# Denoise-First Motion Correction and Spatial Denoising for Calcium Imaging

Tina Giorgadze, Lindsey Ma

December 15, 2025

**Abstract**

Calcium imaging is a widely used technique for recording population-scale neural activity at high spatial and temporal resolution. However, raw calcium videos suffer from significant noise contamination, motion artifacts, and background fluctuations, all of which complicate downstream demixing and neural signal extraction. Standard pipelines such as Suite2p and PMD-based decompositions typically apply motion correction first, followed by UV or low-rank decomposition and temporal denoising.

This project introduces two methodological modifications: (1) a denoise-first pipeline, where a temporal denoiser is trained on pre–motion-corrected data and used to improve the stability of motion correction and compression; and (2) a spatial denoising variant that replaces temporal regularization with spatial smoothing of component estimates. We evaluate both approaches on multiple calcium imaging datasets and benchmark their performance against the conventional pipeline.

## 1 Background

### 1.1 Calcium Imaging in Neuroscience

Calcium imaging provides an indirect readout of neural activity by tracking fluorescence changes associated with calcium influx. Two-photon and one-photon microscopes capture these signals as time-lapse movies, often tens of thousands of frames long, with fields of view that can span hundreds of thousands of pixels. Although this technique generates stable, high-throughput recordings, the resulting videos are affected by several sources of distortion: shot noise and camera noise introduce frame-to-frame fluctuations; motion from breathing, behavior, or slow drift disrupts spatial alignment; and background fluorescence adds spatially varying contamination.

These issues motivate preprocessing steps that improve signal quality before any downstream demixing or activity inference. To extract meaningful neural signals, standard preprocessing includes motion correction, denoising, dimensionality reduction (UV decomposition or SVD), and finally demixing.

In the classical approach, denoising occurs *after* motion correction, and the denoiser is trained on the motion-corrected temporal components.

# 2 Old Pipeline (Baseline)

The existing pipeline (1) this project expands upon follows the following standard steps:

1. Apply rigid or piecewise rigid motion correction to the raw movie.
2. Compute an initial UV decomposition.
3. Train a blind-spot temporal denoiser on the temporal component $V$.
4. Apply PMD decomposition using the trained denoiser to obtain a compressed representation.

This pipeline performs well when motion correction is stable. However, noisy or low-SNR movies can produce unstable shift trajectories, blurred templates, and degraded UV initialization, which then propagates into the denoiser training stage.

# 3 Methodology

## 3.1 Denoise-First Pipeline

To address the sensitivity of motion correction to noise, we introduce a denoise-first modification:

1. Take the raw, pre-motion-corrected movie and flatten it.
2. Compute an initial UV decomposition to extract temporal traces $V$.
3. Train a temporal denoiser on these raw traces.
4. Perform motion correction on the movie.
5. Run final PMD decomposition using the same pretrained pre-motion correction denoiser.

The motivation behind this change is to see whether we see any improvement by ensuring that the potential poor quality of motion correction that tends to arise from noisy data doesn't negatively influence the denoiser.

To better understand the role and the influence of a temporal denoiser, we use compare approaches: a blind-spot denoiser and a Gaussian smoother.

### 3.2 Temporal Denoiser Module

To denoise temporal activity traces extracted from the UV decomposition, the old pipeline employs a self-supervised blind-spot neural network that predicts each time point using only its temporal neighbors. The key component is a masked one-dimensional convolutional layer, where the center element of the kernel is forced to zero. This prevents the network from accessing the current noisy observation when predicting it, thereby enforcing a self-supervised objective.

The denoiser architecture contains two parallel pathways. A sequence of standard convolutions extracts multi-scale temporal features, while a stack of masked convolutions with increasing dilation enforces the blind-spot constraint and expands the receptive field without permitting direct access to the target entry. The outputs of all blind-spot layers are concatenated and passed through a $1 \times 1$ convolution to produce per-timepoint predictions.

The network predicts both a mean trace and a total variance trace. The mean represents the denoised temporal signal, while the variance quantifies the model's uncertainty and is constrained to be positive via a Softplus activation. Training is performed using a Gaussian negative log-likelihood loss,

$$\mathcal{L} = \log(\sigma_t^2) + \frac{(x_t - \mu_t)^2}{\sigma_t^2},$$

which encourages the model to learn smooth temporal structure while adapting its variance estimate to local noise levels.

Before full inference, a quantile of the predicted variance is used to estimate the per-trace observation noise variance. At test time, the final denoised estimate is computed through a Bayesian shrinkage rule,

$$\hat{x}_t = \frac{\sigma^2_{\text{noise}}}{\sigma^2_{\text{total}}}\mu_t + \frac{\sigma^2_{\text{signal}}}{\sigma^2_{\text{total}}}x_t,$$

which balances the network prediction with the original observation based on the relative contributions of signal and noise. Because the model operates on fixed-length windows, long traces are processed using overlapping segments that are averaged in the overlap region to ensure smoothness.

This denoiser requires no clean labels and learns temporal structure directly from the dataset. When integrated into the PMD pipeline, it improves compression quality, reduces temporal roughness, and enhances downstream motion correction and component extraction.

### 3.3 Gaussian Spatial Denoising

In addition to temporal denoising, we implement a simple spatial denoiser based on Gaussian smoothing of individual frames. Gaussian filtering replaces each pixel with a weighted average of its spatial neighborhood, where weights decay with distance from the center according to a Gaussian kernel. For an image $I(x, y)$, the denoised output is

$$\tilde{I}(x, y) = (G_\sigma * I)(x, y) = \sum_{u,v} G_\sigma(u, v)\, I(x - u,\, y - v),$$

where $G_\sigma$ is a 2D Gaussian kernel with standard deviation $\sigma$. This operation reduces high-frequency pixel noise while preserving underlying cellular structures.

Within our framework, the Gaussian filter is applied frame-by-frame to the raw movie prior to motion correction. Its role is to suppress local noise fluctuations that interfere with alignment and inflate the spatial roughness of PMD components. Unlike the temporal blind-spot model, Gaussian denoising does not require a training dataset: each pixel is replaced using only spatial information in the same frame. This makes it inexpensive to compute and easy to control through the choice of $\sigma$.

In the denoise-first pipeline, spatial denoising serves two purposes. First, it stabilizes motion correction by reducing pixel-level noise that can cause erroneous shift estimates. Second, it improves the quality of spatial components extracted during PMD decomposition by attenuating high-frequency artifacts that otherwise produce noisy or fragmented basis elements. Although Gaussian smoothing may oversmooth very fine structures if applied aggressively, moderate smoothing improves both the effective signal-to-noise ratio and the interpretability of extracted components.

## 3.4 Spatial Denoising Module

### Motivation

While temporal denoising improves the smoothness and interpretability of activity traces, a substantial fraction of noise in calcium imaging data is spatial rather than temporal. Shot noise, background fluorescence, and optical artifacts manifest as high-frequency spatial fluctuations that corrupt the spatial footprints extracted by PMD. These artifacts often appear as speckled or fragmented structure in the spatial components ($U$), complicating downstream demixing and interpretation.

Motivated by this observation, we introduce a **spatial denoiser** that operates directly on PMD spatial components. Rather than enforcing temporal smoothness on $V$, this approach regularizes $U$ by exploiting local spatial correlations while explicitly preventing the denoiser from trivially copying the noisy input. The goal is to obtain sharper, more compact spatial maps that better reflect underlying neuronal morphology, while remaining fully self-supervised and compatible with existing PMD workflows.

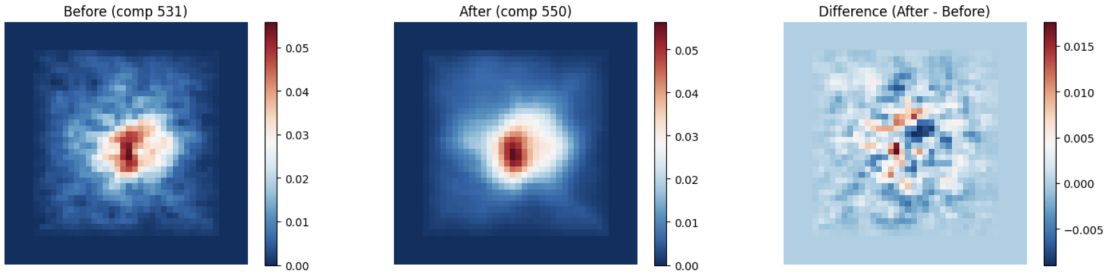



Figure 1: Example noisy versus denoised spatial components (**U**), shown as zoomed-in neuron patches.

### Spatial Denoiser Methodology Overview

The spatial denoiser is implemented as a blind-spot convolutional neural network trained directly on spatial components extracted from an initial PMD decomposition. The full workflow is:

1. Run PMD without any denoiser to obtain initial spatial and temporal factors $(U, V)$.
2. Extract individual spatial components $U_i$ and crop them to their nonzero support.
3. Train a blind-spot spatial denoiser on these cropped components using a self-supervised objective.
4. Apply the trained denoiser to all spatial components.
5. Re-run PMD with the spatial denoiser integrated into the decomposition loop.

This two-pass strategy mirrors the denoise-first temporal pipeline, but shifts regularization into the spatial domain.

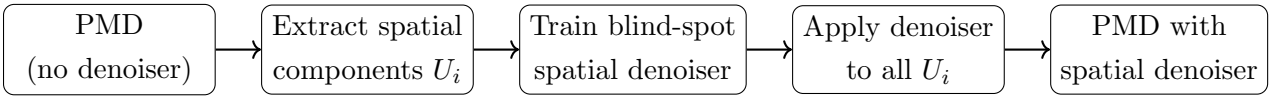



Figure 2: Pipeline diagram showing PMD → spatial denoiser training → PMD with spatial denoiser

### Blind-Spot Spatial Network

To enable self-supervised training, the spatial denoiser adopts a blind-spot architecture: the network is explicitly prevented from accessing the center pixel when predicting its denoised value. This is achieved using masked convolutions, where the central kernel weight is fixed to zero.

The network consists of two parallel paths:

- A **regular convolutional encoder** that progressively builds spatial features.
- A **blind-spot path** composed of masked convolutions with increasing dilation factors.

Each dilation level captures spatial context at a different scale, allowing the model to learn both local and moderately long-range spatial correlations without seeing the target pixel.

The outputs of all blind-spot layers are concatenated and fused using a $1 \times 1$ convolution, producing the final prediction.

**Algorithm 1** Blind-Spot CNN with Dilated Masked Convolutions Architecture

**Require:** Input image $x$

$e_1, \ldots, e_5 \leftarrow$ regular convolutions on $x$

$b_1 \leftarrow \text{MaskedConv}(x, d{=}1)$

$b_{k+1} \leftarrow \text{MaskedConv}(e_k, d{=}k{+}1),\ k = 1, \ldots, 5$

$f \leftarrow \text{Concat}(b_1, \ldots, b_6)$

$\mu \leftarrow \text{Conv}_{1\times1}(f)$

$\sigma^2 \leftarrow \text{Softplus}(\text{Conv}_{1\times1}(f))$

**return** $(\mu, \sigma^2)$

### Probabilistic Output and Training Objective

Following the temporal denoiser design, the spatial network predicts two quantities for each pixel:

- A mean estimate $\mu(x, y)$, representing the denoised spatial signal.
- A total variance estimate $\sigma^2(x, y)$, constrained to be positive via a Softplus activation.

Training minimizes a Gaussian negative log-likelihood:

$$\mathcal{L} = \sum_{x,y} \log \sigma^2(x, y) + \frac{(U(x, y) - \mu(x, y))^2}{\sigma^2(x, y)}.$$

This formulation allows the network to adaptively weight regions with higher uncertainty, rather than enforcing uniform smoothing across the image.

Training is fully self-supervised and requires no clean spatial labels.

### Inference and Wiener-Style Fusion

At inference time, the predicted total variance is decomposed into signal and noise components. A noise variance estimate is obtained using a low quantile of the predicted variance distribution for each spatial component. The final denoised output is computed using a Bayesian fusion rule:

$$\hat{U} = \frac{\sigma^2_{\text{noise}}}{\sigma^2_{\text{total}}} \cdot \mu + \frac{\sigma^2_{\text{signal}}}{\sigma^2_{\text{total}}} \cdot U.$$

This Wiener-style filtering balances the model prediction with the original observation, preserving strong signal regions while suppressing noise-dominated pixels.

**Algorithm 2** Spatial Variance-Guided Denoising

**for** each spatial component $U_i$ **do**
    Normalize $U_i$
    Pad spatially to avoid boundary artifacts
    $(\mu_i, \sigma_i^2) \leftarrow \text{SpatialNetwork}(U_i)$
    $\sigma^2_{\text{noise},i} \leftarrow \text{Quantile}(\sigma_i^2, q)$
    $\sigma^2_{\text{signal},i} \leftarrow \max(\sigma_i^2 - \sigma^2_{\text{noise},i}, 0)$
    $U_i^{\text{denoised}} \leftarrow \frac{\sigma^2_{\text{noise},i}}{\sigma_i^2}\,\mu_i + \frac{\sigma^2_{\text{signal},i}}{\sigma_i^2}\,U_i$
    Denormalize and unpad
**end for**
**return** all denoised $U$

### PMD with Combined Spatial and Temporal Denoising

The trained spatial denoiser is wrapped as a PMD-compatible module and applied during PMD updates of the spatial basis. This allows denoising to be interleaved with factor updates, rather than applied as a standalone post-processing step.

**Algorithm 3** PMD with Integrated Spatial Denoising

**Require:** Video $X$, block size $b$, max rank $r$; Roughness thresholds $(\tau_s, \tau_t)$; Optional denoisers $\mathcal{D}_s, \mathcal{D}_t$

Estimate pixelwise mean $\mu$ and variance $\sigma^2$
**for all** overlapping blocks $\mathcal{B}$ **do**
    $X_{\mathcal{B}} \leftarrow \text{Standardize}(X[\mathcal{F}, \mathcal{B}])$
    Estimate temporal basis $V_{\mathcal{B}}$ from downsampled data

    **if** $\mathcal{D}_t$ exists **then** $V_{\mathcal{B}} \leftarrow \mathcal{D}_t(V_{\mathcal{B}})$
    **end if**
    Regress full-resolution data onto $V_{\mathcal{B}}$ to obtain $U_{\mathcal{B}}$

    **if** $\mathcal{D}_s$ exists **then** $U_{\mathcal{B}} \leftarrow \mathcal{D}_s(U_{\mathcal{B}})$
    **end if**
    Orthogonalize $(U_{\mathcal{B}}, V_{\mathcal{B}})$
    Retain components with roughness $< (\tau_s, \tau_t)$, Aggregate $U_{\mathcal{B}}, V_{\mathcal{B}}$ into global factors
**end for**
Regress full video onto aggregated spatial basis, **return** PMD factors $(U, V, \mu, \sigma^2)$

Spatial and temporal denoising address complementary noise sources. Temporal denoisers suppress frame-to-frame fluctuations, while spatial denoisers target pixel-level artifacts and fragmented structure. We therefore evaluate a combined strategy in which both denoisers are trained on an

initial PMD decomposition and jointly applied during a final PMD pass.

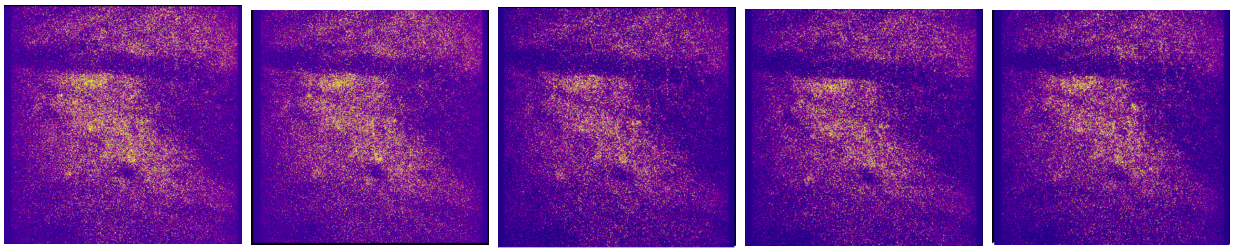
# 4 Data

## 4.1 Data used for Denoise-first pipeline

Experiments were performed on large-scale calcium imaging movies with dimensions $T \approx 15{,}000$ frames, $H, W \approx 512$, stored in memory-mapped format. However, the denoise-first method and the spatial denoising method were initially tested on different datasets. Everything was run on GPU.

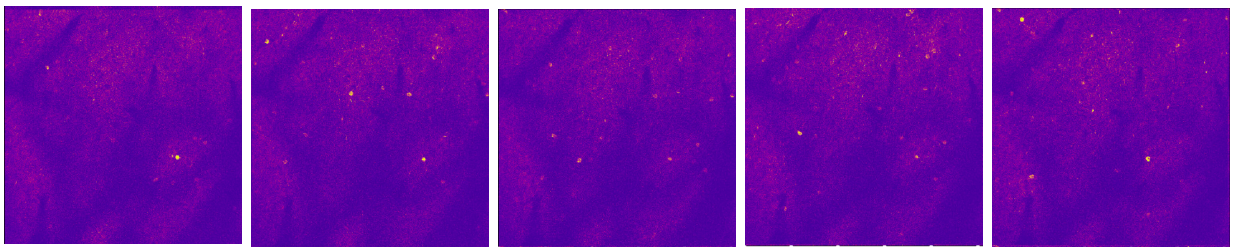
The denoise-first method was tested on two pre-motion corrected movies, both with dimensions 512 and and timeframes 10,000 and around 16,000. Preprocessing included reading raw movies, applying UV decomposition, training Gaussian-based or learned temporal denoisers, and performing piecewise-rigid motion correction.

All experiments were conducted using PyTorch-based implementations and the `maskNMF`/PMD framework. The denoise-first approach was tested on two such movies: one with 10000 frames, one with 16115 frames. Timestamped frames from both datasets are available in Figure 3, 4. We see a lot of noise, as well as a few bright yellow dots indicating neurons with the strongest signals at different points of the imaging process. We see that the first movie has more activity, and seems noisier than the second.



Frame 0 Frame 2414 Frame 5028 Frame 7969 Frame 9999

Figure 3: Five frames from the first raw movie.



Frame 0 Frame 2486 Frame 7613 Frame 11098 Frame 16114

Figure 4: Five frames from the second raw movie.

Next, we inspect the displacements (both vertical and horizontal) of the raw movie in order to see the effect of motion correction. This step is done by applying piecewise-rigid motion correction

to the movie, computing the shifts, and then looking at the deviations from the mean shifts.

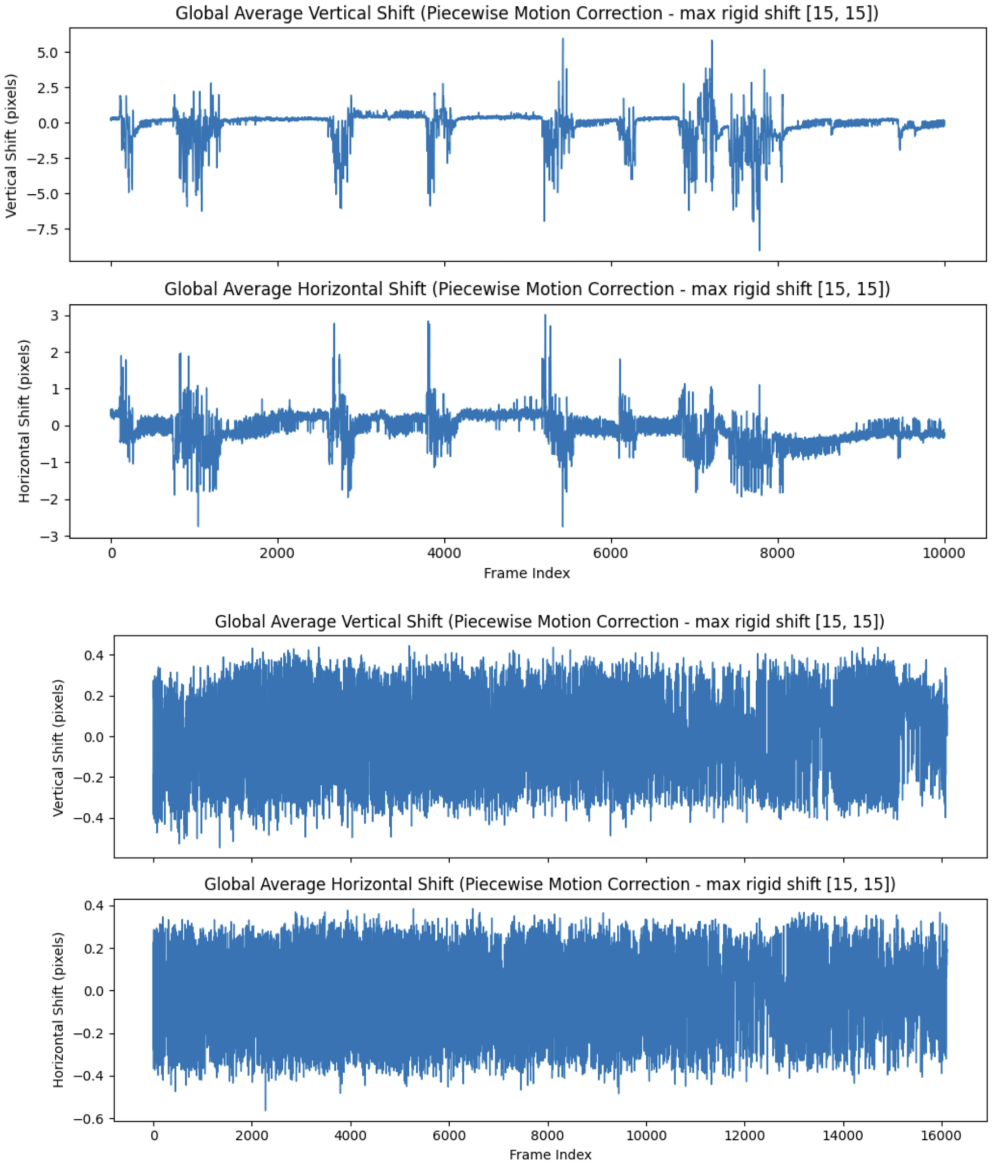


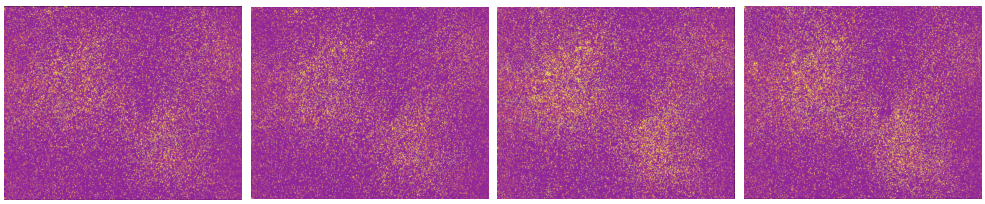

Figure 5: Shifts of both movies (top is the first, bottom is the second)

We can see in Figure 5 that the first raw movie is less stable than the second, as the shifts are on a much larger scale. That aligns what we see visually: the first movie (10,000 time frames) is much more shaky than the second. This is also evident in Figure 4, where the frames look consistent at each time stamp.

## 4.2 Data used for spatial denoising

The spatial denoising method was tested on one in-vivo post-motion corrected movies from IBL(4) that were readily available with dimensions 500 and 620 and timeframes 11700. The datasets were

then denoised using a decomposition, and a spatial denoiser was trained on the $U$ component before plugging it into the final compression.



Frame 1369 Frame 2035 Frame 3680 Frame 5799

Figure 6: Four frames from the raw movie used to denoise.

## 5 Evaluation Metrics

To compare the performance of the old and new pipelines, we need to come up with performance metrics. Usually, this step can be tricky as comparisons often tend to be done by visually comparing pre- and post-processing videos. For the purposes of this project, we want to evaluate the performance of the final compression, as well as compare the speed and visuals of each method.

So, for the compression metrics in the denoise-first approach, we look at the following measurements:

- PMD rank, which tells us how much the movie was compressed given its initial dimensions $W, H, T$
- Size compression (ratio between the sizes in GB) of original and compressed videos
- MSE of reconstructed frames, which calculates (normalized) mean squared error between the original and PMD-reconstructed frames
- Correlation between original and compressed frames
- Residuals - namely, to see if there is still noticeable signal in residuals
- Speed of compression for both pipelines and all denoising options
- Measurements such as VAF (variance accounted for) and PSNR (peak signal-to-noise ratio) to measure the quality of compression.

Since the existing pipeline already performs well, we just want to see if using a denoise-first approach, as well as swapping denoisers will affect performance in any meaningful way.

# 6 Results

### Denoise-first results

## 6.1 Revisiting Old Pipeline

The old pipeline, which trains a blind-spot temporal denoiser on motion corrected data and later passes it to the compression step, does a fairly good job producing a stable, denoised output as shown in Figure 7. We can see that the final PMD result camputer the activity both when it's weaker (top panel) and stronger (bottom panel).

The compression takes **6.3 minutes**, and has the final PMD rank of **8622**.

Table 1: Compression metrics for old (MOCO-first) pipeline

| **Metric** | **Dataset 1 (10,000 frames)** | **Dataset 2 (16,115 frames)** |
|---|---|---|
| Rank | 8622 | 27695 |
| Compression Ratio | 28.0x | 9.0x |
| nMSE | 0.80 | 0.69 |
| Correlation | 0.4373 | 0.5536 |
| Compression Speed | 6.3m | 15.1m |
| VAF | 22.26% | 33.03% |
| PSNR | 27.66 | 31.96 |

Table 1 shows that the old pipeline (MOCO-first) performs slightly differently on the two datasets. As we saw in Figures 3 and 4, the second dataset $\sim 1.62$ times larger than the first, and is much more stable based on the shifts seen in Figure 5. The compression performs better on this dataset (higher correlation between original and reconstructed frames, lower normalized MSE, higher variance accounted for, and higher PSNR. The metrics in which the compression performs better on the first dataset are compression speed (makes sense due to the difference in sizes), rank, and compression ratio. The residual plots can be seen in Figure 8, where we notice that both residual plots reflect some structure of the original videos, which may explain why the VAF metric wasn't good for either of the movie (both in the 30% range.

## 6.2 Denoise-First

**Performance at first glance:** We first evaluate the denoise-first pipeline using the same metrics. We used the same two raw movies to train a pre-motion correction temporal denoiser, then apply piecewise-rigid motion correction to the movies, and finally compress them using the final PMD that incorporates the pre-MOCO denoiser.

Figure 9 compares two original frames to denoised and compressed frames. We see that visually, the results from new and old pipelines (Fig 7 and Fig 9) look very similar – essentially identical. So, it makes sense to compare the two methods using the metrics seen in Table 1:

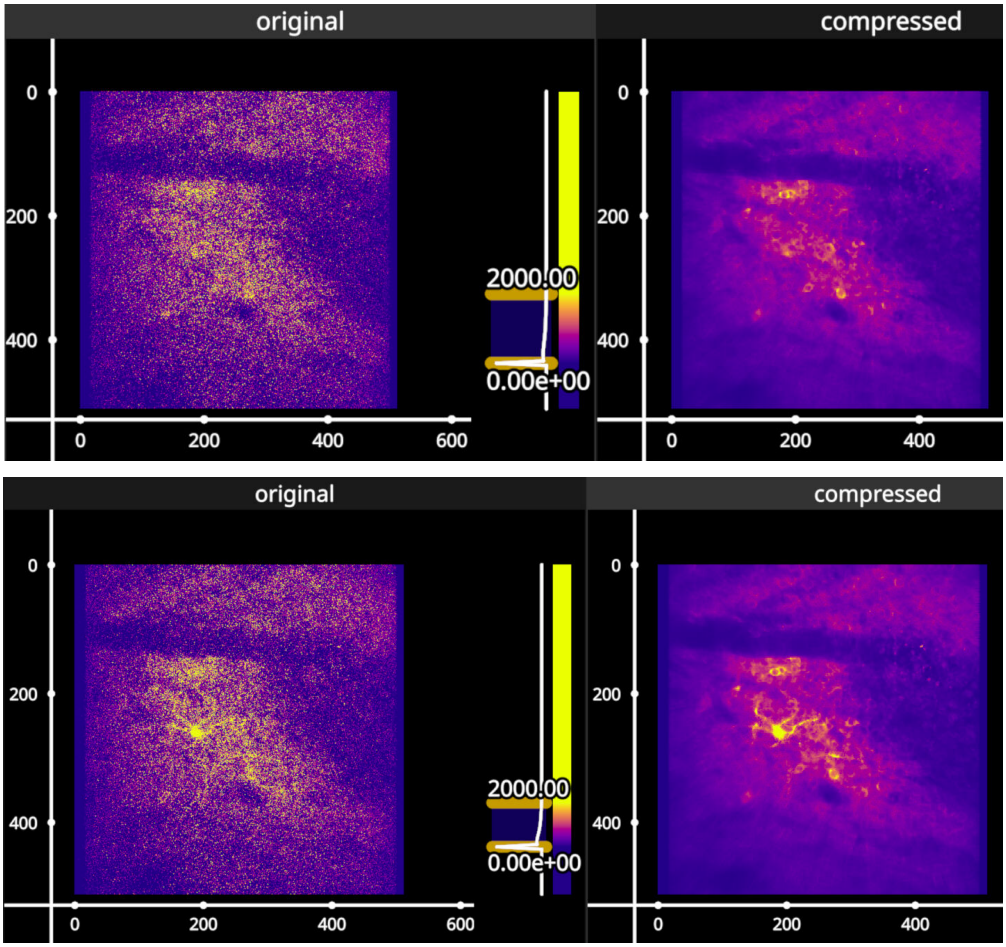



Figure 7: Old pipeline: original movie VS final compression at frames 0 and 8250; frames selected to best show activity.

Table 2: Compression metrics for **New** (Denoise-first) pipeline

| **Metric** | **Dataset 1 (10,000 frames)** | **Dataset 2 (16,115 frames)** |
|---|---|---|
| Rank | 8539 | 27689 |
| Compression Ratio | 27.7x | 9.0x |
| nMSE | 0.80 | 0.694 |
| Correlation | 0.4376 | 0.5528 |
| Compression Speed | 6.6m | 9.8m |
| VAF | 22.26% | 32.94% |
| PSNR | 27.66 | 31.95 |

One thing to note is that rank is an unstable metric, and changes slightly with each run. Still, it is useful when it comes to getting a general sense of PMD performance. The numbers reported in the Tables 1 and 2 show ranks from the corresponding experiments/runs. That being said, we see that a lot of metrics have remained almost unchanged when changing the order of denoising and motion correction: nMSE, correlation (up to three decimal points), VAF, PSNR all look identical. There's a slight difference in speed (6.6 vs 6.3 minutes), as well as rank (8539 vs 8622). The

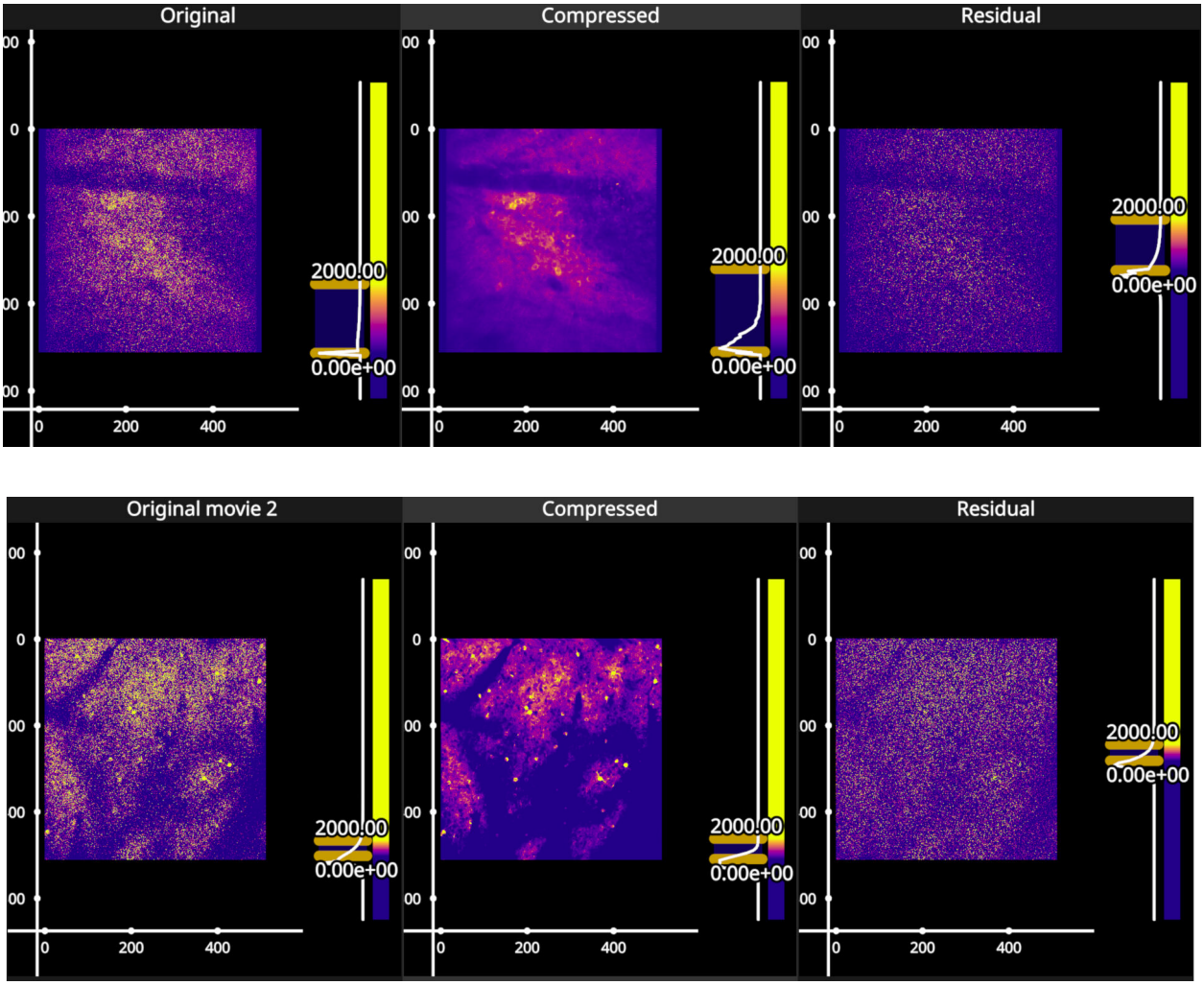



Figure 8: Old pipeline: Residuals for both movies (top panel is first movie, bottom panel is second movie

residuals for both movies analyzed also look very similar between the old and the new pipeline: they both show some weak signal, implying that the denoising step wasn't perfect and some signal leaked into the residual, as seen in top right corners of both rows of Figure 10.

So, the compression results from the denoise-first method don't look much different from MOCO-first pipeline. Any differences we see (speed, for example) is likely specific to the specific run of the experiment, much like the difference in rank. Next, we take a look at the difference in performance when using a Gaussian temporal denoiser instead of the blind-spot denoiser.

**Gaussian vs. Blind-spot denoiser** In order to see the impact of the choice of a denoiser on the final denoising/compression, we decided to implement a Gaussian denoiser, which has the advantage of not needing to be trained. That can save up to 10 minutes spent on training the blind-

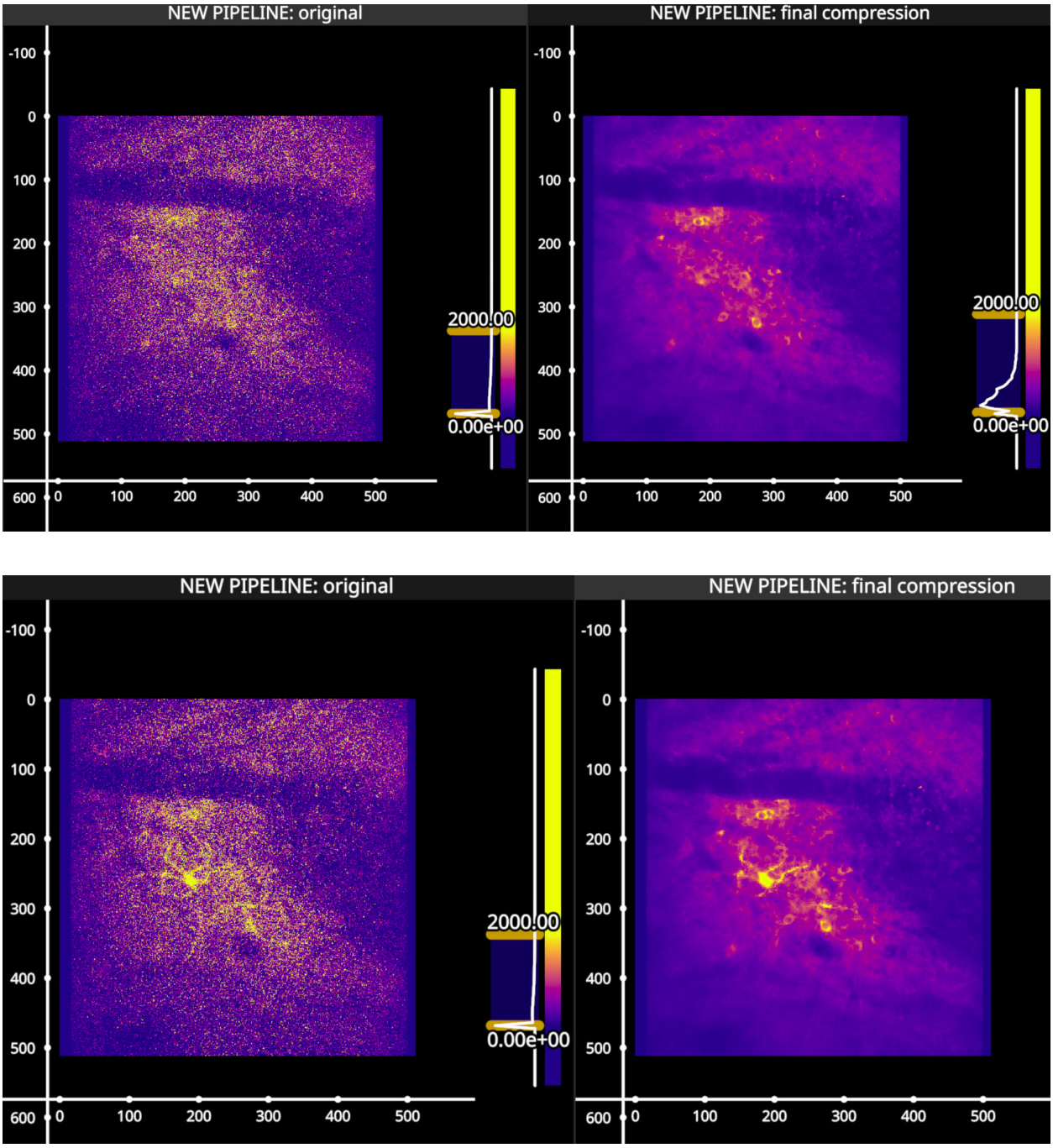



Figure 9: New pipeline: original movie VS final compression for frames 0 (top) and 8250 (bottom).

spot denoiser. The Gaussian denoiser is imported directly as a package (2), and altered slightly to work in the current neural network structure required by the PMD compression.
Next, we look at the compression speed and quality when using the Gaussian denoiser.

When comparing Tables 1, 2, and 3, the biggest observation is that using a Gaussian denoiser does not worsen any of the metrics. If anything, compression using Gaussian denoising achieves lower nMSE, higher compression ratio, comprable (not worse) VAF and PSNR, all while consis-

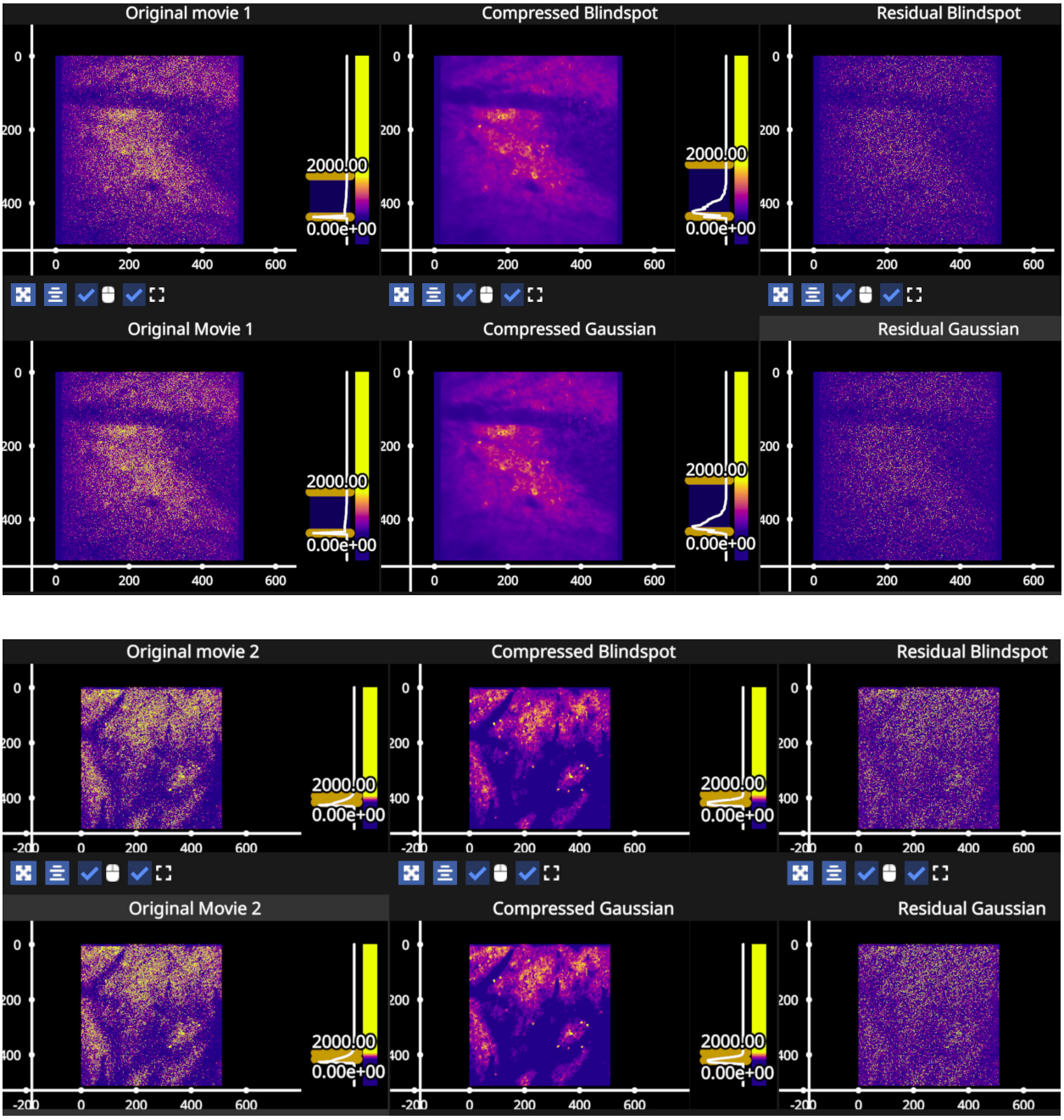



Figure 10: New pipeline: Residuals for both movies (top panel is first movie, bottom panel is second movie) and both denoisers

tently using lower final rank (6039 vs 8622 and 8539 for dataset 1, 21274 vs 27689 and 27695 for dataset 2). An even more noticeable difference is the speed of compression. By using a Gaussian denoiser, the compression time was consistently shrunk from 6.3 and 6.6 minutes to only 1.3 for the shorter movie (dataset 1), and from 15.1 and 9.8 minutes to only 1.8 minutes for the longer movie. Saving this much time while achieving better compression ratio without losing any significant signal or dropping VAF is a very strong result.

Table 3: Compression metrics for **New** (Denoise-first) pipeline using Gaussian temporal denoiser

| Metric | Dataset 1 (10,000 frames) | Dataset 2 (16,115 frames) |
|---|---|---|
| Rank | 6093 | 21274 |
| Compression Ratio | 39.5x | 11.7x |
| nMSE | 0.80 | 0.684 |
| Correlation | 0.4394 | 0.5622 |
| Compression Speed | 1.3m | 1.8m |
| VAF | 22.67% | 34.399% |
| PSNR | 27.67 | 32.02 |

Visually, there is indeed no clear difference between the compressed videos when using the two different denoisers. Figure 11 shows Gaussian and blind-spot denoisers, which yield identical results to the naked eye. Similarly, residuals for both movies look consistent between the denoisers (Figure 10), with Gaussian residual for the second movie seemingly having weaker signal than the blindspot denoiser.

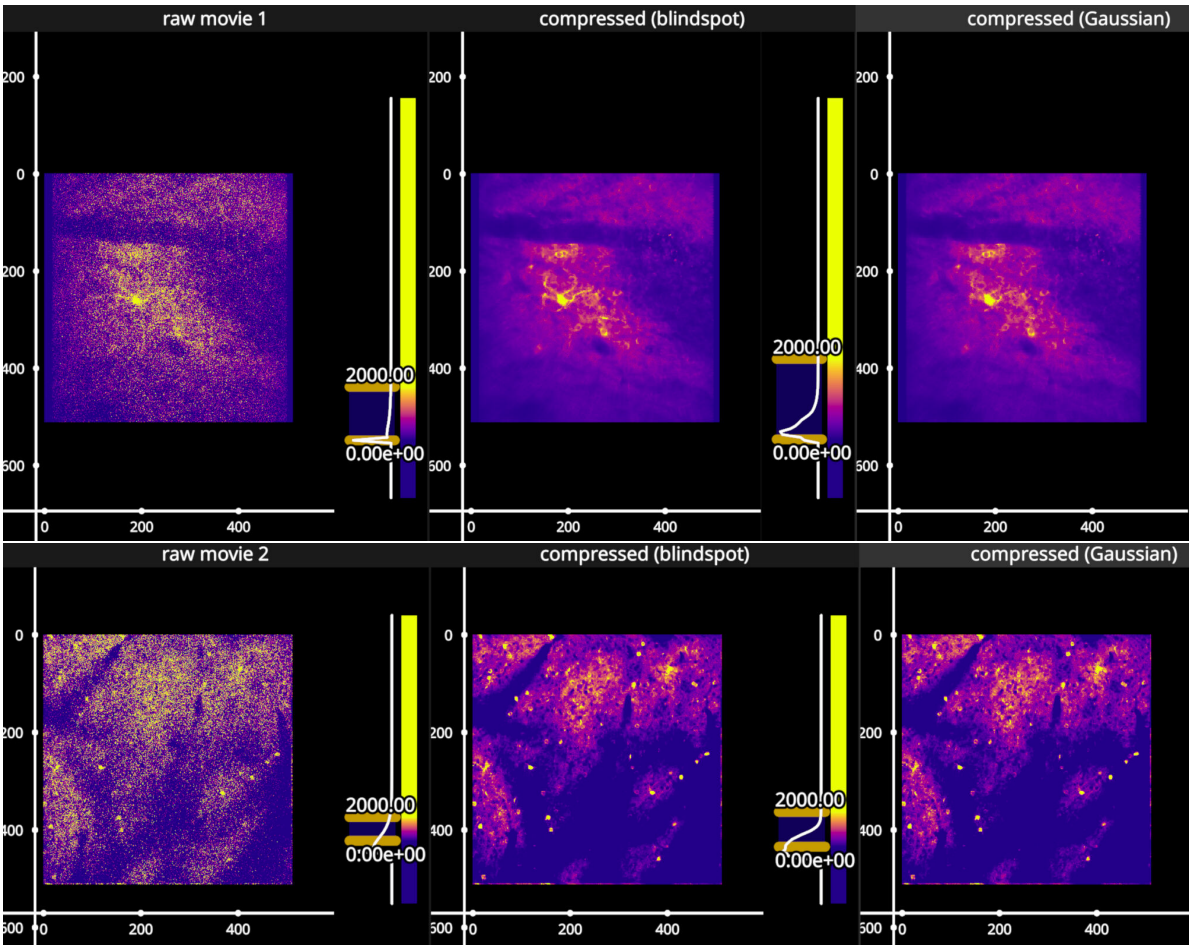



Figure 11: Raw movies, compressed (blindspot) and compressed (Gaussian) compared for frames 8250 (top) and 14032 (bottom).

**To summarize the performances of Gaussian vs Blindspot denoisers,** it seems like using the Gaussian denoiser saves considerable time in both the training section (needs to training) and the compression section (significant decreases in runtime) while maintaining (if not improving) compression quality and variance retained. However, blind-spot temporal denoising is much more interesting and less simplistic than the Gaussian smoothing.

### 6.3 Evaluation of Spatial Denoising and Spatial+Temporal Combined Denoising

We evaluate spatial denoising primarily at the level of individual spatial components by looking at visual performance of $U$ maps and PMD ranks.

Table 4: PMD Rank as an Implicit Spatial Roughness Indicator

| Method | PMD Rank |
|---|---|
| Baseline PMD | 2592 |
| Temporal Denoising Only | 2738 |
| Spatial Denoising Only | 2654 |
| Spatial + Temporal Denoising | 4700 |

In Table 4, PMD rank under different denoising strategies. Changes in rank reflect the interaction between denoising and PMD's spatial roughness constraints.

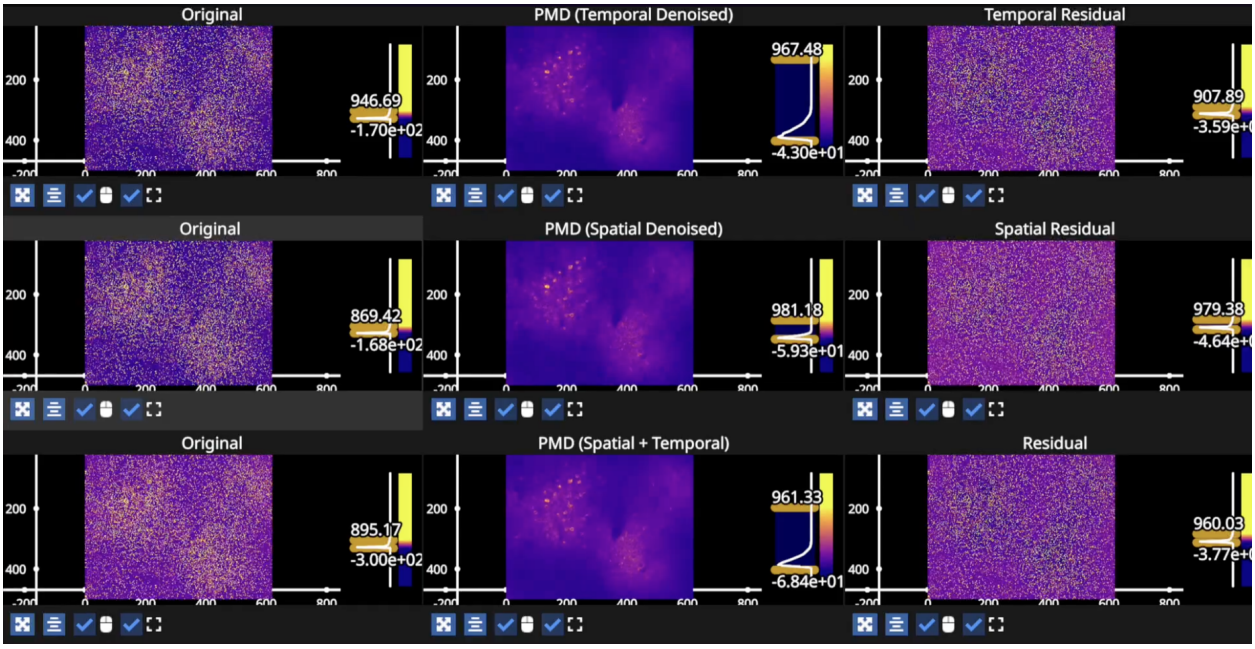



Figure 12: Comparison of PMD Denoising Strategies (Temporal, Spatial, and Spatial+Temporal) with Residual Analysis

As seen in 12, the temporal denoising result exhibits a more stable signal, whereas the spatial denoising introduces fluctuations over time (not shown here), leading to reduced stability. The

joint spatiotemporal result is largely identical to the temporal one, with only minor differences.

Qualitative comparisons show that spatial denoising produces cleaner components, particularly in low-SNR regions, see Figure 13. We looked at five different components denoted as yellow squares in the left column, and compared the spatial components $U$ before and after spatial denoising. We observe consistently cleaner outputs (third column). This shows that spatial denoising is effective, but we next need to compare the results to the existing pipeline (which uses temporal denoising), as well as the results of spatial+temporal denoising.

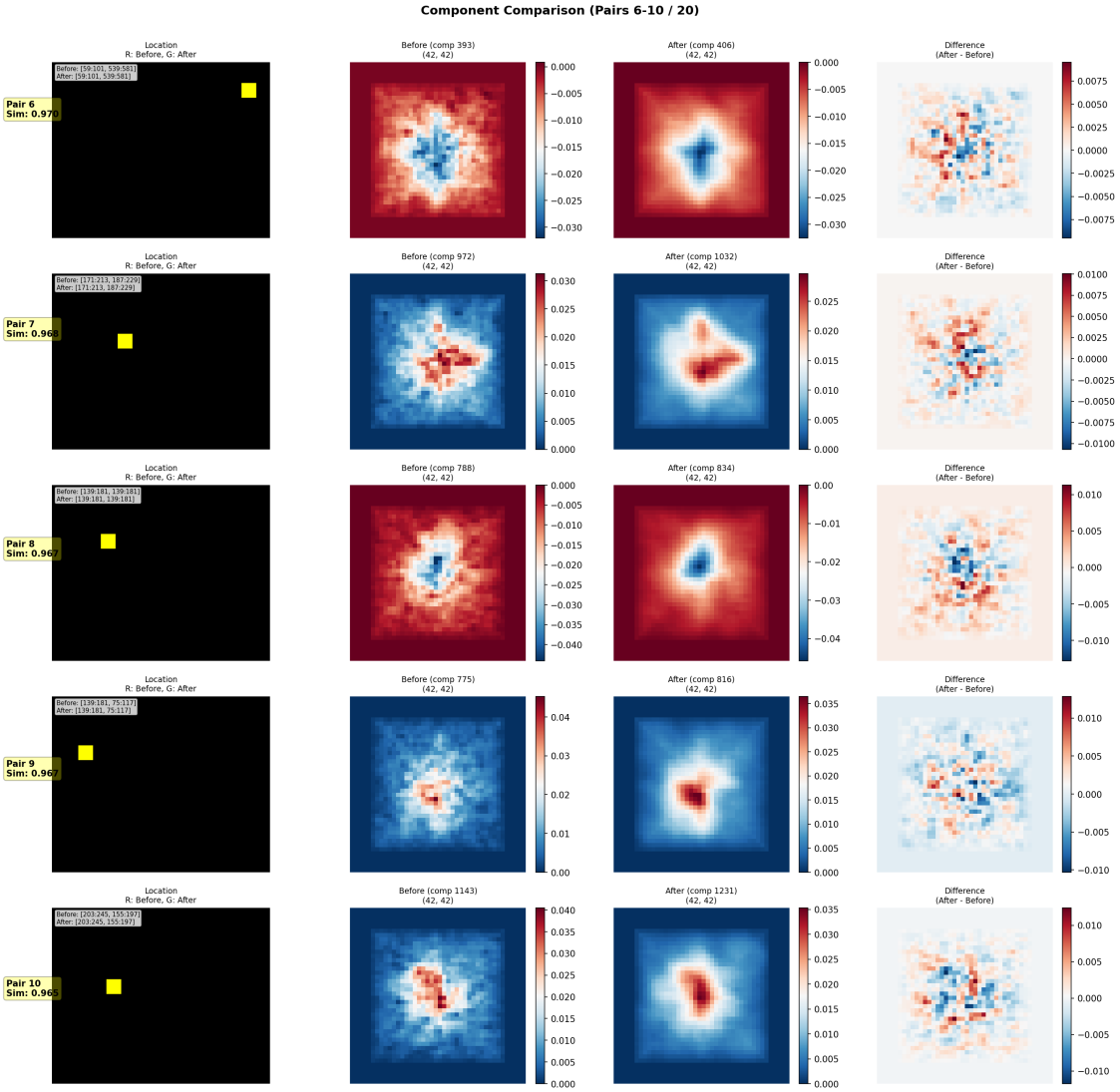



Figure 13: Side-by-side per-component ($U$) comparisons before and after **Spatial Denoising**

Next, Figure 14 shows the difference in PMD results with no-denoising, spatial-only, temporal-only, and joint temporal and spatial denoising (note: the temporal denoising used in this part is

the existing blind-spot temporal denoiser). The spatial-only denoising seems to result in slightly more blurriness, but also shows clearer shapes and structures. Spatial + temporal denoising yields even smoother results, with a most of background noise removed.

So, we see smoother results with spatial-only denoising, but we sometimes observe different shapes than expected (for example, row three Figure 14.) The joint spatial+temporal approach combines characteristics of both the spatial and temporal results.

This suggests that spatial and temporal regularization reinforce each other rather than competing. All three approaches (the temporal, spatial, and joint spatiotemporal methods) require less than 10 minutes to run, indicating comparable and efficient computational performance.

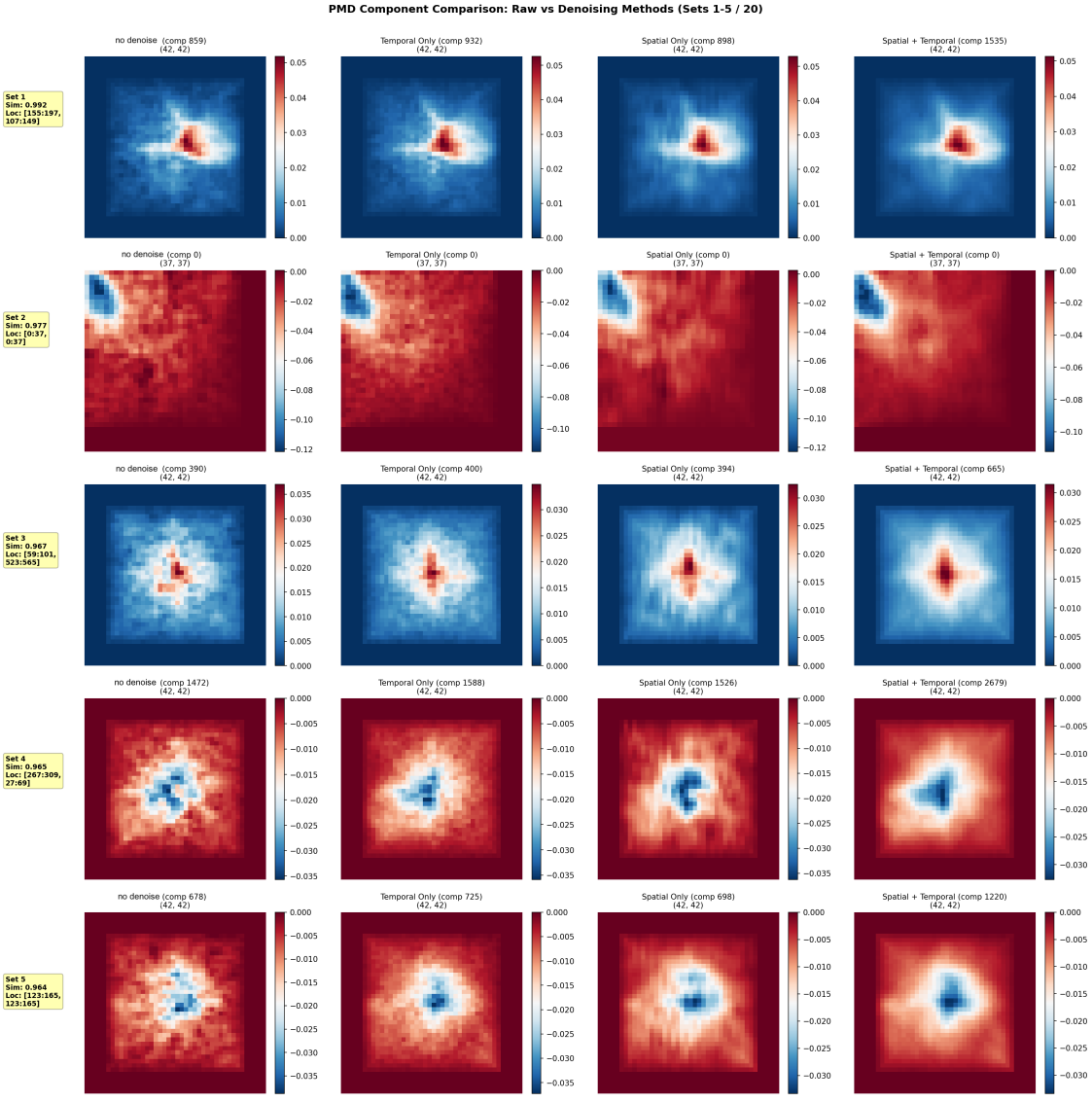



Figure 14: Comparison of PMD components under no-denoising, temporal-only, spatial-only, and combined denoising

# 7 Discussion

The goal of this project was to 1. investigate the impact of altering the existing pipeline for processing calcium imaging data by training a temporal denoiser on pre-motion corrected data, and 2. explore the performance of a spatial denoiser versus an existing temporal one.

For the denoise-first pipeline, we saw no major differences/advantages of either method over the other. Both methods (training the temporal denoiser pre and post motion correction) yielded similar results,yet resulted in different compressions scales. We did, however, see several benefits of using a more straightforward Gaussian denoiser instead of the existing blind-spot denoiser including significant decreases in runtime, and slight improvements in compression.

As for the spatial denoising, we see that it yields smoother, but less consistent results. The joint denoising, however, generates visually more interpretable results. Next, we can further evaluate the performance through downstream separation.

This project had several current limitations, which would all be interesting next steps. First, we only had access to two pre-motion corrected movies, one of which, as we saw in Figure 5, barely had any motion to begin with. It would be useful to see how the methods differ across more movies. Second, due to time restriction and long runtime of several parts of the code, we weren't able to perform as many experiments as we'd like, so the results (namely, measurements such as rank and runtime) had to be collected from only a handful of runs. Third, performances of compression, denoising, and motion correction tend to be difficult to evaluate, so it would be beneficial to brainstorm better ways to measure those performances. Last but not least, after seeing significant differences between blind-spot and Gaussian denoisers, it would be interesting to explore how other denoisers perform relative to one another.

As for the spatial denoising part of the project, we only had time to compare blind-spot spatial denoising to blind-spot temporal denoising. In other words, we didn't get to compare Gaussian (or any other) denoising methods to spatial denoising. As an immediate next step, we would do a joint temporal-spatial denoising using Gaussian and blindspot denoisers.

Finally, a major next step that we didn't have time for is integrating spatial (and joint spatial-temporal) denoising in the Denoise-first pipeline. We wouldn't expect to see a difference in perfomance, as the denoise-first method performed very similarly to the existing MOCO-first method. Still, it would be important to verify.

## References


[1] Amol Pasarkar. *maskNMF-toolbox*. GitHub repository, 2025. `https://github.com/apasarkar/masknmf-toolbox/tree/main`.

[2] Virtanen, P. et al. *SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python*. Nature Methods, 17:261–272, 2020.

[3] Pasarkar, A.*, Kinsella, I., Zhou, P., Wu, M., Pan, D., Fan, J. L., Wang, Z., Abdeladim, L., Peterka, D. S., Adesnik, H., Ji, N., and Paninski, L. *maskNMF: A denoise-sparsify-detect approach for extracting neural signals from dense imaging data*. 2023.

[4] IBL Flatiron Institute. Suite2p imaging plane data (MT009, 2021-08-30). `https://ibl.flatironinstitute.org/public/resources/2023-12-01_suite2p/MT009/20210830/1/suite2p/`