Computational Long Exposure Mobile Photography

ERIC TABELLION, NIKHIL KARNAD, NOA GLASER, BEN WEISS, DAVID E. JACOBS, and YAEL PRITCH,
Google Research, USA

(a) Foreground blur examples

(b) Background blur examples

Fig. 1. At the tap of the shutter button, our smartphone camera system captures, processes and outputs both conventional and long exposure corresponding photographs in a few seconds, as shown in the top row. Examples of foreground blur captured hand-held are shown in (a), and examples of background blur captured without precise tracking, are shown in (b). Our computational long exposure photography pipeline handles both use cases fully automatically.

Long exposure photography produces stunning imagery, representing moving elements in a scene with motion-blur. It is generally employed in two modalities, producing either a foreground or a background blur effect. Foreground blur images are traditionally captured on a tripod-mounted camera and portray blurred moving foreground elements, such as silky water or light trails, over a perfectly sharp background landscape. Background blur images, also called panning photography, are captured while the camera is tracking a moving subject, to produce an image of a sharp subject over a background blurred by relative motion. Both techniques are notoriously challenging and require additional equipment and advanced skills. In this paper, we describe a computational burst photography system that operates in a hand-held smartphone camera app, and achieves these effects fully automatically, at the tap of the shutter button. Our approach first detects and segments the salient subject. We track the scene motion over multiple frames and align the images in order to preserve desired sharpness and to produce aesthetically pleasing motion streaks. We capture an under-exposed burst and select the subset of input frames that will produce blur trails of controlled length, regardless of scene or camera motion velocity. We predict inter-frame motion and synthesize motion-blur to fill the temporal gaps between the input frames. Finally, we composite the blurred image with the sharp regular exposure to protect the sharpness of faces or areas of the scene that are barely moving, and produce a final high resolution and high dynamic range (HDR) photograph. Our system democratizes a capability previously reserved to professionals, and makes this creative style accessible to most casual photographers.

More information can be found on our project webpage: https://motion-mode.github.io/.

CCS Concepts: • **Computing methodologies** → **Computational photography**; **Computer vision problems**; **Computer graphics**.

Additional Key Words and Phrases: machine learning, mobile computing

ACM Reference Format:
Eric Tabellion, Nikhil Karnad, Noa Glaser, Ben Weiss, David E. Jacobs, and Yael Pritch. 2023. Computational Long Exposure Mobile Photography. *ACM Trans. Graph.* 42, 4, Article 48 (August 2023), 15 pages. https://doi.org/10.1145/3592124

Authors' address: Eric Tabellion, Nikhil Karnad, Noa Glaser, Ben Weiss, David E. Jacobs, Yael Pritch, Google Inc., 1600 Amphitheatre Parkway, Mountain View, CA, 94043, USA.

1 INTRODUCTION

Mobile photography is ever present in consumers' daily lives and is often superseding traditional photography. Using image burst capture and post-processing techniques, modern mobile phones' imaging pipelines produce very high quality results, providing high dynamic range tone mapping, exceptional low light performance and simulating depth-of-field bokeh effects, which were previously achievable only with much bigger and heavier cameras and lenses.

Despite these outstanding improvements, long exposure mobile photography remains poorly treated to the best of our knowledge. Existing solutions don't help users produce results where moving and static scene elements appear blurry and sharp respectively. This juxtaposition of sharp against blurry is a key property of a compelling image, that cannot be achieved by simply exposing a hand-held camera sensor for a longer duration.

Traditional long exposure photography is typically performed in one of two ways, according to the scene and situation. One approach produces a foreground blur effect (e.g. silky waterfall, light trails, etc.) over a sharp background, using very long exposure times that can last up to several seconds. This requires using a tripod, as even a slight camera shake can cause undesired loss of background sharpness. Additionally, a neutral density (ND) filter must be added to the lens, to avoid over-exposing the sensor. A second approach, called panning photography, produces a rendition with a sharp moving subject over a background that is blurred with motion relative to the subject. It is achieved by tracking the moving subject with the camera, while keeping the shutter open with the exposure time increased modestly, e.g. half a second, and the aperture slightly reduced to avoid over-exposing the image. The photographer must track the subject motion as precisely as possible to avoid undesired loss of subject sharpness, while also pressing the shutter button at the right moment. Both approaches require advanced skills, practice and choosing the camera shutter speed manually, taking into account how fast the scene is moving to achieve the desired result.

The main contribution of this paper is a computational long exposure mobile photography system, implemented in two variants, which democratize the two aforementioned use cases. It is implemented in a new camera mode called "Motion Mode" on Google Pixel 6 and 7 smartphones, which allows the user to easily capture these effects, without the need for a tripod or lens filter, nor the need to track the moving subject precisely or at all. Our method is fully automatic end-to-end within each variant: after the user chooses which of foreground or background blur result they wish to produce, we generate long exposure 12 megapixel photographs at the tap of the shutter button, while compensating for camera and/or subject motion, thereby preserving desired background and subject sharpness. The main components of our system are:

- Capture schedule and frame selection, producing normalized blur trail lengths independent of scene or camera velocity,
- Subject detection that combines gaze saliency with people and pets face region predictions, and tracking of their motion,
- Alignment of input images to cancel camera hand-shake, stabilize the background in the presence of moving foreground elements, or to annul subject motion while producing pleasing background motion blur trails,
- Dense motion prediction and blur synthesis, spanning multiple high resolution input frames and producing smooth curved motion blur trails with highlight preservation.

Furthermore, our system architecture, which includes several neural networks, performs efficiently on a mobile device under constrained compute and memory budgets, implementing an HDR imaging pipeline that produces both related conventional and long exposure results in just a few seconds.

2 RELATED WORK

2.1 Mobile Phone Computational Photography

Many computational photography advances in recent years define today's mobile photography capabilities. The work from Hasinoff et al. [2016] describes a mobile camera pipeline that captures, aligns and merges bursts of under-exposed raw images. Combined with the work of Wronski et al. [2019], they are able to strongly improve the Signal to Noise Ratio (SNR), dynamic range and image detail, overcoming the limitations of small smartphone sensors and lenses. Our system is built on top of such a computational imaging foundation.

To handle very low light situations without using a flash, Liba et al. [2019] employ a scene motion metering approach to adjust the number of input burst frames and determine their exposure time. Similarly, we adjust the frame capture schedule based on scene motion, estimated when the shutter button is pressed, for the purpose of normalizing the resulting amount of motion-blur.

Since the small camera lenses used on smartphones cannot produce shallow depth-of-field effects optically, Wadhwa et al. [2018] design a synthetic bokeh approach, that relies on a semantic person segmentation neural network, with the intent to isolate a subject from a distracting background. Our system is analogous, as we strive to isolate a subject from the relative motion of the background, while attempting to emphasize the dynamic nature of the scene.

2.2 Auto-tracking a Subject (background blur)

Determining the subject of a background blur capture is a hard problem. Many synthetic long exposure pipelines avoid it altogether by requiring manually tagging the subject region, or using a heuristic such as the middle region of the image [Lancelle et al. 2019; Luo et al. 2018; Mikamo et al. 2021]. In contrast, we present a pipeline which determines the subject region automatically by predicting visual saliency and face regions.

Using saliency-driven image edits to highlight a main subject from a distracting background was introduced in [Aberman et al. 2022]. Existing methods to detect and track subject motion over time include [Stengel et al. 2015], which use gaze saliency prediction to detect the subject and optical flow to track its motion, and [Mikamo et al. 2021], which require the user to specify the subject region using a bounding box and similarly track its motion. In our work, we detect the subject using a combination of gaze saliency and semantic segmentation using a highly efficient mobile architecture inspired by [Bazarevsky et al. 2019]. We track its motion using feature tracking, and introduce an alignment regularization term to result in more visually pleasing motion-blur trails, which are more consistent with overall subject motion.

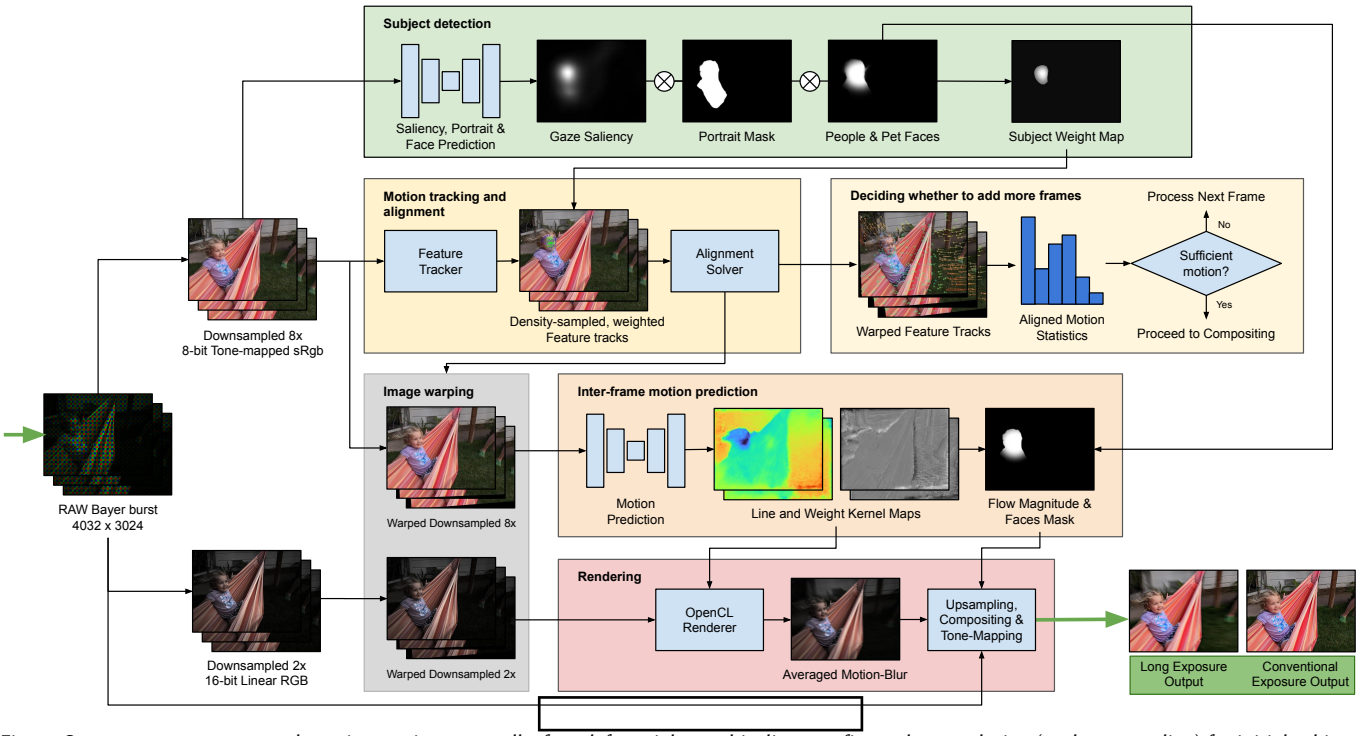


Fig. 2. Our system processes raw burst images incrementally, from left to right on this diagram, first at low resolution (8x downsampling) for initial subject predictions, feature tracking, motion analysis and motion prediction. Images are also processed at half resolution (2x downsampling) to synthesize motion-blur before being upsampled back to full 12 megapixel resolution for compositing and tone-mapping the final results. More detail is provided in Section 3.

2.3 Stabilizing the Background (foreground blur)

Images captured by handheld cameras are often shaky and may often contain parallax. In the foreground-blur case, we need to stabilize the background to keep the resulting regions sharp. This can be solved for using structure-from-motion (SFM) techniques [Liu et al. 2014] to construct a 3-d representation of the scene [Hartley and Zisserman 2004], then a stabilized camera path can be solved for in 3-d, and the scene finally re-rendered using the new smooth path [Liu et al. 2009]. However, these techniques rely on 3-d reconstruction, for which a fast and robust implementation is challenging [Liu et al. 2011]. At the other end of the spectrum are 2-d stabilization techniques that use much simpler motion models such as global homographies or affine transformations [Morimoto and Chellappa 1998]. These techniques are fast and robust, but cannot model effects such as parallax, rolling-shutter, or lens distortion. There is a large body of work that extends these 2-d methods, such as using gyroscopic sensors only [Karpenko et al. 2011], gyroscopes with face detection [Shi et al. 2019], targeted crops [Grundmann et al. 2011] and trajectory filtering [Liu et al. 2011]. Our method is analogous to techniques that start with 2-d feature trajectories to estimate per-frame global transformations and refine this estimate with spatially varying image warps [Liu et al. 2013; Zaragoza et al. 2013] to achieve the desired trade-off between speed and robustness.

2.4 Synthesizing Motion Trails

There is a large body of prior work on synthesizing motion-blur, in the context of non-photorealistic rendering [Lee et al. 2009], stop-motion animation [Brostow and Essa 2001], or 3D computer graphics rendering in real-time [Rønnow et al. 2021] or offline [Lehtinen et al. 2011; Navarro et al. 2011]. There is work describing single photograph post-processing interactive applications to create artistic motion effects [Luo et al. 2018, 2020; Teramoto et al. 2010] or that process multiple previously stabilized images and can achieve non-physical renditions [Salamon et al. 2019].

Our work compares more directly to prior work on computational long exposure from multiple photographs or video frames. Lancelle et al. [2019] describe a pipeline that can handle both foreground and background blur effects, but requires substantial user interaction to handle all the cases. Like other approaches, they require significant compute time in an offline processing application, as they rely on expensive optical-flow based image warping or frame interpolation, to synthesize smooth motion-blur spanning the input frames pairwise. In contrast, our pipeline is fully automatic, is integrated in a smartphone photo camera and produces the result in a few seconds at 12 megapixel resolution. To synthesize motion-blur, we use a line kernel prediction neural network, derived from [Brooks and Barron 2019], combined with a GPU rendering algorithm that can handle the input image alignment warping implicitly, while producing smooth and curved motion-blur trails spanning multiple input frames.

3 SYSTEM OVERVIEW

A diagram of our computational long-exposure system is shown in Figure 2. The stream of captured raw images is processed incrementally at two different resolutions through four stages, each

corresponding to a row of the diagram in Figure 2: initial subject detection, motion analysis, motion prediction and rendering. The initial saliency and face prediction stage (Section 4.2) computes the main signals for our subject detection, producing a normalized weight map. The motion analysis stage is responsible for tracking (Section 4.3) and aligning (Section 4.4) a set of feature points corresponding to the detected subject or to the background, and for selecting frames based on motion statistics (Section 4.5). The motion prediction stage (Section 4.6) predicts dense line kernel and weight maps, that are used in the rendering stage (Section 4.7) to produce smooth motion-blur spanning a given input frame pair. The final compositing stage (Section 4.8) layers the final results while preserving the sharpness of important areas in the final image.

The first three stages use as their input, images that have been tone-mapped and converted to sRGB, downsampled by a factor of 8 to a low resolution of 504 x 376. This resolution is chosen to achieve low latency when processing frames, which is dominated by the dense motion prediction neural network. This also ensures that the receptive field covers most practical motion disparities in the input full resolution images. The last stage however, uses the intentionally under-exposed burst raw images converted to 16-bit linear RGB. The high bit-depth is necessary to preserve the scene's high dynamic range during rendering, i.e. to avoid saturating the highlights and banding artifacts in the shadows. Images are downsampled by a factor of 2 as a trade-off that preserves enough detail in the final result while operating within a reduced memory footprint.

The incremental processing loop converts and downsamples an additional burst frame at each iteration, feeding the three last stages of our pipeline and resulting in an accumulated averaged motion-blur image. The loop stops when the frame selection criteria is reached, using an estimate of motion-blur trails' length. We then composite the final results, while upsampling images back to full resolution. At the end of our pipeline, images are converted to a low dynamic range 8-bit representation, using tone-mapping to preserve the high dynamic range visual appearance of the scene.

4 IMPLEMENTATION

4.1 Burst Capture

Our camera system captures frames at a rate of 30 frames per second, using fully automatic aperture, shutter speed, and focus settings. These settings may adjust dynamically to scene changes across frames, and our system performs well with all but abrupt variations.

In the background blur case, we target scenes with fast moving nearby subjects. When tapping the shutter button, the most recently captured frame is used as the base frame to produce the conventional exposure [Hasinoff et al. 2016], as it depicts the peak of the action chosen by the user. Up to 8 additional frames in the past may then be selected by our algorithm to produce the background blur effect.

When producing a foreground blur effect, we target scenes with a much larger range of scene motion velocity, including slow and far away moving content. To produce a compelling effect, this requires extending the capture for a duration up to several seconds, according to scene motion. When the shutter button is pressed, we quickly analyze the scene motion statistics using the last 5 frames seen through the camera viewfinder and automatically determine a subsequent capture duration that aims to satisfy our frame selection criteria. We use a lightweight variant of the motion tracking and image alignment described in the next few sections, that operates in under 50ms, to compute an estimate of scene velocity. With this estimate, under a constant velocity assumption, we trivially derive a capture duration that yields the desired blur trail length (see Section 4.5). Given the extended capture duration of up to 7 seconds, we also derive a frame processing rate, to select an evenly distributed subset of up to 12 captured frames for processing, balancing the compute budget with a suitable temporal sampling rate. The captured frames selected for processing are queued up for immediate concurrent processing by the following stages, thereby hiding some of the processing latency during capture.

4.2 Automatic Subject Detection

In the background blur case, we want the effect of a fixed subject with the rest of the world blurred behind them. Therefore, we automatically detect and track the main subject, and align the input frames to negate its motion. The subject is represented as a weight map, and is used in solving for the inverse subject motion alignment.

The main subject is first predicted using the proxy task of attention saliency. For this task, we use a mobile-friendly 3-level U-Net with skip connections, with an encoder comprised of 15 BlazeBlock with 10 base channels [Bazarevsky et al. 2019] and a corresponding decoder made of separable convolutions and bi-linear upsampling layers. It is distilled from a larger model trained on the SALICON dataset [Jiang et al. 2015]. To focus on the peak of saliency in our signal, we re-normalize predicted values to the interval $[0, 1]$ and zero out values below a threshold (we empirically chose 0.43).

The saliency signal tends to peak on the subject center, so we complement it with a face signal, which helps keep subject faces sharp, which is especially important in subjects with complex articulated motion. We compute the face signal by first predicting human, cat, and dog face regions, then feathering the resulting regions using a `smootherstep` falloff [Ebert et al. 2003], and lastly masking it by a whole-subject segmentation similar to that of [Wadhwa et al. 2018].

We combine the saliency and face signals as follows, to produce the subject weight map with per pixel weight $w = s\ (1 + f)$, where $s \in [0, 1]$ is the saliency signal value and $f \in [0, 1]$ is the face signal value, followed by a re-normalization to the interval $[0, 1]$. The face signal is also used in the compositing step to preserve face sharpness, as described in Section 4.8.

4.3 Motion Tracking

We use a feature tracking library based on [Grundmann et al. 2011] for extracting the motion tracks used in subsequent image alignment. Motion track statistics are also used to select frames, to determine when sufficient motion has been captured in the scene.

Subject tracking in background blur requires a high concentration of tracks on the subject for stable, high quality alignments. As a latency optimization, we use rejection sampling over an image grid with cells of 5 x 5 pixels each, to generate feature tracks with density proportional to the subject weight map (Section 4.2). We only attempt to extract feature tracks in cells where a sampled uniform random variable $v \in [0, 1]$ is smaller than the corresponding average track-weight at that grid location.

Fig. 3. City scene. Top: Traffic moving through a busy city intersection with foreground motion vectors (red) and background motion vectors (green). Bottom: Our foreground blur (left) and background blur (right) results.

4.4 Image Alignment

Given the feature track correspondences from Section 4.3, we first estimate global transforms to align all the frames to our reference frame. This cancels out overall camera motion, including both hand-shake and sweeping motions used to track subjects. The remaining image alignment stages are specific to the desired motion-blur effect: foreground or background blur. For the purpose of illustration, we pick an example scene that could be rendered as either: a taxicab passing through a busy city intersection as shown in Figure 3.

4.4.1 Foreground blur. To keep the background as sharp as possible, we must account for spatial effects such as parallax, rolling shutter, and lens distortion. After applying global transforms, we compute a residual vector as the position difference between a transformed tracked feature and its corresponding position in the base frame. We then use the residuals to estimate local refinement transforms on a grid of vertices across the image. The resulting spatially varying warp cancels motion in the background while retaining motion in the foreground, producing sharp backgrounds as in Figure 3 (bottom-left).

In [Zaragoza et al. 2013], the authors weight points by distance from each grid vertex to produce a spatially varying as-projective-as-possible warp. Our approach to placing a grid and estimating local transforms is similar, but we weight our points uniformly and use a hard cut-off for point inclusion during local similarity transform estimation for better performance. The estimation is controlled by the support radius of each vertex (shown as magenta circle in Figure 4), i.e. the maximum distance from a vertex that a feature point needs to be for inclusion in the local refinement estimation. We found that setting this radius to 1.5 times the cell size of the mesh grid and using a grid size of 8 x 6 cells, was large enough for the local refinement transforms to vary smoothly across the entire field-of-view, yet small enough that disparate scene objects from different parts of the scene do not affect each other. The estimated transforms are applied to each vertex to then generate a spatially varying mesh that aligns the background of any frame to that of the reference frame. To optimize latency, the application of this mesh is folded into the downstream rendering stage by passing a texture of local 2-d displacement vectors to the GPU.

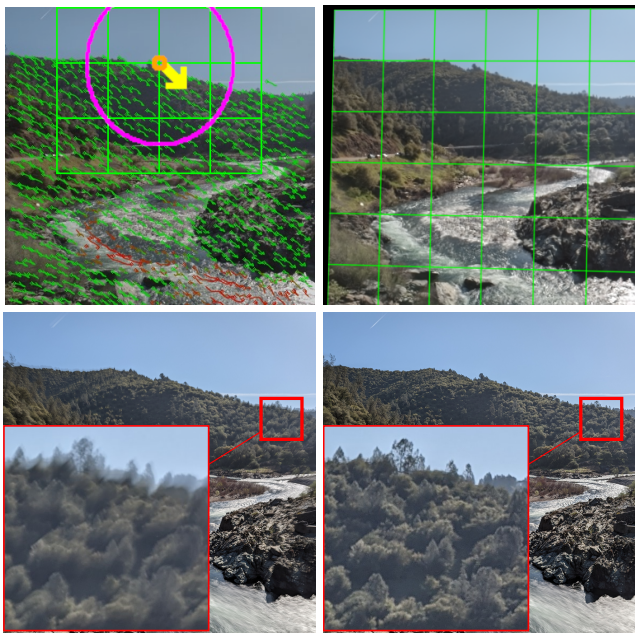


Fig. 4. Spatially varying warp. Top-left: The background flow vectors (green) inside one of the mesh vertex (orange) support regions (magenta) contribute to its local displacement (yellow). Top-right: The resulting mesh and spatially varying warp that aligns the background to that of the reference frame. Foreground blur results using only a single global homography transform (bottom-left) and using our spatially varying mesh warp (bottom-right). Insets are displayed at 5x magnification.

4.4.2 Background blur. In this case, as shown in Figure 3 (bottom-right), we want the foreground to be as sharp as possible. We use the subject mask from Section 4.2 to select the subset of feature tracks that correspond to the foreground subject. With this as a starting point, we further use spectral clustering to select the most salient motion cluster to help discern the right motion segment to track and to remove outliers [Porikli 2004]. This is especially useful for articulated subjects, such as a running person whose limbs move differently from their torso.

The goal of our objective function is to balance two goals: (1) subject sharpness: minimize the overall reprojection error of the salient point correspondences, and (2) temporal smoothness: keep the transformed background motion vectors as parallel as possible to those from the previous time step, as shown in Figure 5 and Figure 16.

Given a pair of frames i and j with point correspondences $\mathbf{x}_i$ and $\mathbf{x}_j$, we define a similarity transform that scales uniformly ($s_{j,i} \in \mathbb{R}^1$), rotates ($\mathbf{R_{j,i}} \in SO(2)$) and translates ($\mathbf{t_{j,i}} \in \mathbb{R}^2$) 2-dimensional points from frame i to frame j as follows.

$$\hat{\mathbf{x}}_\mathbf{j} = s_{j,i}\mathbf{R_{j,i}}\mathbf{x_i} + \mathbf{t_{j,i}} \tag{1}$$

For simplicity, we omit the from-and-to indices from the transform parameters s, $\mathbf{R}$ and $\mathbf{t}$ and define our objective function as follows.

$$\underset{s,\mathbf{R},\mathbf{t}}{\text{minimize}} \quad \lambda_f E_f(s,\mathbf{R},\mathbf{t}) + \lambda_b E_b(s,\mathbf{R},\mathbf{t}) \tag{2}$$

The scalars λ_f and λ_b are the relative weights of the objective function terms. The subject sharpness term E_f is defined on the foreground points $\mathbf{x} \in \mathcal{X}_f$ as the L-2 norm of the reprojection error of transformed point correspondences (using Eq. 1) as follows.

$$E_f(s,\mathbf{R},\mathbf{t}) = \sum_{\mathbf{x}\in\mathcal{X}_f} \|\mathbf{x_j} - s\mathbf{R}\mathbf{x_i} - \mathbf{t}\|_2 \tag{3}$$

The background term E_b is used as a temporal smoothness prior to penalize background flow vectors that are not parallel to their counterparts from the previous frame pair. Given three distinct frame indices i, j and k, this is defined using vector dot product as a measure of parallelism as follows:

$$E_b(s_{i,k},\mathbf{R}_{i,k},\mathbf{t}_{i,k}) = \sum_{\mathbf{x}\in\mathcal{X}_b} \underset{L_1}{\text{smooth}}\left(\frac{|\mathbf{v}_{i,j}\cdot\mathbf{v}_{j,k}|}{\|\mathbf{v}_{i,j}\|\|\mathbf{v}_{j,k}\|}\right) \tag{4}$$

where $\mathbf{v}_{p,q} = \hat{\mathbf{x}}_q - \mathbf{x}_q$ is the directional displacement vector between a point in the reference frame q and its transformed point correspondence from frame p. The smooth-L_1 loss function from [Huber 1964] is used to prevent vanishing gradients during the optimization.

$$\underset{L_1}{\text{smooth}}(x) = \begin{cases} 0.5x^2, & \text{if } |x| < 1 \\ |x| - 0.5, & \text{otherwise} \end{cases} \tag{5}$$

Without loss of generality, we chose $i = 0$ as the fixed reference frame index, and set $k = j + 1$ in Eq. 4 to estimate the transforms incrementally for each new frame pair $(j, j+1)$ in a single forward pass through the frame sequence. We solved this non-linear optimization problem using the Ceres library [Agarwal et al. 2022]. We use values 1 and 10 for λ_f and λ_b respectively, to balance the effect of both error terms.

The temporal regularization term E_b is not defined for the first frame pair, which we handle as a special case using a constraint on the scale and rotation parameters, s and $\mathbf{R}$, from Eq. 2. Our key observation comes from subjects of background blur shots that undergo rotational motion in the image plane. Simply inverting this rotation produces undesirable multiple sharp regions in the result, as shown in Figure 5. In traditional panning photography, it is uncommon to attempt the rotation blur effect and exceedingly difficult to achieve subject sharpness in this manner (i.e. rotating the camera around an axis centered away from the subject). Instead, it is typically done panning the camera, tracking the overall trajectory of the subject, and our method aims for these outcomes.

To estimate the initial scale and rotation, we use the integrated estimation technique from [Zinßer et al. 2005]. We then constrain the estimated rotation, $\mathbf{R}$ in Eq. 2, to prevent any additional sharp regions from detracting away from the sharp subject. We empirically found that constraining the roll angle to 25% of its estimated value helps make the blur field more linear, as shown in Figure 5, with only the subject kept sharp as desired. More examples of image alignment for background-blur scenes are provided in Section 5.2.

(a) Unconstrained rotation (b) Constrained rotation

Fig. 5. Undesirable rotations. Fully inverting the subject's rotation (a) gives us an undesirable result with an additional sharp region below the subject. Even though the rotating blur can be a fun effect, the sharpness region at the center of rotation attracts the viewer's attention away from the main subject and degrades the subject separation from the background, both of which goes against photography composition rules. We alleviate this by constraining the estimated rotation (b).

4.5 Frame Selection

Our system uses a frame selection mechanism that computes an estimate of motion-blur trails' length, to decide when the incremental frame processing outer-loop should stop (see Section 3). First, we use the transformations computed by the alignment solver to transform the motion feature tracks to the reference space of the base frame, where they align spatially with the corresponding tracked features' motion-blur trails in the output image. The length of each aligned track can then be computed, and we use a high percentile of the track length distribution as an estimate of overall blur trail length. This estimate is finally compared to a constant target setting, to decide whether the frame selection criteria is satisfied.

We measure the track length in percentage of image diagonal, a metric that is largely insensitive to image resolution or aspect-ratio. In the case of foreground blur, our criteria is for the 98^{th} percentile to reach a target of 30%, producing relatively long and smooth blur trails for the fastest moving object. In the background blur case, we use the 80^{th} percentile and a target of 2.8%, producing short blur trails for a larger area of the background, aiming to preserve subject sharpness and avoid losing the context of the surrounding scene. These settings were derived empirically, iterating over large collections of input bursts.

4.6 Motion Prediction

Once the input low-resolution images are aligned, we feed them through a motion-blur kernel-prediction neural network, one input frame pair at a time, predicting a pair of line and weight kernel maps at each iteration. The low-resolution kernel maps are used to synthesize motion-blur segments at half resolution, spanning the corresponding input frames, as described in Section 4.7.

The motion prediction model is responsible for predicting the parameters of two spatial integrals along line segments, which approximate the temporal integral defining the averaging of colors seen through each motion-blurred output pixel, during the corresponding time interval. We use a model based on [Brooks and Barron 2019], with further modifications that improve the trade-off between performance and image quality, allowing us to fit within a reasonable memory and compute budget on mobile devices.

Their mathematical formulation predicts weight maps W_i per input frame i in a given image pair k, with $N = 17$ channels, which are used to weigh each corresponding texture sample along the predicted line segments. We simplify this model by predicting only a single channel, used to weigh the result of the integral from each input frame. An example gray-scale map can be seen in Figure 2, showing that the network predicts approximately equal weights everywhere across input images, except in areas of dis-occlusion where the weights favor the result from one of the two inputs. This simplification significantly reduces system complexity and memory use, and allows for more of the network capacity to be devoted to predicting the line segments.

In addition, we eliminate artifacts due to the predicted line segments' endpoint error [Zhang et al. 2016], causing them to meet imperfectly at the end of the spanned time interval, and resulting in very noticeable artifacts in the middle of blur trails, as illustrated in Figure 6. To avoid this issue, we scale the input image texture samples further by a normalized decreasing linear ramp function w_n, that favors samples close to the output pixel and gradually down-weighs samples further away along each predicted line segment. The intensity of the output pixel (x, y) for the input frame pair k is:

$$I_k(x,y) = \sum_{i\in\{k,k+1\}} \frac{W_i(x,y)}{\sum_{n=0}^{N-1} w_n} \sum_{n=0}^{N-1} w_n \, I_i(x_{in}, y_{in}) \qquad (6)$$

with $w_n = 1 - n/N$, and with sampled positions:

$$x_{in} = x + \left(\frac{n}{N-1}\right) \Delta_i^x(x,y) \quad \text{and} \quad y_{in} = y + \left(\frac{n}{N-1}\right) \Delta_i^y(x,y)$$

where Δ_i are the predicted line segments.

We also modify the network architecture as follows. First, we replace the leaky ReLU convolution activations throughout, with a parameterized ReLU [He et al. 2015], where the slope coefficient is learned. Next, to avoid common checkerboard artifacts [Odena et al. 2016], we replace the 2x resampling layers to use average pooling for downsampling, and bi-linear upsampling followed by a 2x2 convolution. This results in a model labeled "Ours-large" analyzed in Section 5. Furthermore, to improve the balance between the number of floating operations, number of parameters and receptive field, we further reduce the U-Net model topology to only 3 levels, where each level is using a 1x1 convolution, followed by a ResNet block [He et al. 2016] with four 3x3 convolution layers. This results in a model labeled "Ours" with significantly fewer learned parameters.

As shown in Figure 6, the ramp function w_n brings a significant benefit to our learned single weight model, as it causes the predicted line segments to span spatially in each input image, the equivalent of the full time interval being integrated. When our model is trained with this term ablated, resulting in the model "Ours-abl.", the network predicts line segments that span approximately half of the time interval on each side, causing the noticeable discontinuity in the middle of blur trails. More examples can be found in the model comparison analysis provided in Section 5.

4.7 Rendering

The line and weight kernel maps output by the motion prediction network are used by a renderer that synthesizes the motion-blurred image. The renderer is implemented in an OpenCL kernel, which runs very efficiently on the mobile device's GPU, leveraging the hardware texturing units while adaptively sampling texture lookups in the half resolution input images (the number of texture samples N is adjusted proportionally to the length of the predicted line vectors). Motion prediction and rendering iterations can be performed one input frame-pair at a time, producing piecewise-linear motion-blur trails. Kernel maps are up-sampled from low to half-resolution by using bi-linear texture lookups.

(a) Blur from single image pair (b) Without weight ramp (Ours-abl.) (c) With weight ramp (Ours)

Fig. 6. Motion prediction model ramp function ablation. (a) Rendering of a motion-blurred moving train synthesized from a single input image pair. Both a motion-blurred closeup of the front of the train and corresponding input image pair overlaid with a vector field visualization representing the predicted line segments, is shown in (b) and (c) using the models "Ours-abl." and "Ours", i.e. without and with the ramp function w_n, respectively. In image (b)-left, the blue arrows indicate the full span of motion blur trails and the red arrows showcase the gap discontinuities in the middle of blur trails that are most noticeable.

4.7.1 Spline interpolation. Piecewise-linear motion interpolation may introduce jagged visual artifacts in motion trails. To interpolate the motion more smoothly, we interpolate the inferred instantaneous flow Δ_i between frames using cubic Hermite splines.

The instantaneous flow δ_i at each pixel is inferred by constructing a vector $H(\Delta_i^+, \Delta_i^-)$ parallel to $(\Delta_i^+ + \Delta_i^-)$, with magnitude equal to the harmonic mean of $|\Delta_i^+|$ and $|\Delta_i^-|$. Superscripts + and − refer to time directions. If Δ_i^+ and Δ_i^- deviate by an angle θ from a straight-line path, the vector is further scaled by a factor of $(\theta/\sin\theta)$ for smaller angular deviations (< 90°), tapering this adjustment back towards zero for larger deviations (where the path doubles back acutely) to avoid singularities. These correctional factors reduce overshoot, and keep the parametric spline speed more stable for regions of moderate curvature.

$$\delta_i = H(\Delta_i^+, \Delta_i^-)\ (\theta/\sin\theta) \times \begin{cases} 1, & \theta \le \pi/2 \\ 1 - (2\theta/\pi - 1)^4, & \theta > \pi/2 \end{cases} \qquad (7)$$

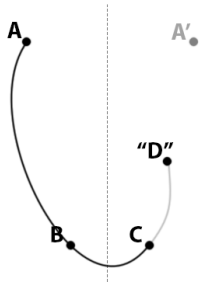


Fig. 7. Our spline extrapolation strategy. See Section 4.7.1.

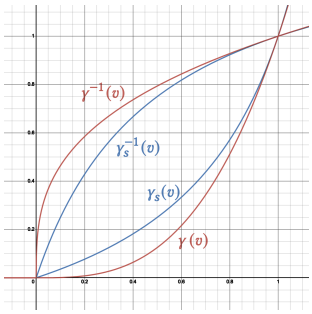


Fig. 8. Comparison of traditional gamma γ vs. our soft gamma γ_s. See Section 4.7.3.

For the accumulated blur of I_k on the interval [k .. k+1] for output position (x, y), we solve for a parametric 2D cubic spline path ρ(x, y, t) satisfying four constraints:

- ρ(x, y, 0) = (x, y)
- ρ(x, y, 1) = (x, y) + $\Delta_i^+(x, y)$
- $\rho'(x, y, 0) = \delta_i(\rho(x, y, 0))$
- $\rho'(x, y, 1) = \delta_{i+1}(\rho(x, y, 1))$

We then accumulate the blur along this path by sampling uniformly in parameter space, normalizing the weight of each sample to compensate for the non-uniform spatial sampling in image space in order to ensure spatially uniform brightness along motion trails. At the burst endpoints we extrapolate the flow beyond the first and last frames by attempting to preserve the curvature of the flow through those endpoints. As shown in Figure 7: if 'C' represents the final frame in a burst, a motion trail position at the "next" frame D is extrapolated by reflecting A in the line bisecting BC (constructing A'), then clamping the magnitude of CA' to |BC| to form CD. The flow at C is then inferred from points {B,C,D}.

4.7.2 Frame accumulation. In practice, the blur is accumulated in several passes: two passes per frame pair, weighted to fall off linearly between one frame and the next. For an output pixel at position p at frame I_i, the blur between frame I_i and I_{i+1} is accumulated by using the aforementioned flow splines to determine the projected position p' in frame I_i at relative time t. For K frame pairs in the burst, 2K such passes (K forward, K backward) are computed and summed to produce the final blur result. For each temporal direction:

$$I(x,y) = \sum_{i=0}^{K-1} \sum_{n=0}^{N-1} I_i(\rho_i(x,y,t_n))\ |\rho_i'(x,y,t_n)|\ w_n \tag{8}$$

4.7.3 Soft Gamma Colorspace. Very bright highlights (e.g. car headlights) tend to saturate the camera sensor, resulting in their blurred motion trails becoming unrealistically dim even when processed in linear colorspace. The clipping is due to the finite range of the input sensor, and the brightness loss becomes noticeable when the clipped input highlight energy is distributed (i.e. synthetically motion-blurred) over many output pixels.

To work around this limitation, we process the blur in an intentionally non-linear colorspace, using an invertible gamma-like "soft gamma" function γ_s, shown in Figure 8, on the interval [0..1].

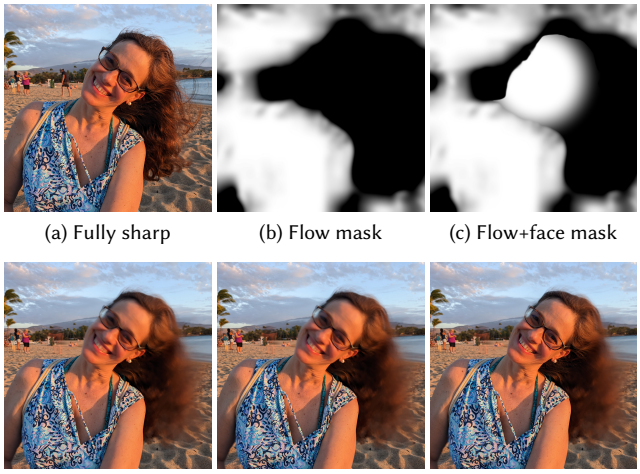
This adjusts the brightness curve in the opposite direction from a linear-to-sRGB color transformation, emphasizing highlights without crushing shadows, allowing the nonlinear frames to be stored with usable fidelity in 16-bit buffers. The function is applied to the warped downsampled 2x buffers on creation, using a value of 3.0 for k, and is later inverted (by reapplying with $k = 1.0/3.0$) after accumulating the blur for all frames. (See ablation in Section 5).

$$\gamma_s(v) = \frac{v}{v + (1-v)\ k} \quad \approx \quad v^k \tag{9}$$

This is homologous to the Bias family of functions in [Schlick 1994], but our reparameterization in Eq. 9 makes clearer the connection to the corresponding gamma curve with exponent k. The idea of processing the blur in the modified colorspace was inspired by the Ordinal Transform technique in [Weiss 2006]. Our goal is similar to the clipped highlight recovery technique in [Lancelle et al. 2019], which in comparison uses a more abrupt discontinuous highlight boosting function, that may further clip the signal.

4.8 Compositing

The synthetically blurred image described in Section 4.7 is computed at half resolution to satisfy device memory and latency constraints. Accordingly, even perfectly aligned, zero-motion regions of the blurred image will lose detail due to the upsampling of the result computed at half resolution. To preserve details, we composite the blurred image with a maximally sharp regular exposure where we expect things to be sharp. Two categories need this protection: 1) stationary scene content, and 2) semantically important subjects with little movement, as shown in Figure 9.



(a) Fully sharp (b) Flow mask (c) Flow+face mask

(d) Fully blurred (e) Flow protected (f) Flow+face protected

Fig. 9. Compositing. The regular exposure (a) is composited with the synthetically blurred image (d) to produce our final output with details preserved. A flow-based mask (b) protects nearly motionless image regions to produce (e), note the preserved texture detail in the subject's dress. Further including face signals in the mask (c) also preserves moving, but semantically important image regions (f).

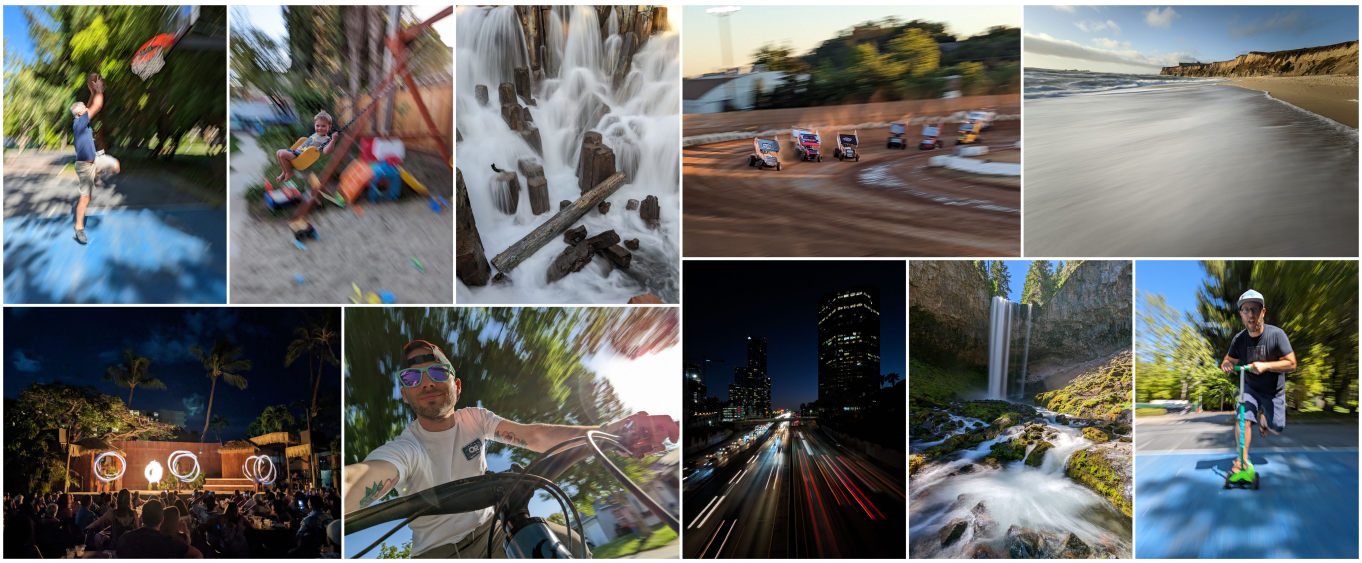


Fig. 10. Several foreground and background blur examples produced by our system. Several more examples can be found in the supplementary material, along with the corresponding regular exposure and a tone-mapped input burst example. Input RAW burst examples are available on our project webpage: https://motion-mode.github.io/.

For category 1, we produce a mask of pixels with very little motion across the entire set of frame pairs, $M_{\texttt{flow}}$:

(1) Compute a per-pixel maximum motion magnitude $|F|$ across all frame pairs.
(2) Compute a reference motion magnitude $|F|_{\texttt{ref}}$ that's effectively a robust max over all pixels in $|F|$ (i.e., 99th percentile).
(3) Rescale and clamp the per-pixel motion magnitudes such that anything below $\alpha|F|_{\texttt{ref}}$ is mapped to 0 and anything above $\beta|F|_{\texttt{ref}}$ is mapped to 1. We use values 0.16 and 0.32 for α and β respectively.

$$M_{\texttt{flow}} = \frac{|F| - \alpha|F|_{\texttt{ref}}}{\beta|F|_{\texttt{ref}} - \alpha|F|_{\texttt{ref}}}$$

(4) Apply a bilateral blur using the sharp image as a guide [He et al. 2013], to ensure that any edges in $M_{\texttt{flow}}$ correspond to real edges and minimize artifacts where the flow field is unreliable (e.g., uniform or texture-less regions like skies).

Category 2 is more complicated and breaks from the physical behavior of optical motion blur in favor of aesthetics. E.g., if a scene has two subjects moving with different trajectories, it would be impossible to sharply align on both simultaneously. Even a single subject can be impossible to align due to movement within the subject, e.g., changes in facial expression, etc. An image with a blurry subject face is a (bad) blurry image. Our solution is to reuse the semantic face signal described in 4.2, modified to only include the faces that have low average feature movement in the aligned reference frame.

Finally, we combine the flow and clipped face masks with a simple max operator. Figure 9 shows the cumulative effect of the two mask types on the final composite.

5 RESULTS

Figure 10 shows several foreground and background blur typical use cases, captured and processed using our system. The bursts were all captured hand-held and the results were generated fully automatically, without the need to adjust any settings. In both cases, what makes these long exposure photographs successful is the presence and contrast between sharp and blurry elements.

The on-device latency of our system varies according to the number of frames selected for processing. The latency for the main stages (see Figure 2), measured on a Google Pixel 7 device, are as follows. Subject detection, including 8x downsampling and tone-mapping of the base frame: 330ms; motion tracking and alignment, including 8x downsampling and tone-mapping: 55ms per frame; inter-frame motion prediction, including concurrent 2x downsampling and RAW to linear conversion: 77ms per selected frame pair; rendering: 42ms per selected frame pair; final upsampling, compositing and tone-mapping of both image results: 790ms. In the background blur case, a small number of frames are typically selected (e.g. 4), leading to a short total latency (e.g. 1.7s). In the foreground blur case, a higher number of frames are typically selected (e.g. 12) but most of the processing is happening during the extended capture (see Section 4.1) and the latency is therefore largely hidden from the user.

5.1 Track Weights Comparison

In the following ablation, we compare the effect of including face-region upweighting in motion track weight maps for background-blur alignment quality. (Please refer to Section 4.2 for more details).

We find that including both gaze saliency and face detections in the motion track weight map benefits image subjects with complex articulated motion (which can cause the wrong part of the subject to be tracked). A representative example is shown in Figure 11.

(a) Without face semantic masking (b) With face semantic masking

Fig. 11. Up-weighting track weights in faces helps subjects with complex/articulated motion. Larger images are the long exposure results rendered with compositing masking disabled (as in Figure 9d) for clarity of the comparison. To their right are the intermediate outputs: track weights are visualized as added green/red channel values, motion track clusters in the top image, and the highest weight selected cluster in the bottom image. (a) Gaze saliency alone peaks the weights on the cat's center, assigning the highest weight to the tracks cluster on the cat's body - resulting in undesirable alignment. (b) Our pipeline's results: up-weighting the face-region causes its motion cluster to be selected, resulting in the desired long exposure tracking the cat's face.

5.2 Image Alignment Comparison

In Figure 16, we showcase additional examples of image alignment on background-blur scenes, comparing the aesthetics of results when the regularization term E_b from Eq. 2 is excluded (left) and included (right). In the left-hand side column of Figure 16, we observe that optimizing just for the subject's sharpness E_b doesn't account for the background of the scene. Consequently, sudden changes in transform parameters over time are allowed, resulting in different parts of the field of view having motion blur in completely different directions. By adding the temporal regularization term E_b, we get the results on the right-hand side column of Figure 16 with consistent blur trails. The second example showcases the effect of dampening the rotational parameter, avoiding the blur vortex (green insets).

5.3 Motion Prediction Comparison

We compare models described in Section 4.6 with those from [Brooks and Barron 2019] that use uniform weights, labelled "BB19-uni.", and that learn $N = 17$ weights per input image, labelled "BB19".

All the compared models were trained with the same hyper-parameters described in [Brooks and Barron 2019]. To supervise the training, we generate a bracketed dataset of input image triplets from many videos, as described in [Reda et al. 2022], synthesizing the pseudo ground-truth motion-blurred image using a previously trained FILM frame interpolation model. To evaluate our model we used a test set with 2000 examples and report the PSNR and SSIM, i.e. comparing the synthesized motion-blur image to the pseudo ground-truth, in Table 1.

A visual comparison on 512 x 384 image crops is provided in Figure 17 and shows that our model performs visually similarly to "BB19" (e.g. blur smoothness, handling of dis-occlusions), despite the significant simplifications of our implementation to run on mobile devices. It also reveals that both models "Ours-abl." and "BB19-uni." suffer from the same discontinuity artifacts in the middle of blur trails, which are described in Section 4.6.

Table 1. Comparing motion prediction models evaluation performance (PSNR and SSIM) and properties: number of learned parameters in millions (M-Par.), number of floating point operations at the evaluated image input resolution in billions (B-Flop) and receptive field computed based on [Dumoulin and Visin 2016] in pixels (Rec. Field). Our simplified model to run on mobile devices, shows comparable quality performance to models with 1.7 times as many parameters, both quantitatively and qualitatively (shown in Figure 17).

Model	PSNR.	SSIM	M-Par.	B-Flop	Rec. Field
BB19	41.78	0.9862	7.057	107.28	202
BB19-uni.	40.07	0.9803	7.056	106.81	202
Ours-large.	41.32	0.9842	7.301	107.40	202
Ours	40.78	0.9823	4.173	114.67	128
Ours-abl.	40.61	0.9808	4.173	114.67	128

Our model runs in under 80ms on a Google Pixel 7 mobile device's TPU [Gupta 2021], when given an input image pair from our low resolution image pipeline.

5.4 Rendering Comparison

5.4.1 Motion Interpolation. In Figure 12, we compare the effects of piecewise-linear flow interpolation vs cubic spline interpolation. Particularly when camera or object motion from frame to frame is irregular, spline interpolation can impart a much more natural and photorealistic appearance to the motion trails.

Fig. 12. Comparison of interpolation methods described in Section 4.7.1 on a scene with a car traversing a roundabout. Top: An intermediate frame from the input burst. Bottom, left to right: average of several successive input frames, linear flow interpolation, spline flow interpolation. Piecewise-linear motion trails look synthetically generated, revealing the number of input frame pairs used to render the image, whereas curved motion trails look optically generated as in a single continuous long exposure. See supplementary material for more examples.

(a) sRGB (b) Linear (c) Soft gamma

Fig. 13. Colorspace comparison of the blurring operation. (a) sRGB colorspace blur loses most of the motion-blurred highlights intensity. (b) Linear colorspace is physically correct but produces dull blur trails due to clipping occurring in the sensor prior to blurring. (c) Soft gamma colorspace blur, described in Section 4.7.3, is able to preserve strong motion-blurred highlights and increases blur trails contrast and color saturation. See supplementary material for more examples.

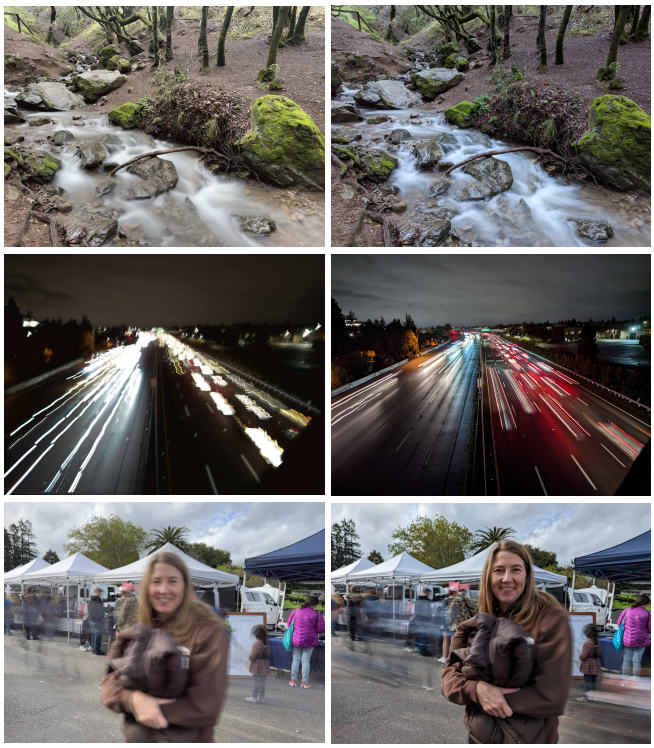
5.4.2 Rendering Colorspace. In Figure 13, we compare the results of performing the blurring operation in a conventional sRGB colorspace, versus a linear physically correct colorspace, versus our non-physical "soft gamma" colorspace, obtained by adjusting the linear-space image in a direction opposite from a usual linear to sRGB color transformation. The figure illustrates how blurring in the soft-gamma colorspace emphasizes and preserves the brightness of the motion trails in the highlights and generally increases their contrast and color saturation.

5.5 Comparison to Mobile Phone Camera Applications

Unlike other works which realize a long exposure effect [Lancelle et al. 2019; Lee et al. 2009; Luo et al. 2018, 2020; Mikamo et al. 2021; Teramoto et al. 2010], our pipeline is a responsive mobile phone capture experience. Therefore, we also compare our results to released capture experiences for consumer phones.

Several mobile applications allow a more manual control of the capture schedule on mobile phones such as Even Longer, Moment, Neoshot, and Procam 8 (all available in the iOS App Store). These apps do not seem to have frame alignment capability, and therefore require a tripod for capturing sharp long exposure images. Spectre, released on iOS, seems to have stabilization and auto-exposure capabilities. Through a capture study of dozens of scenes, we found the hand-held performance of Spectre to be inconsistent. Figure 14 shows representative comparisons of the results of our pipeline with Spectre.

To our knowledge, our pipeline is the only mobile-phone capture experience with all of the following features: background-blur alignment (automatically tracking and keeping a moving subject sharp), robust foreground-blur alignment (keeping the background sharp), motion interpolation for smooth motion trails (compared to results showing temporal undersampling), and face-region sharpness protection (keeping slightly moving subjects sharp).



(a) Spectre (b) Ours

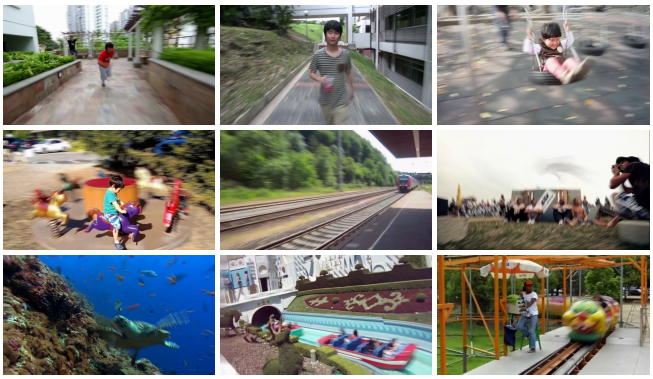
Fig. 14. Comparison of the app Spectre (a) [Spectre [n.d.]] vs. our method (b), on scenes captured hand-held. The light trail scene in the middle row was captured in very windy conditions. Our pipeline shows better background alignment, subject preservation, and more noticeable motion trails. A more extensive comparison with a few additional apps can be found in the supplement.

5.6 Evaluation on Existing Datasets

In Figure 15, we evaluate our method on the publicly available video dataset in [Liu et al. 2014]. The images are also available in the supplement, and can be compared to their results (see their Figure 9), as well as results in [Lancelle et al. 2019] (see their Figure 15).

Our automatic subject detection aligns the result on people's faces when they are detected, and on visually salient features otherwise, which matches the selected subject from previous work in many scenes. When multiple faces are detected, our method chooses to align on the largest face, which may lead to a different outcome (Figure 15a middle-right and lower-right examples). We also observe a possible mismatch when no faces are present or are too small to be detected, e.g. while our saliency signal reacts to the most colorful or brightest areas of the image (Figure 15a lower-left and lower-middle examples respectively).

Even though we use a simple 2D image alignment approach (see Section 4.4.2), our method leads to comparable subject stabilization in most cases. Our similarity transform solver is able to preserve subject sharpness and models a relative virtual camera motion that is similar to that of compared works and is sometimes more accurate (Figure 15a center and Figure 15b right examples).



(a) Our pipeline, without using the segmentation mask data.

(b) Our pipeline, using a manual saliency override mask.

Fig. 15. Several examples showing our method evaluated on the input video dataset in [Liu et al. 2014]. We include a few examples in (a) middle-right and lower row, where our subject detection leads to a different outcome vs. their 3D aware segmentation-driven approach. We add corresponding comparable results in (b), obtained by manually overriding our saliency signal with a subject mask.

Our rendering approach is most similar to [Lancelle et al. 2019] but differs in the implementation to interpolate motion between frames. Our method scales to very high resolution images in an efficient manner on a mobile device, and we find the resulting motion-blur quality to be comparable. Both works benefit from integrating motion-blur from the input images spanning the whole time-interval, unlike the approach in [Liu et al. 2014], which uses a spatially-varying blur of only the base frame. Our method correctly renders the dynamic motion of the scene handling dis-occlusions, showing other moving scene objects, people's moving limbs, and the motion of the background, all relative to the subject and as seen through the virtual camera aligned over time. In contrast, blurring only the base frame assumes the scene is static and only the aligned camera transformation affects the blur. This is most noticeable when comparing our turtle result in Figure 15b-left to theirs. Our system renders the relative coral motion direction correctly, as can be seen in the input video, and shows the turtle's moving fin and the swirling individual motion of surrounding fish.

The amount of blur is normalized by our frame selection algorithm described in Section 4.5. Our stylistic background blur length target is shorter than the results in [Lancelle et al. 2019], and is motivated by the goal to preserve subject sharpness and scene context.

6 LIMITATIONS AND FUTURE WORK

Background blur scenes with very small subjects tend to significantly increase the occurrence of saliency and portrait mask mispredictions and feature tracking errors, ultimately resulting in an undesirable alignment and preserving the sharpness of the incorrect image region. Although our system can handle reasonably small subjects, this problem can be improved further by refining these predictions using the appropriate sub-region of the input images down-sampled at a higher resolution.

Our motion prediction model with receptive field window of 128 pixels can handle approximately 64 pixels of motion disparity at the chosen input low resolution. In our system, this corresponds to 512 pixels of disparity at full resolution, which is a good practical upper bound when capturing 12 megapixel bursts at 30fps. Larger motion disparities across frame pairs cause inaccurate predicted kernel maps and result in significant artifacts in the synthesized motion-blur. When these rare cases are detected in the motion analysis stage of our pipeline, we decide to output only the sharp exposure.

None of the models we tested perfectly handle motion silhouettes, when the foreground and background are moving between perpendicular and opposite directions or in the presence of large motion, causing swirly looking or disocclusion artifacts. We also notice similar artifacts in areas where a cast shadow moves in a different direction than the object where the shadow is projected onto. Some examples can be found in the supplementary material in Figure 1. Resolving these challenging cases is left for future work.

7 CONCLUSION

In this paper, we described a long exposure computational photography system, that is able to produce high quality long exposure foreground or background blur effects. Our system operates in a smartphone camera app and outputs both long and conventional exposures fully automatically, in just a few seconds after pressing the shutter button. We described the key elements that make our system successful: adapting the burst capture schedule to scene motion velocity, separating the main subject from the background and tracking their motion, creating custom aesthetic alignments of input burst images, synthesizing smooth curved motion-blur spanning multiple underexposed HDR sharp input images, and compositing sharp and motion-blurred results to protect important scene locations - exceeding what would be physically possible with traditional long-exposure photography. The careful combination of these elements gives our system the ability to produce aesthetically pleasing blur trails and to preserve sharpness where it is most needed. The end-to-end integration of our system into a consumer mobile device camera makes it possible for casual users to access a creative style previously reserved to more advanced photographers.

ACKNOWLEDGMENTS

"Motion Mode" and the work described in this paper is the result of many individuals' effort in Google Research's Creative Camera team and the Google Camera App team, with help from many collaborating partners in the Gcam, reModel, Pixel and Silicon teams. We are especially thankful to Janne Kontkanen for leading the research on FILM, to Gabriel Nava, Shamvi Punja, Qiurui He, Alex Schiffhauer, Steven Hickson, Po-Ya Hsu, Zhijie Deng, Yen-Hsiang Huang, Kiran Murthy and Sam Hasinoff for their contributions, and to Tim Brooks and Jon Barron for early research. Credit for photography and image quality evaluations goes to our staff photographers Michael Milne, Andy Radin, Nicholas Wilson, Brandon Ruffin, as well as Chris Haney, Sam Robinson, Christopher Farro and Cort Muller. We are also very grateful to Orly Liba, Cassidy Curtis, Kfir Aberman and David Salesin for their help reviewing early drafts of this paper.

REFERENCES

Kfir Aberman, Ju He, Yossi Gandelsman, Inbar Mosseri, David E. Jacobs, Kai Kohlhoff, Yael Pritch, and Michael Rubinstein. 2022. Deep Saliency Prior for Reducing Visual Distraction. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, Los Alamitos, CA, USA, 19819–19828. https://doi.org/10.1109/CVPR52688.2022.01923

Sameer Agarwal, Keir Mierle, and The Ceres Solver Team. 2022. *Ceres Solver*. Google, Inc. https://github.com/ceres-solver/ceres-solver

Valentin Bazarevsky, Yury Kartynnik, Andrey Vakunov, Karthik Raveendran, and Matthias Grundmann. 2019. BlazeFace: Sub-millisecond Neural Face Detection on Mobile GPUs. *ArXiv* abs/1907.05047 (2019).

Tim Brooks and Jonathan T Barron. 2019. Learning to Synthesize Motion Blur. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, Los Alamitos, CA, USA, 6833–6841. https://doi.org/10.1109/CVPR.2019.00700

Gabriel J. Brostow and Irfan Essa. 2001. Image-Based Motion Blur for Stop Motion Animation. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '01)*. Association for Computing Machinery, New York, NY, USA, 561–566. https://doi.org/10.1145/383259.383325

Vincent Dumoulin and Francesco Visin. 2016. A guide to convolution arithmetic for deep learning. https://doi.org/10.48550/ARXIV.1603.07285

David S Ebert, F Kenton Musgrave, Darwyn Peachey, Ken Perlin, and Steven Worley. 2003. *Texturing & modeling: a procedural approach*. Morgan Kaufmann.

Matthias Grundmann, Vivek Kwatra, and Irfan Essa. 2011. Auto-directed video stabilization with robust L1 optimal camera paths. In *2011 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, Los Alamitos, CA, USA, 225–232. https://doi.org/10.1109/CVPR.2011.5995525

Monika Gupta. 2021. Google Tensor is a milestone for machine learning. https://blog.google/products/pixel/introducing-google-tensor

R. I. Hartley and A. Zisserman. 2004. *Multiple View Geometry in Computer Vision* (second ed.). Cambridge University Press, ISBN: 0521540518.

Samuel W. Hasinoff, Dillon Sharlet, Ryan Geiss, Andrew Adams, Jonathan T. Barron, Florian Kainz, Jiawen Chen, and Marc Levoy. 2016. Burst Photography for High Dynamic Range and Low-Light Imaging on Mobile Cameras. *ACM Trans. Graph.* 35, 6, Article 192 (dec 2016), 12 pages. https://doi.org/10.1145/2980179.2980254

Kaiming He, Jian Sun, and Xiaoou Tang. 2013. Guided Image Filtering. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35, 6 (2013), 1397–1409. https://doi.org/10.1109/TPAMI.2012.213

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In *2015 IEEE International Conference on Computer Vision (ICCV)*. IEEE Computer Society, Los Alamitos, CA, USA, 1026–1034. https://doi.org/10.1109/ICCV.2015.123

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, Los Alamitos, CA, USA, 770–778. https://doi.org/10.1109/CVPR.2016.90

Peter J. Huber. 1964. Robust Estimation of a Location Parameter. *The Annals of Mathematical Statistics* 35, 1 (1964), 73 – 101. https://doi.org/10.1214/aoms/1177703732

Ming Jiang, Shengsheng Huang, Juanyong Duan, and Qi Zhao. 2015. SALICON: Saliency in Context. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, Los Alamitos, CA, USA, 1072–1080. https://doi.org/10.1109/CVPR.2015.7298710

Alexandre Karpenko, David Jacobs, Jongmin Baek, and Marc Levoy. 2011. *Digital video stabilization and rolling shutter correction using gyroscopes*. Technical Report. Stanford University CS. 13 pages.

M. Lancelle, P. Dogan, and M. Gross. 2019. Controlling Motion Blur in Synthetic Long Time Exposures. *Computer Graphics Forum* 38, 2 (2019), 393–403. https://doi.org/10.1111/cgf.13646

H. Lee, C. H. Lee, and K. Yoon. 2009. Motion based Painterly Rendering. *Computer Graphics Forum* 28, 4 (2009), 1207–1215. https://doi.org/10.1111/j.1467-8659.2009.01498.x

Jaakko Lehtinen, Timo Aila, Jiawen Chen, Samuli Laine, and Frédo Durand. 2011. Temporal Light Field Reconstruction for Rendering Distribution Effects. *ACM Trans. Graph.* 30, 4, Article 55 (jul 2011), 12 pages. https://doi.org/10.1145/2010324.1964950

Orly Liba, Kiran Murthy, Yun-Ta Tsai, Tim Brooks, Tianfan Xue, Nikhil Karnad, Qiurui He, Jonathan T. Barron, Dillon Sharlet, Ryan Geiss, Samuel W. Hasinoff, Yael Pritch, and Marc Levoy. 2019. Handheld Mobile Photography in Very Low Light. *ACM Trans. Graph.* 38, 6, Article 164 (nov 2019), 16 pages. https://doi.org/10.1145/3355089.3356508

Feng Liu, Michael Gleicher, Hailin Jin, and Aseem Agarwala. 2009. Content-Preserving Warps for 3D Video Stabilization. *ACM Trans. Graph.* 28, 3, Article 44 (jul 2009), 9 pages. https://doi.org/10.1145/1531326.1531350

Feng Liu, Michael Gleicher, Jue Wang, Hailin Jin, and Aseem Agarwala. 2011. Subspace Video Stabilization. *ACM Trans. Graph.* 30, 1, Article 4 (feb 2011), 10 pages. https://doi.org/10.1145/1899404.1899408

Shuaicheng Liu, Jue Wang, Sunghyun Cho, and Ping Tan. 2014. TrackCam: 3D-Aware Tracking Shots from Consumer Video. *ACM Trans. Graph.* 33, 6, Article 198 (Nov 2014), 11 pages. https://doi.org/10.1145/2661229.2661272

Shuaicheng Liu, Lu Yuan, Ping Tan, and Jian Sun. 2013. Bundled Camera Paths for Video Stabilization. *ACM Trans. Graph.* 32, 4, Article 78 (jul 2013), 10 pages. https://doi.org/10.1145/2461912.2461995

Xuejiao Luo, Nestor Z. Salamon, and Elmar Eisemann. 2018. Adding Motion Blur to Still Images. In *Proceedings of the 44th Graphics Interface Conference (GI '18)*. Canadian Human-Computer Communications Society, Waterloo, CAN, 108–114. https://doi.org/10.20380/GI2018.15

Xuejiao Luo, Nestor Z. Salamon, and Elmar Eisemann. 2020. Controllable Motion-Blur Effects in Still Images. *IEEE Transactions on Visualization and Computer Graphics* 26, 7 (2020), 2362–2372. https://doi.org/10.1109/TVCG.2018.2889485

Michihiro Mikamo, Ryo Furukawa, and Hiroshi Kawasaki. 2021. A Method for Adding Motion-Blur on Arbitrary Objects by Using Auto-Segmentation and Color Compensation Techniques. In *2021 IEEE International Conference on Image Processing (ICIP)*. 1854–1858. https://doi.org/10.1109/ICIP42928.2021.9506443

C. Morimoto and R. Chellappa. 1998. Evaluation of image stabilization algorithms. In *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP '98 (Cat. No.98CH36181)*, Vol. 5. 2789–2792 vol.5. https://doi.org/10.1109/ICASSP.1998.678102

Fernando Navarro, Francisco J. Serón, and Diego Gutierrez. 2011. Motion Blur Rendering: State of the Art. *Computer Graphics Forum* 30, 1 (2011), 3–26. https://doi.org/10.1111/j.1467-8659.2010.01840.x

Augustus Odena, Vincent Dumoulin, and Chris Olah. 2016. Deconvolution and Checkerboard Artifacts. *Distill* (2016). https://doi.org/10.23915/distill.00003

Fatih Porikli. 2004. Learning object trajectory patterns by spectral clustering. In *2004 IEEE International Conference on Multimedia and Expo (ICME)*, Vol. 2. IEEE Computer Society, Los Alamitos, USA, 1171–1174. https://doi.org/10.1109/ICME.2004.1394427

Fitsum Reda, Janne Kontkanen, Eric Tabellion, Deqing Sun, Caroline Pantofaru, and Brian Curless. 2022. FILM: Frame Interpolation For Large Motion. In *Computer Vision – ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part VII*. Springer-Verlag, Berlin, Heidelberg, 250–266. https://doi.org/10.1007/978-3-031-20071-7_15

Mads J.L. Rønnow, Ulf Assarsson, and Marco Fratarcangeli. 2021. Fast analytical motion blur with transparency. *Computers & Graphics* 95 (2021), 36–46. https://doi.org/10.1016/j.cag.2021.01.006

Nestor Z. Salamon, Markus Billeter, and Elmar Eisemann. 2019. ShutterApp: Spatio-temporal Exposure Control for Videos. *Computer Graphics Forum* 38, 7 (2019), 675–683. https://doi.org/10.1111/cgf.13870

Christophe Schlick. 1994. *Fast Alternatives to Perlin's Bias and Gain Functions*. Academic Press Professional, Inc., USA, 401–403.

Fuhao Shi, Sung-Fang Tsai, Youyou Wang, and Chia-Kai Liang. 2019. Steadiface: Real-Time Face-Centric Stabilization On Mobile Phones. In *2019 IEEE International Conference on Image Processing (ICIP)*. 4599–4603. https://doi.org/10.1109/ICIP.2019.8803679

Spectre. [n.d.]. Spectre app. https://spectre.cam. Accessed: 2023-01-17.

Michael Stengel, Pablo Bauszat, Martin Eisemann, Elmar Eisemann, and Marcus Magnor. 2015. Temporal Video Filtering and Exposure Control for Perceptual Motion Blur. *IEEE Transactions on Visualization and Computer Graphics* 21, 5 (2015), 663–671. https://doi.org/10.1109/TVCG.2014.2377753

Okihide Teramoto, In Kyu Park, and Takeo Igarashi. 2010. Interactive Motion Photography from a Single Image. *Vis. Comput.* 26, 11 (nov 2010), 1339–1348. https://doi.org/10.1007/s00371-009-0405-6

Neal Wadhwa, Rahul Garg, David E. Jacobs, Bryan E. Feldman, Nori Kanazawa, Robert Carroll, Yair Movshovitz-Attias, Jonathan T. Barron, Yael Pritch, and Marc Levoy. 2018. Synthetic Depth-of-Field with a Single-Camera Mobile Phone. *ACM Trans. Graph.* 37, 4, Article 64 (jul 2018), 13 pages. https://doi.org/10.1145/3197517.3201329

Ben Weiss. 2006. Fast Median and Bilateral Filtering. *ACM Trans. Graph.* 25, 3 (jul 2006), 519–526. https://doi.org/10.1145/1141911.1141918

Bartlomiej Wronski, Ignacio Garcia-Dorado, Manfred Ernst, Damien Kelly, Michael Krainin, Chia-Kai Liang, Marc Levoy, and Peyman Milanfar. 2019. Handheld Multi-Frame Super-Resolution. *ACM Trans. Graph.* 38, 4, Article 28 (jul 2019), 18 pages. https://doi.org/10.1145/3306346.3323024

Julio Zaragoza, Tat-Jun Chin, Michael S. Brown, and David Suter. 2013. As-Projective-As-Possible Image Stitching with Moving DLT. In *2013 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, Los Alamitos, CA, USA, 2339–2346. https://doi.org/10.1109/CVPR.2013.303

Congxuan Zhang, Ling ling Zhu, Zhen Chen, Ding ding Kong, and Xuan Shang. 2016. An Improved Evaluation Method for Optical Flow of Endpoint Error. In *Proceedings of the International Conference on Computer Networks and Communication Technology (CNCT 2016)*. Atlantis Press, 312–317. https://doi.org/10.2991/cnct-16.2017.44

Timo Zinßer, Jochen Schmidt, and Heinrich Niemann. 2005. Point set registration with integrated scale estimation. In *International conference on pattern recognition and image processing*. 116–119.

Fig. 16. Background blur aesthetics visualized across different parts of the field of view (red and blue insets) comparing results without (left), and with (right) the regularization term E_b from Eq. 2. By additionally imposing the rotational constraints from Section 4.4, we remove an undesirable sharp region surrounded by a blur vortex (green inset). Insets are displayed at 5x magnification.

(a) Input average (b) FILM (ground truth) (c) BB19-uni. (d) BB19 (e) Ours (f) Ours-abl.

Fig. 17. Model comparisons from single input image pairs shown overlaid in column (a). Column (b) is rendered using recursive frame interpolation with the FILM model from [Reda et al. 2022], and is used as ground truth when training our models. Columns (c) and (d) are rendered with the model from [Brooks and Barron 2019], with uniform and learned weights respectively. Columns (e) and (f) are rendered with our model, with and without the ramp function w_n respectively. Differences are subtle, showing that our mobile model simplifications do not affect image quality substantially. The examples on each row showcase blur quality and disocclusions with various amount of motion disparity. The last row contains disocclusions with opposite motion. Several additional challenging examples can be found in the supplementary material, we encourage the reader to compare the accompanying images, to see differences more clearly.