A. Weights for the base fusion

Since the base component only contains low frequencies and cannot be super-resolved, we propose a simple pipeline consisting of i) alignment of the LR base components $B_i$ to the reference, ii) temporal fusion via weighted average to attenuate noise, iii) upscaling using bilinear interpolation. For the temporal fusion the weights in the weighted average are simply the exposure times:

$$B^{LR}(x) = \frac{\sum_i e_i \text{Warp}(B_i^{LR}(x))}{\sum_i e_i}$$  (S1)

In this section we will provide a justification for this choice, which is based on two approximations.

Approximate noise model for the base. The base results from the convolution with a Gaussian kernel $G$. At pixel $x$ we have

$$B_i^{LR}(x) = \sum_h G(h) I_i^{LR}(x + h).$$

Assuming the signal-dependent Gaussian noise model of Eq. (2), we have that $B_i^{LR}(x)$ also follows a Gaussian distribution with the following mean and variance:

$$\mathbb{E}\{B_i^{LR}(x)\} = \sum_h G(h) I_i^{LR}(x + h)$$

$$\mathbb{V}\{B_i^{LR}(x)\} = \frac{\alpha}{e_i^2} \sum_h G^2(h) I_i^{LR}(x + h) + \frac{\beta}{e_i^2} \sum_h G^2(h).$$

We are going to assume that the clean LR image $I_i^{LR}$ varies smoothly in the filter support, and thus

$$\mathbb{E}\{B_i^{LR}(x)\} \approx I_i^{LR}(x), \quad \mathbb{V}\{B_i^{LR}(x)\} \approx \frac{\alpha e_i I_i^{LR}(x) + \beta}{e_i^2}.$$  (S2)

where $\alpha = a \sum_h G^2(h)$ and $\beta = b \sum_h G^2(h)$. This rough approximation allows us to use a signal-dependent Gaussian noise model like (2). The approximation is only valid in regions where the image is smooth (away from edges, textures, etc.). However, these are the regions in which we are mainly interested, since it is where the low frequency noise present in the base becomes more noticeable.

Approximate MLE estimator for the weights. After alignment, for a given pixel $x$ we have different values acquired with varying exposure times, which we are going to denote as $z_i = \text{Warp}(B_i^{LR}(x))$ to simplify notation. We also have the corresponding clean LR base images $B_i^{LR}$, and we are going to assume that they coincide after alignment, i.e. $y = \text{Warp}(B_i^{LR}(x))$ for $i = 1, ..., m$. We would like to estimate $y$ from the series of observations

$$z_i \sim \mathcal{N}(\sigma^2_i(y)), \quad \sigma^2_i(y) = \frac{\alpha e_i y + \beta}{e_i^2}.$$

This problem occurs in HDR imaging, when estimating the unknown irradiance given noisy acquisitions with varying exposure times [1, 6]. Each $z_i$ is an unbiased estimator of $y$. Therefore, if the variances were known, we can minimize the MSE with the following weighted average, where the weights are the inverse of the variances:

$$\hat{y} = \frac{\sum_i w_i z_i}{\sum_i w_i}, \quad w_i = \frac{e_i^2}{\alpha e_i y + \beta}.$$  (S3)

The problem is that the weights depend on the unknown $y$. In [6] Granados et al. solve this problem with an iterative weighted average:

$$w_i^0 = \frac{e_i^2}{\alpha e_i z_i + \beta},$$

$$w_i^k = \frac{e_i^2}{\alpha e_i y^k + \beta}, \quad y^{k+1} = \frac{\sum_i w_i^k z_i}{\sum_i w_i^k}, \quad k = 1, 2, ...$$

It can be shown that this converges to the maximum likelihood estimate.
In our case, we are going to simplify expression (S3) by assuming that $\alpha e_i y \gg \beta$, and therefore $w_i \approx \frac{e_i}{\alpha y}$. Under this assumption, we obtain

$$\hat{y} = \frac{\sum e_i z_i}{\sum e_i}.$$  \hspace{1cm} (S4)

This assumption holds for brighter pixels and well exposed images [1].

### B. HDR-DSP architecture

Our HDR-DSP architecture has 3 trainable modules: Motion estimator, Encoder, and Decoder. The Feature Shift-and-Pool block does not have any trainable parameters. Our motion estimator follows the work of [17].

<table>
<thead>
<tr>
<th>Modules</th>
<th>Layers</th>
<th>Nb parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Motion Estimator</td>
<td>FNet</td>
<td>1744354</td>
</tr>
<tr>
<td>Encoder</td>
<td>Conv2d</td>
<td>332992</td>
</tr>
<tr>
<td></td>
<td>ResBlock(64)</td>
<td>776065</td>
</tr>
<tr>
<td>FS&amp;P</td>
<td>Conv2d</td>
<td>0</td>
</tr>
<tr>
<td>Decoder</td>
<td>Conv2d</td>
<td>776065</td>
</tr>
<tr>
<td></td>
<td>ResBlock(64)</td>
<td>776065</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2853411</td>
</tr>
</tbody>
</table>

| Table S2. HDR-DSP Architecture |

We train HDR-DSP in two stages, first pretraining the motion estimator and then the end-to-end system.

**Phase 1: Pre-train the Motion Estimator.** Training the motion estimator in the case of images obtained with different exposures is a challenging task. We first pretrain it on the simulated dataset to ensure that it produces accurate flows. We monitor the quality of the estimations by comparing with the ground truth flows, until reaching an averaged error of 0.05 pixel.

For training the motion estimator our first choice was to use the $L_1$ distance between the reference image and the radiometrically corrected warped image. However, the quality of the estimated flows were not acceptable (with errors above 0.1 pixel). Indeed, since motion estimation relies on the photometric consistency between frames, it is very sensitive to the intensity fluctuations between frames (as it is the case for our normalized LR frames $I_{LR_i}$), which results in imprecise alignments.

To prevent this issue we compute the warping loss on the details rather than on the images, which is common in traditional optical flow [11, 17]. The loss is computed for each flow $F_{i \rightarrow r}$ estimated by the MotionEst module

$$\ell_{mc}(\{F_{i \rightarrow r}\}_{i=1}^m) = \sum_i \|\text{Detail}(I_{LR_i}^{LR}) - \text{Detail}(\text{Pullback}(I_{LR_i}^{LR}, F_{i \rightarrow r}))\|_1 + \lambda_1 TV(F_{i \rightarrow r}), \hspace{1cm} (S5)$$

where Pullback computes a bicubic warping of $I_{LR_i}^{LR}$ according to a flow. Detail applies a high-pass filter to the images, TV is the finite difference discretization classic Total Variation regularizer [16], and $\lambda_1 = 0.003$ is a hyperparameter controlling the regularization strength.

We set the batch size to 32 and use Adam [9] with the default Pytorch parameters and a initialized learning rate of $10^{-4}$ to optimize the loss. The pre-training converges after 50k iterations and takes about 3 hours on one NVIDIA V100 GPU.

**Phase 2: Train the whole system end-to-end.** We then use the pretrained motion estimator and train the entire system end-to-end using the total loss:

$$\text{loss} = \ell_{self} + \lambda_2 \ell_{mc}. \hspace{1cm} (S6)$$

We set $\lambda_2 = 3$ in our experiments. Furthermore, to avoid boundary issues, the loss does not consider values at a distance below 2 pixels from the border of the frames.

We train our model on LR crops of size $64 \times 64$ pixels and validate on LR images of size $256 \times 256$ pixels. During training, our network is fed with a random number of LR input images (from 4 to 14) in each sequence. We set the batch size to 16 and optimize the loss using the Adam optimizer with default parameters. The learning rates are initialized to $10^{-3}$, then scaled by 0.3 each 400 epochs. The training takes 20h (1200 epochs) on one NVIDIA V100 GPU.
D. Trainable feature pooling alternative

The feature pooling block FSP described in Section 4.1 of the main article does not have any trainable parameters. In this section we investigate the use of a trainable layer, named PoolNet, for performing this task.

We considered a simple trainable network PoolNet that performs feature pooling (Table S3) instead of statistical feature poolings (Avg-Max-Std) as presented in the paper. To this aim, PoolNet takes as input the concatenation of N features J\textsuperscript{HR\_i} and N weights W\textsuperscript{HR\_i} (computed by the SPMC [19] module from N LR images) and produces the fused HR features. Then, the Decoder network reconstructs the HR detail image from the fused features.

A drawback of PoolNet is that it can only be applied on a fixed number of frames. Table S4 compares the performance of the PoolNet (which replaces the Avg-Max-Std feature pooling) trained on 4 and 14 frames with our original method. We can see that in the case of small number of frames, PoolNet attains a performance comparable to our HDR-DSP method using the Avg-Max-Std feature pooling. However, in the case of 14 frames, there is a big gap of 0.3 dB between our method and PoolNet. It seems that it is more difficult for PoolNet to capture the necessary statistics from many features.

E. Alternative exposure weighting strategies

As discussed in the main paper, the LRs with longer exposure time should contribute more to the reconstruction because of their high signal-to-noise ratio. In our proposed method, we use the un-normalized LR images as additional input to the Encoder so as that the Encoder perceives the noise level in each LR image. Subsequently, the Encoder can decide which features are more important.

We also evaluated an alternative strategy to weight the features (WF) based on the exposure times. This simply consists in weighting the features J\textsuperscript{LR\_i} by the corresponding exposure time in the SPMC module. Actually, this was inspired from the ME S&A method.

This strategy leads to slightly worse yet adequate feature encodings (-0.08dB) as shown in Table S5. Moreover, using both feature weighting and LRs encoding (third column) leads to the same performance as only using LRs encoding. This implies that the Encoder already encodes the necessary information about the signal-dependent noise on the features.

F. Adaptation of existing methods to multi-exposure sequences

We detail here the adaptations to the algorithms we used in the comparisons.

ME S&A. Multi-exposure Shift-and-add is a weighted version of the classic shift-and-add method [5, 7, 8, 12] designed for multi-exposure sequences. Usually, S&A produces the HR image by registering the LR images onto the HR grid using the corresponding optical flows. After the registration step, the intensities of the LR images are splatted to the neighborhood integer-coordinate pixels using some kernel interpolation. Finally, pixel-wised aggregation is done to obtain the HR output image. Therefore a naive method consists of using the classic S&A method on the normalized LR images. However this ignores the different signal-to-noise ratios in the normalized images and fails to greatly reduce the noise. Using the same arguments as in the Sec. A, we propose the weighted S&A for multi-exposure sequence

$$\tilde{I}^{HR} = \frac{\sum_{i=1}^{m} \text{Register}(\tilde{I}^{LR\_i})}{\sum_{i=1}^{m} e_i}$$

(S7)

where Register maps and splats the un-normalized images $\tilde{I}^{LR\_i}$ onto the HR grid.

Base-detail ACT (BD ACT). ACT [2] is a traditional multi-image super-resolution method developed for Planet SkySat single-exposure sequences. It formulates the reconstruction as an inverse problem and solves it by an iterative optimization method. BD ACT extends ACT to support multi-exposure images by adopting the same base-detail strategy as proposed in HDR-DSP: the details of the images are fused by ACT, and the base is reconstructed by the upsampled average of the bases of the input images.

HighRes-net (HR-net) and RAMS. HighRes-net [4] and RAMS [18] are two super-resolution methods for multi-temporal PROBA-V satellite images. However in the PROBA-V dataset, the identity of the LR reference image is unavailable. This hinders the true potential of the methods trained on this dataset. As a result we use the reference-aware super-resolution [14] of HighRes-net and RAMS. In HighRes-net, the reference image is used as a shared representation for all LR images. Each LR image is embedded jointly with this reference before being recursively fused. In RAMS, each LR image is aligned to the reference image before being input to the residual attention block. The registration step of RAMS is done with inverse compositional algorithm [3], which is robust to noise and brightness change. As HighRes-net and RAMS are supervised methods, we also use a radiometric correction on the output before computing the loss [4].

Table S3. PoolNet (N) architecture for trainable feature pooling.

<table>
<thead>
<tr>
<th>Input</th>
<th>( N ) Features (64 channels) + ( N ) Weights (1 channel)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( \text{Conv2d}(\text{in}=N(64+1), \text{out}=256, k=1, s=1, p=0) )</td>
</tr>
<tr>
<td></td>
<td>( \text{ReLU} )</td>
</tr>
<tr>
<td></td>
<td>( \text{Conv2d}(\text{in}=256, \text{out}=128, k=1, s=1, p=0) )</td>
</tr>
<tr>
<td></td>
<td>( \text{ReLU} )</td>
</tr>
<tr>
<td></td>
<td>( \text{Conv2d}(\text{in}=128, \text{out}=64, k=1, s=1, p=0) )</td>
</tr>
<tr>
<td></td>
<td>( \text{ReLU} )</td>
</tr>
<tr>
<td>Output</td>
<td>fused HR feature</td>
</tr>
</tbody>
</table>

Table S4. Trainable feature pooling evaluation. Since the trainable feature pooling networks PoolNet only accepts a fixed number of frames (in this case 4 and 14) we compare it with HDR-DSP also trained with fixed number of frames.

<table>
<thead>
<tr>
<th>Methods</th>
<th>HDR-DSP</th>
<th>HDR-DSP 4</th>
<th>HDR-DSP 14</th>
<th>PoolNet 4</th>
<th>PoolNet 14</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSNR (dB) 4 frames</td>
<td>52.81</td>
<td>52.69</td>
<td>51.31</td>
<td>52.76</td>
<td>N/A</td>
</tr>
<tr>
<td>PSNR (dB) 14 frames</td>
<td>55.85</td>
<td>54.26</td>
<td>55.53</td>
<td>N/A</td>
<td>55.55</td>
</tr>
<tr>
<td>PSNR (dB) variable n frames</td>
<td>54.70</td>
<td>53.85</td>
<td>54.07</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Table S5. Handling of the signal-dependent noise

<table>
<thead>
<tr>
<th>Methods</th>
<th>HDR-DSP</th>
<th>DSP (+WF - LR)</th>
<th>DSP (+WF)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSNR (dB) ME</td>
<td>54.70</td>
<td>54.62</td>
<td>54.70</td>
</tr>
</tbody>
</table>

using the normalized LR images as input. We also use the loss on the details to train the motion estimator in DSA.

G. Execution time

Table S6 reports the execution time of the methods studied on the synthetic multi-exposure dataset. Due to its convolutional architecture, HighRes-net is the fastest. HDR-DSP is slightly more costly than DSA since it performs feature pooling instead of a simple average and requires fusing the bases together. ME S&A and BD-ACT are both executed on CPU, the later being quite costly due to the linear spline system inversion.

H. Additional comparisons using real SkySat sequences

Figure S1 presents results obtained on real multi-exposure SkySat images using 9 frames. This is a challenging sequence as it contains moving vehicles. Note how the road markings are better seen in the HDR-DSP result. However, since HDR-DSP does not account for moving objects (the motion estimator only predicts smooth motion within a range of 5 pixels) the cars are blurry.

Figure S2 shows another example of reconstruction on a real sequence of 7 SkySat images. Even though there are only 7 images in this sequence and most of them are very noisy, HDR-DSP is able to produce a clean image. The fine details are well restored.

I. Exposure error analysis

We observed a discrepancy between the reported exposure time by Planet and correct normalization ratios. This can be explained by measurement imprecision since the quantities are in sub-milliseconds range, or by local illumination effects such as vignetting. To estimate the correct exposure ratio for a given pair of images, we registered the images using phase correlation, masked saturated pixels and computed the spatial median of the ratio between the two frames. We then validated visually that such exposure ratios were more precise that the reported exposure time (less flicker was observed).

Figure S3 shows the relation between the reported ratio, and the estimated one. We find that errors are usually in the order of a few percent, but also observe larger errors. The nominal exposure times range from 0.4ms to 4.5ms. Note that the absolute error in exposure the time measurement is probably constant regardless of the exposure time. However, when computing the ratio of two exposures with errors this might result in a large divergence of the ratio, especially if the exposure in the denominator is a short one.

Note that for the proposed super-resolution method, we used the imprecise, reported exposure times and not the estimated one, as the estimation method itself can fail.
Table S6. Execution time (s) on 200 sequences of size $15 \times 256 \times 256$ pixels.

<table>
<thead>
<tr>
<th>Methods</th>
<th>RAMS</th>
<th>ME S&amp;A</th>
<th>HR-net</th>
<th>BD-ACT</th>
<th>DSA</th>
<th>HDR-DSP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time (s)</td>
<td>93</td>
<td>276</td>
<td>22</td>
<td>555</td>
<td>82</td>
<td>97</td>
</tr>
</tbody>
</table>

Figure S1. Super-resolution from a real multi-exposure sequence of 9 SkySat images. Top row: Original low resolution images with different exposures. Middle row: Reconstructions from five methods, including ours trained with self-supervision (right). Bottom row: Zoom on a detail of the results.

References


Figure S2. Super-resolution from a real multi-exposure sequence of 7 SkySat images. Top row: Original low resolution images with different exposures. Middle row: Reconstructions from five methods, including ours trained with self-supervision (right). Bottom row: Zoom on a detail of the results.

Figure S3. Normalized estimated exposure ratio with respect to provided exposure time.


