
- CV-BM3D behaves robustly with respect to motion across the 30-burst dataset, producing typical wavelet denoising results, without any artifacts that can be definitively ascribed to motion. Depending on the noise level chosen, results may look either too noisy or oversmoothed, but a reasonable balance was generally available, at the cost of some detail. For higher noise levels, residual wavelet basis functions were sometimes visible at a pixel scale in the result, and isolated hot pixels were sometimes visually exaggerated by the denoising.
- Our align and merge method, like CV-BM3D, is very robust to motion, with no objectionable artifacts across the 30-burst dataset. When alignment does break down, our method degrades gracefully to the base frame and the resulting denoising sometimes has the appearance of motion blur. Our method generally dominates all other approaches in this comparison at both detail preservation and denoising. We attribute this success primarily to our robust merging approach and the accurate noise model enabled by processing raw images.

As a reminder, this evaluation is only a comparison of alignment and merging quality. Our paper represents an entire system for both low-light and HDR imaging, from capture strategy to finishing, which runs efficiently on mobile devices and reliably produces artifact-free results.

Runtime performance As table 1 shows, performance for these burst fusion methods vary widely over several orders of magnitude. While platform differences make comparing runtimes challenging,

[1] http://www.cs.tut.fi/~foi/GCF-BM3D

Figure 3: *Burst fusion results, for a low-light scene with moderate motion. For readability the crops have been made uniformly brighter. Readers are encouraged to zoom aggressively (300% or more). Our method denoises effectively while retaining the finest detail of all methods. In areas where alignment was unsuccessful (foreground person in rightmost crop), our results degrade to the appearance of motion blur. CV-BM3D recovers less detail and produces a slightly blotchy appearance, but behaves robustly with motion. Photoshop has very little denoising effect, likely due to overly conservative deghosting. Both variants of [Liu et al. 2014] demonstrate ghosting (face in the middle crops) and show discontinuities in the amount of denoising near motion boundaries (rightmost crop).*

Figure 4: *Burst fusion results, for an indoor scene with heavy motion. Readers are encouraged to zoom aggressively (300% or more). Our method denoises while preserving detail, and shows no merging artifacts despite heavy motion and blurry input. CV-BM3D performs comparably but retains somewhat more noise. Photoshop has very little denoising effect, likely due to overly conservative deghosting. The [Liu et al. 2014] results take most image content from a different and sharper frame, however the fused result is oversmoothed, shows severe posterization and blocky artifacts (face and foot, two leftmost crops), and also demonstrates ghosting (flesh tone over shirt in leftmost crop, orange wood texture over boot in rightmost crop).*

Figure 5: *Burst fusion results, for a bright outdoor scene with varying motion. Readers are encouraged to zoom aggressively (300% or more). For this bright and relatively low dynamic range scene, merging confers limited improvement over capturing a single input frame. Our method, CV-BM3D, and Photoshop perform comparably, with the denoising effect most visible in low-texture regions. However, Photoshop introduces strong colored ghosting artifacts in clipped pixel regions (middle crop). Both variants of [Liu et al. 2014] demonstrate blocky artifacts (near left man's back, leftmost crop) and sacrifice more fine detail than other methods. The pixel-based variant also produces hazy results for this burst (all crops), perhaps related to the pyramid blending approach.*

method	platform	type	cores used	average processing time (sec)
ours (align and merge)	Qualcomm Snapdragon 810	mobile	4+4 CPU	1.7
[Liu et al. 2014], pixel-based	i5 3.2GHz	desktop	1 CPU	2.2
[Liu et al. 2014], patch-based	i5 3.2GHz	desktop	1 CPU	40.7
CV-BM3D [Dabov et al. 2007]	i7 3.2GHz	desktop	1 CPU	300
Photoshop "Merge to HDR Pro"	i7 2.8GHz Macbook Pro	laptop	unknown	25

Table 1: *Align and merge runtime performance, averaged over our dataset of 30 bursts (section 6). The dataset consists of 264 images in total, each of which is 12–13 Mpix, corresponding to an average of 113 Mpix per burst.*

it is clear that both our method and the faster pixel-based variant of [Liu et al. 2014] are at least an order of magnitude faster than all other methods in the comparison. After adjusting for platform differences, our method and the pixel-based variant of [Liu et al. 2014] still have roughly comparable performance. However, since their implementation does not make use of SIMD they may have significant room for optimization.

7 Comparison with raw burst fusion

Burst fusion methods starting from raw input are less common than those starting from JPEG. To date, previous raw-based burst fusion methods have concentrated on the benefit of jointly demosaicking and merging multiple frames, e.g., [Farsiu et al. 2006; Heide et al. 2014], taking advantage of subpixel alignment to recover high frequency content lost to Bayer undersampling. While our raw-based approach is several orders of magnitude faster than these methods, our less sophisticated treatment of undersampling—aligning tiles to a multiple of 2 pixels and relying on our robust merge to handle aliasing issues—limits the detail we can recover at the finest scale.

To compare our system with previous raw burst fusion methods, we use the recent FlexISP method [Heide et al. 2014] as a representative example, and ran our method on their small dataset.

Experimental details The FlexISP dataset for burst fusion consists of 5 bursts, with resolutions ranging from 0.4–1.8 MPix, generated by downsampling or cropping higher-resolution input frames. Of these, 2 bursts are synthetic, created by warping and adding noise to a ground truth raw image. An additional 2 bursts are small crops from handheld sequences of 18 MPix dSLR images of static scenes. The final burst is a crop from a handheld portrait sequence captured with a 3 MPix machine vision camera. None of the bursts include significant scene motion or motion blur. Raw input frames were provided by the FlexISP authors.

Note that while the FlexISP paper [Heide et al. 2014] and supplemental material seem to imply otherwise, all FlexISP image results for burst fusion used only the first *8 images* in each burst as input [Heide and Kautz 2016]. Accordingly, we restricted our method to use the first 8 images of each burst as well.

The most direct comparison between our method and FlexISP would involve holding the raw-to-JPEG finishing pipeline constant (with the exception of the demosaicking integrated in FlexISP). Unfortunately, this kind of direct comparison is only possible for the synthetic bursts, for which the raw-to-JPEG finishing consists purely of demosaicking. For the other bursts in the FlexISP dataset, linear pre-tonemapped FlexISP results were not available, nor were we able to reproduce the color and tonemapping of the FlexISP perfectly. Despite this mismatch, visual comparisons are still informative.

Summary of burst fusion results We include results for all 5 bursts in supplemental material, to allow detailed inspection at 1:1 magnification. We also include a comparison with BM3D [Dabov et al. 2007] applied to the demosaicked first frame, which FlexISP uses as a denoising prior in all their results [Heide and Kautz 2016]. Here we summarize our high-level findings, and in figures 6–8 present results for 3 of these bursts.

- None of the results show artifacts due to motion. This is expected, given that the dataset does not include significant scene motion, motion blur, or parallax due to camera motion. All bursts in the FlexISP dataset except the portrait (figure 8) are of static scenes, and motion in the portrait scene is mild.
- Replacing our robust temporal merge with a simple temporal average produces significantly more denoising, without ghosting artifacts and with only a mild loss of detail. This shows that our alignment method works well for scenes with mild motion. The difference between these two merging strategies also illustrates how conservatively our robust merge behaves, particularly in scenes with very low SNR (figure 6).
- BM3D produces oversmoothed results, with residual wavelet basis functions sometimes visible. This level of denoising reflects the tuning chosen by the FlexISP authors; other tradeoffs between noise and detail are possible. Perhaps the smoother tuning makes BM3D more effective as a denoising prior for FlexISP.
- For scenes with very low SNR (figure 6), our method denoises less aggressively than either FlexISP or BM3D. In part this is an aesthetic choice. For the 12–13 Mpix images our method normally handles, luminance noise at this spatial scale is generally not objectionable. Our reduced denoising also follows from a conservative merging approach designed to handle real-world scene motions.
- For scenes with low SNR (figures 6–7), FlexISP recovers an impressive amount of fine detail not visible in the input image. Neither BM3D nor our method recovers as much fine detail, despite our weaker denoising. This demonstrates the value of subpixel alignment and joint demosaicking to account for undersampling. However, it is unclear to what extent these results generalize to more realistic scenes. Because the scenes are planar (figure 6) or nearly so (figure 7), subpixel alignment is well explained by the global homography fit that FlexISP uses to initialize its alignment [Heide and Kautz 2016].
- For the one real scene with moderate SNR (figure 8), our method recovers a similar amount of detail as FlexISP, and joint demosaicking does not appear to confer an advantage. While it is difficult to generalize from a single burst, this may reflect the fact that the burst is closer in SNR and spatial scale to the input normally handled by our system.

Runtime performance FlexISP reports timings for a 16-image [sic] 0.4 MPix burst (6.4 Mpix total). Since all FlexISP burst fusion image results used BM3D as a denoising prior [Heide and Kautz 2016] we compare against corresponding timings. For the GPU implementation of FlexISP, optimized with a reduced number of

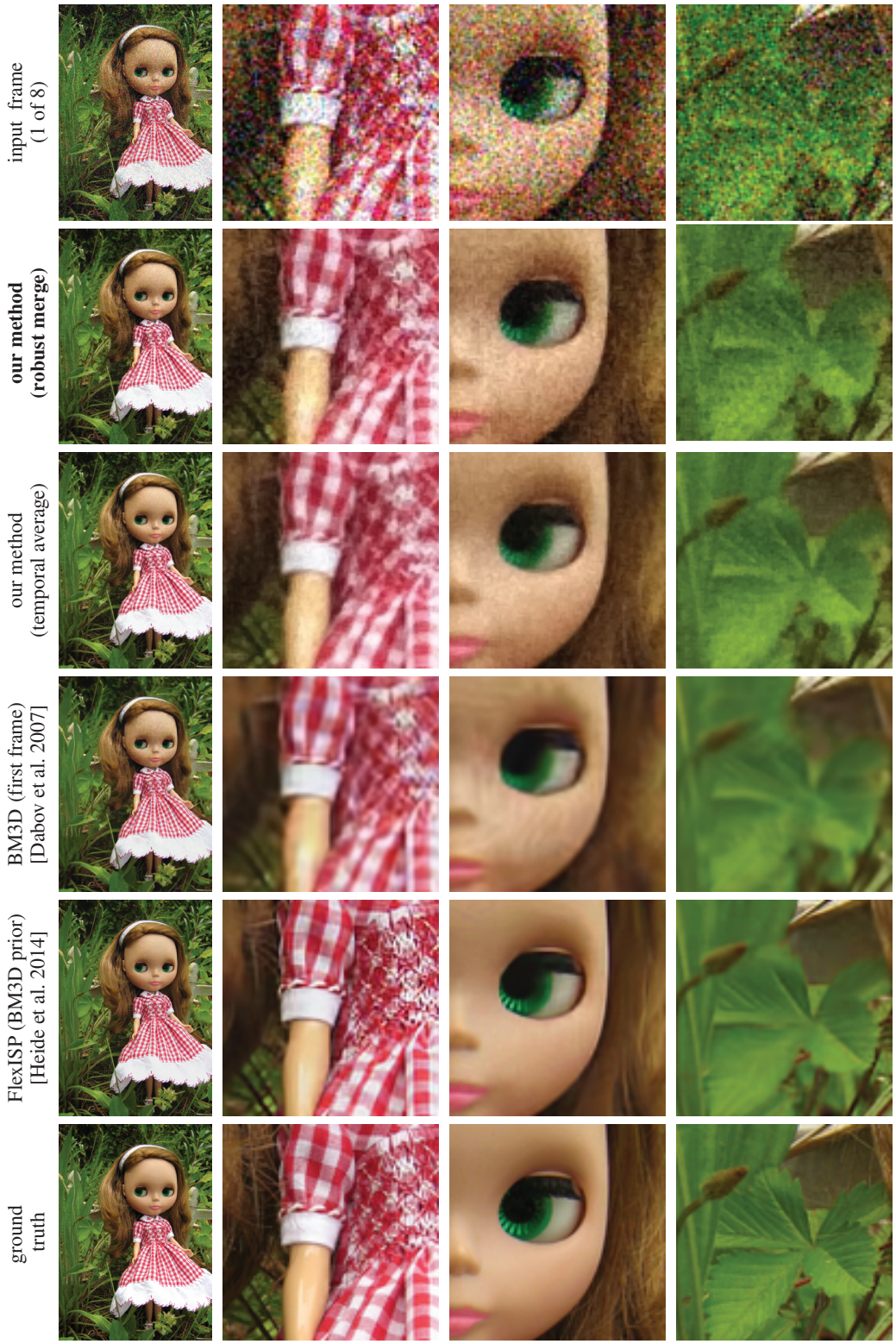


Figure 6: *Raw burst fusion results, for a very noisy 0.4 Mpix synthetic example. Each burst frame was generated by warping the ground truth image with a global homography, then adding synthetic noise. Readers are encouraged to zoom in. FlexISP recovers fine detail not visible in other methods (isolated strands of hair, fine texture on dress, striations on leaves). Replacing our robust temporal merge with a simple temporal average yields significantly stronger denoising, showing how conservatively our robust merge behaves in low-SNR scenes like this one. While our method recovers less detail, it is several orders of magnitude faster than FlexISP.*

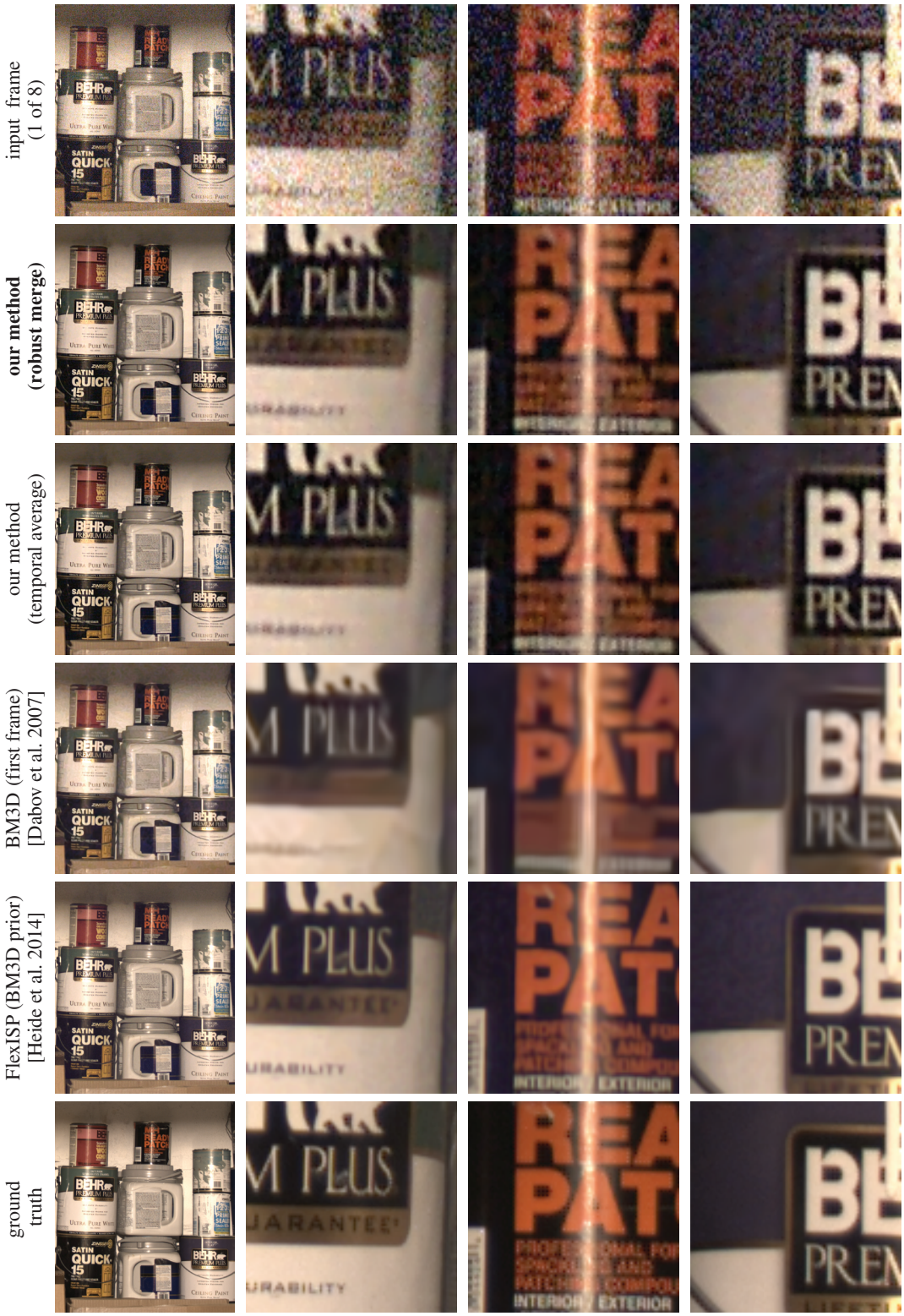


Figure 7: *Raw burst fusion results, for a 0.8 Mpix crop of a very dark static indoor scene, captured with a 15 Mpix dSLR at ISO 12800. This cropped portion of the scene is nearly planar. Readers are encouraged to zoom in. FlexISP recovers fine detail not visible in other methods (small text on paint cans, not visible with other methods). Replacing our robust temporal merge with a simple temporal average increases the strength of denoising at the expense of slight loss of detail. While our method recovers less detail, it is several orders of magnitude faster than FlexISP.*

Figure 8: *Raw burst fusion results, for a 1.8 Mpix crop of a dark indoor portrait, captured with a 3 Mpix machine vision camera, with mild natural scene motion. Readers are encouraged to zoom in. In this example, our method recovers a similar amount of fine detail as FlexISP. The improvements in fine contrast demonstrated by our method are due in part to the sharpening in our finishing pipeline. Replacing our robust temporal merge with a simple temporal average increases the strength of denoising and reduces chroma aliasing (resolution chart, middle crop) at the expense of slight loss of detail. In the FlexISP result, the green pixel artifacts on the forehead (leftmost crop) are due to a hot pixel in the input. Aliasing artifacts due to demosaicking are visible in all methods, with our method showing mild chroma aliasing, and BM3D and FlexISP showing a cross-hatching pattern (right area of resolution chart, middle crop).*

iterations and an approximation to accelerate BM3D, they report 0.82 sec on a 250W desktop GPU and 16.7 sec for a 11W tablet. Assuming linear scaling with number of input pixels, to match the 133 Mpix bursts handled by our system (table 1), the adjusted performance of FlexISP is 14.5 sec on desktop and 295 sec on tablet. By comparison, our system takes 1.8 sec for a corresponding amount of work (1.7 sec for align and merge, from table 1, plus 0.1 sec for demosaicking). In summary, our method on a 2W mobile CPU is 8.0x and 164x faster than FlexISP on desktop and tablet respectively. On a performance per watt basis, our method is about 1000x and 900x more efficient than FlexISP on desktop and tablet respectively.

References

ADOBE INC., 2016. Photoshop CC 2015.1.2, http://www.adobe.com/creativecloud.html.

COFFIN, D., 2016. dcraw 9.26, http://www.cybercom.net/~dcoffin/dcraw.

COHEN, S., AND GUIBAS, L. J. 1997. The earth mover's distance: lower bounds and invariance under translation. Tech. Rep. STAN-CS-TR-97-1597, Stanford University.

DABOV, K., FOI, A., AND EGIAZARIAN, K. 2007. Video denoising by sparse 3D transform-domain collaborative filtering. *EUSIPCO*.

FARNEBÄCK, G. 2002. *Polynomial Expansion for Orientation and Motion Estimation*. PhD thesis, Linköping University, Sweden.

FARSIU, S., ELAD, M., AND MILANFAR, P. 2006. Multi-frame demosaicing and super-resolution of color images. *TIP*.

HEIDE, F., AND KAUTZ, J., 2016. Personal communication.

HEIDE, F., STEINBERGER, M., TSAI, Y.-T., ROUF, M., PAJK, D., REDDY, D., GALLO, O., LIU, J., HEIDRICH, W., EGIAZARIAN, K., KAUTZ, J., AND PULLI, K. 2014. FlexISP: A flexible camera image processing framework. *SIGGRAPH Asia*.

HIRAKAWA, K., AND PARKS, T. W. 2005. Adaptive homogeneity-directed demosaicing algorithm. *TIP*.

KUGLIN, AND HINES. 1975. The phase correlation image alignment method. *The International Conference on Cybernetics and Society*.

LIU, Z., YUAN, L., TANG, X., UYTTENDAELE, M., AND SUN, J. 2014. Fast burst images denoising. *SIGGRAPH Asia*.

MAGGIONI, M., BORACCHI, G., FOI, A., AND EGIAZARIAN, K. 2012. Video denoising, deblocking, and enhancement through separable 4-D nonlocal spatiotemporal transforms. *TIP*.

MERTENS, T., KAUTZ, J., AND REETH, F. V. 2007. Exposure fusion. *Pacific Graphics*.

OLIVA, A., AND TORRALBA, A. 2001. Modeling the shape of the scene: A holistic representation of the spatial envelope. *IJCV 42*, 3, 145–175.

STONE, H. S., ORCHARD, M. T., CHANG, E.-C., AND MARTUCCI, S. 2001. A fast direct Fourier-based algorithm for subpixel registration of images. *TGRS*.