ECE 285 Project Report
Turning Back Time: Computational Techniques for Star Trail Removal from Long-Exposure Astrophotography

Avanti Bhandarkar
Electrical and Computer Engineering
A59026610

Nikhil Gandudi Suresh
Electrical and Computer Engineering
A59022903

Abstract

Astrophotography, a popular yet challenging hobby, often involves capturing long-exposure images of the night sky. A common issue in long-exposure astrophotography is the formation of star trails, which occur due to the Earth's rotation and are aesthetically pleasing but not always desirable. Current methods to avoid star trails include using star trackers, which are costly and bulky, making them inaccessible for many enthusiasts. This project aims to develop computational techniques for removing star trails from long-exposure astrophotography images without the need for expensive equipment. We explore various deconvolution methods, including Wiener Deconvolution, Richardson-Lucy Algorithm, Gradient Descent, and Fast Iterative Shrinkage Thresholding Algorithm (FISTA) with Total Variation, to address this problem. We also generate synthetic images to test our methods and employ image priors to enhance the deconvolution results. The code is at https://github.com/GSNikhil/star-trail-removal.

1 Introduction

Astrophotography is an increasingly popular hobby that allows photography enthusiasts to capture the beauty of the night sky. However, it comes with several technical challenges, particularly when attempting to photograph celestial objects in low-light conditions. Long-exposure astrophotography involves keeping the camera's shutter open for extended periods, allowing more light to reach the sensor and thereby revealing fainter objects in the sky and allowing foreground objects to remain visible. However, the Earth's rotation during these long exposures causes stars to appear as streaks or trails across the image, rather than as pinpoints of light. While these trails can be aesthetically pleasing, they are not always desired, especially when the goal is to capture pinpoint stars or specific astronomical events.

Star trails are a natural consequence of the Earth's rotation, which causes stars to appear to move across the sky in circular paths centered around the celestial poles. In the Northern Hemisphere, Polaris (or the Pole Star), remains relatively fixed near the celestial North Pole, while other stars trace circular arcs around it, becoming more pronounced with longer exposure times. Current techniques to avoid star trails often involve the use of star trackers to align the camera's movement with the stars, thus preventing the trails from forming during long exposures. Star trackers are also utilized in other fields such as satellite imaging and professional astronomical observations, where precise tracking is crucial. Despite their effectiveness, the high cost and bulkiness of these devices make them impractical for many amateur astrophotographers, who seek more accessible and less intrusive solutions to achieve trail-free images.

In response to these limitations, our project attempts to develop a computational method for removing star trails from long-exposure astrophotography without the need for expensive or bulky hardware. By leveraging deconvolution techniques, we aim to provide an efficient solution that enhances the quality of astrophotographic images. These techniques are designed to reverse the blurring effects

caused by star trails, restoring the image to a state that more closely resembles a short-exposure shot with pinpoint stars, while also retaining the well-lit foreground resulting from a long-exposure shot.

To evaluate the effectiveness of our proposed methods, we generate synthetic star trail images and apply our algorithms to these test cases. We also incorporate image priors to further refine our results, ensuring that the reconstructed images maintain a high degree of realism and detail. We achieve satisfactory results with synthetic images, which are validated by high Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity (SSIM) Index values across most of our deconvolution techniques. We further verify the qualitative value of our algorithms by reconstructing a 'real' star trail image i.e. deblurring star trails into pinpoint stars while retaining the foreground and visual appeal of the original image.

2 Related Work

Tai et al. [1] presented a Richardson-Lucy deblurring algorithm with regularization priors. The proposed algorithm also used a projective motion path blur model to capture spatially varying blur. Liu et al. [2] proposed an improved Richardson-Lucy algorithm based on radial basis function neural network (RBFNN). They estimated the PSF by using the angular velocity information provided by the gyroscope, before using the RBFNN to to predict the number of iterations required by the RL algorithm to complete the star image deblurring. Jiang et al. [3] also proposed an accelerated Richardson-Lucy algorithm, which incorporated an improved Radon transform method by introducing a combination of Z-function and double threshold mask.

Mourya et al. [4] proposed a blind image deblurring algorithm for a single long-exposure images, where they imposed different priors on point-like and smooth extended sources, and recovers two separate maps for them. They also used appropriate image priors to improve reconstruction. Another interesting PSF estimation method was proposed by Liu et al. [5] where they used a minimum bounding box to describe the space targets and used the box's parameters for kernel estimation. Zhang et al. [6] built on the principle that the intensity of clear star image is in accordance with Laplacian distribution and introduced a lp-regularized deblurring method based on star image intensity. Wang et al. [7] proposed an algorithm to reduce motion blur in star images captured by a sensor. They used a MEMS gyroscope to deduce motion trajectory, and correct blur kernels and employed a reconstruction method for efficiency.

More recently, Whisler and Dai [8] developed a method for removing spatially varying motion blur, which can be used to remove star trails. They experimented with more priors and deblurring algorithms and obtained good results with ADMM and a TV prior. Hou et al. [9] introduced a real-time method to tackle motion blur in star sensors under high dynamics. By employing fast blur kernel estimation and area filtering techniques, they were able to remove trailing stars in a computationally efficient way, even under low SNR conditions.

3 Implementation

3.1 Data Collection and Generation

3.1.1 Real Images

Real image data is collected from public image sharing websites such as Flickr, Unsplash and Imgur. The images are selected according to the following criteria:

- Images containing spatially sparse, concentric star trails (for initial experiments).
- Images containing minimal light pollution noise.
- Images with Polaris (Pole Star) visible.
- Images with readily available metadata about the time of exposure (needed for kernel estimation).

The greatest drawbacks of these images is the lack of a ground truth; we do not have any corresponding short-exposure images to validate our deconvolution results. To counter this, we also generate synthetic star trail images, which are used in our quantitative analyses.

3.1.2 Synthetic Images

Synthetic star trail images are generated using the three-step process shown in Figure 1.

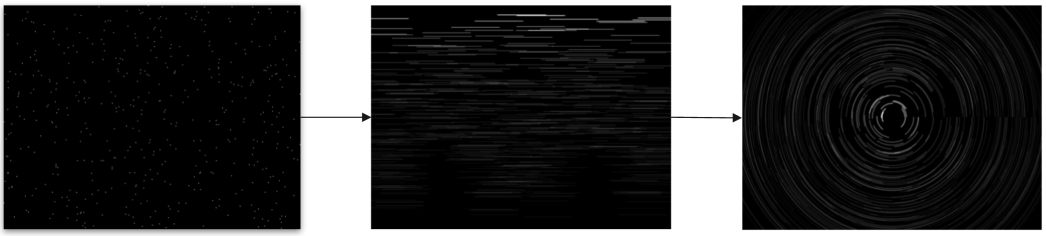

Figure 1: Left: Randomly generated 1 pixel 'stars'. Center: Polar image, blurred with a known kernel. Right: Cartesian image, rotated by a known number of degrees.

The Cartesian image is formed by rotating the Polar image through 5, 10, 15 and 30 degrees. This process acts as our forward model for star trail formation and provides control over parameters such as:

- Degree of blur / time of synthetic exposure - varies by angle of rotation
- Density of star trails - varies by number of randomly generated pixels
- Star intensity - varies by intensity of randomly generated pixels

3.2 Data Preprocessing

3.2.1 Binarization/K-Means Clustering

The first step in image preprocessing is separating star trails from the background or sky. We use either binarization or K-means clustering to isolate the star trails, based on the level of detail in the image we are working with.

For simple images with no foreground objects (i.e. an image containing only the sky and star trails) we perform global binarization. A single threshold value is chosen, and all pixel values above this threshold are set to white (255), while all pixel values below this threshold are set to black (0). Our threshold is set at 128, to prevent the dark background from being retained while also accounting for slight light pollution and background intensity variations.

For complex images with multiple foreground objects, we use the K-Means clustering algorithm to partition the image into 2 clusters. The cluster to which the pixel at [0,0] belongs is set to black (0) and the other cluster is set to white (255). This assignment works for most images since the top-leftmost pixel is usually a sky/background pixel. We also use K-Means clustering to generate binary masks for foreground objects; these masks are used in the recovery process of real images.

3.2.2 Estimating the Center of Rotation

Star trails appear as concentric circles centered at the celestial North Pole or at Polaris when a camera is aimed at either point. We use Circular Hough Transform to detect the circles formed by star trails and their collective center, which is also the center of rotation for our problem.

In a given circle, for each edge point (x,y) and for each possible radius r, potential circle centers (a,b) are calculated using the parametric equations of a circle:

$$a = x–rcos(\theta)$$
$$b = x–rsin(\theta)$$

These values are accumulated and the highest-valued (a,b) pair is assigned as the center of the circle.

However, Circular Hough Transform is not always accurate in estimating the center of rotation, especially when dense star trails are being processed. To overcome this, we implement an alternate

method which allows the user to select the center of rotation manually, by clicking on it in a star trail image. The coordinates of the pixel being clicked are stored and used as the center of rotation in future steps.

3.2.3 Polar Transform

The final preprocessing step is converting the image from Cartesian coordinates (circular star trails of varying lengths i.e. spatially varying) to Polar coordinates (horizontal lines of uniform length i.e. spatially invariant).

The polar transform converts Cartesian coordinates (x,y) to polar coordinates (r, θ), where:

- r is the distance from the origin (center of rotation) to the point (x,y)
- θ is the angle between the line connecting the origin to the point (x,y) and a reference direction (we use the positive x-axis as our reference direction)

The transformation equations are:

$$r = \sqrt{(x–x_0)^2 + (y–y_0)^2}$$

$$\theta = \arctan *2(y–y_0, x–x_0)$$

where (x_0, y_0) is the center of rotation.

The generated Polar image is formed such that a pixel at (r, θ) maps to the corresponding intensity value in the original Cartesian image at (x,y). This effectively "unwraps" the circular star trails into straight lines.

3.3 Estimating of the Deblurring Kernel

To estimate the deblurring kernel (or PSF), only the exposure time is required. A star in the night sky would appear in the same position in 24 hours. In other words, in 24 hours the star covers 360^o (on the projected plane). The angle traversed in h hours is -

$$\theta_h = \frac{h \times 360}{24} = 15h$$

If the polar image is $R \times T$, then there are T samples in 360^o. The number of samples in θ_h is (say N)-

$$N = ceil\left(\frac{T \times \theta_h}{360}\right)$$

$$kernel = \begin{bmatrix} 0 & \dots 0 & 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & \dots 0 & 0 & 0 & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots 1 & 1 & (Ntimes) & 1 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots 0 & 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & \dots 0 & 0 & 0 & 0 & 0 & \dots & 0 \end{bmatrix}_{R \times T}$$

3.4 Deconvolution Methods

Based on the above sections, the star trail problem can be posed as a non-blind deconvolution problem.

Let b is the measurement (star-trail) image, h be the point-spread function acting on the image/scene l, and let n be the noise.

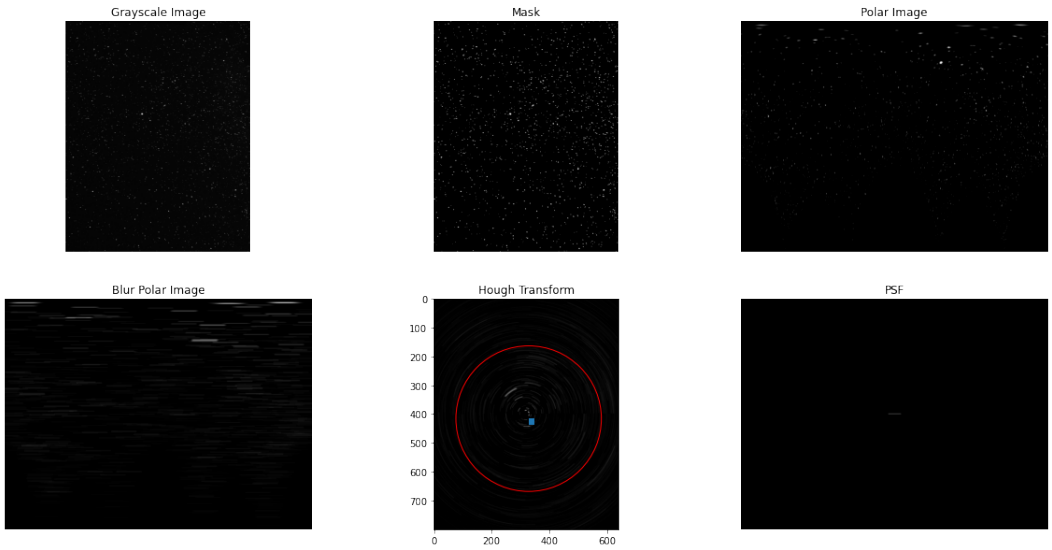

Figure 2: Top Left: Original stars Image. Top Center: Mask or output of Binarization/K-Means Clustering. Top Right: Image in the polar transform. Bottom Left: Polar image with 30degrees blur applied. Bottom Center: Hough Transform to find the center. Bottom Right: Known Point Spread Function

$$b(x, y) = h(x, y) * l(x, y) + n(x, y)$$

where, $*$ represents convolution.

We discuss various methods and the respective outputs below.

3.4.1 Weiner Deconvolution

The Wiener deconvolution method aims to recover the original image $l(x, y)$ from a blurred and noisy observation. The process works in the Fourier domain and can be described by the following equation:

$$l(x, y) = \mathcal{F}^{-1}\left(\frac{1}{H(f)}\left[\frac{1}{1 + \frac{\sigma^2}{|H(f)|^2 S(f)}}\right]\right)$$

where, $H(f)$ is the Fourier transform of $h(x, y)$, $S(f)$ is the mean spectral density of $L(f)$, σ^2 is the mean power spectral density of noise, and $\mathcal{F}^{-1}$ denotes inverse Fourier transform. Wiener deconvolution suffers from ringing artifact, which can be observed in Fig 1.a

3.4.2 Richardson-Lucy Algorithm

Richardson-Lucy is an iterative method to recover an image from a blurred and noisy observation. It works on the principles of maximum likelihood estimation. The update rule can be written as:

$$\hat{l}_{t+1}(x, y) = \hat{l}_t(x, y)\left(\frac{b(x, y)}{\hat{l}_t(x, y) * h(x, y)} * h^*(x, y)\right)$$

3.4.3 Vanilla Gradient Descent

The forward model, as discussed above, can also be written as a matrix product.

$$b(x,y) = h(x,y) * l(x,y) + n(x,y)$$
$$b = circ(h)pad(L) + N$$

In simpler terms, $b = AL + N$, where $A = circ(h)$.

$$\hat{L} = \arg\min_L \frac{1}{2}||AL - b||_2^2$$
$$\nabla_L \frac{1}{2}||AL - b||_2^2 = \nabla_L \frac{1}{2}(AL - b)^T(AL - b) = A^T(AL - b)$$

Gradient descent is a fundamental optimization algorithm used to minimize a function by iteratively moving towards the minimum value of the function. It's widely used in machine learning and image processing for optimizing loss functions.

The basic idea of gradient descent is to adjust the parameters of a function to minimize the cost (or loss) function $J(l)$. The update rule for gradient descent is:

$$\hat{l}_{t+1}(x,y) = \hat{l}_t(x,y) - \nabla J(\hat{l}_t(x,y))$$
$$\hat{l}_{t+1}(x,y) = \hat{l}_t(x,y) - A^T(AL - b)$$

Gradient Descent can be computationally expensive and slow.

3.4.4 Fast Iterative Shrinkage Thresholding Algorithm - FISTA

FISTA allows incorporating non-smooth constraints such as non-negativity, L1 norm, and total variation in the optimization objective. The FISTA used in the project also used Nesterov accelerated gradient descent with a projection operation to enforce non-negativity on the recovered image.

$$t_{k+1} = \frac{1 + \sqrt{1 + 4t_k^2}}{2}$$
$$\beta = \frac{t_k - 1}{t_{k+1}}$$
$$y_{k+1} = l_k - \mu\nabla_x||AL - b||_2^2$$
$$z_{k+1} = PROJ(y_{k+1})$$
$$l_{k+1} = z_{k+1} + \beta(z_{k+1} - z_k)$$

where μ is the step size, and t and β are the Nesterov acceleration terms.

3.4.5 Alternating Direction Method of Multipliers (ADMM)

ADMM is an optimization algorithm that can solve complex problems by breaking them into smaller subproblems, so that each of which is easier to handle. We exploit the separable structure of our problem (our objective function can be decomposed) to use ADMM effectively.

We aim to solve a general minimization problem

$$min_{x,z} f(x) + g(z)$$

such that

$$Ax + Bz = c$$

where f(x) and g(z) are convex functions, A and B are matrices and c is a vector.

ADMM expresses this problem in an augmented Lagrangian form as:

$$\mathcal{L}_\rho(x, z, \lambda) = f(x) + g(z) + \lambda^T(Ax + Bz - c) + \frac{\rho}{2}\|Ax + Bz - c\|_2^2$$

where λ is the Lagrange multiplier and $\rho > 0$ is a penalty parameter.

The update rules for achieving convergence are:

$$\begin{aligned} x^{k+1} &= \arg\min_x \mathcal{L}_\rho(x, z^k, \lambda^k) \\ z^{k+1} &= \arg\min_z \mathcal{L}_\rho(x^{k+1}, z, \lambda^k) \\ \lambda^{k+1} &= \lambda^k + \rho(Ax^{k+1} + Bz^{k+1} - c) \end{aligned}$$

3.5 Incorporating Image Priors

Priors are used to incorporate additional information about the image to improve reconstruction quality. Priors can be viewed as regularization terms added to the optimization problem to impose certain desirable properties on the solution, such as sparsity, smoothness, or edge preservation.

3.5.1 FISTA with Total Variation

Total Variation prior is used to promote piecewise constant solutions, preserving edges. This is done by adding penalty term on the 1D gradient of the image in both x, and y directions - $||\nabla_x(l(x, y))||_1 + ||\nabla_y(l(x, y)||_1$

$$\hat{l} = \arg\min_L \frac{1}{2}||Al - b||_2^2 + \lambda(||\nabla_x(l(x, y))||_1 + ||\nabla_y(l(x, y)||_1), l > 0$$

where λ is the regularization factor. Again, FISTA was used for the optimization to find $l(x, y)$.

3.5.2 ADMM with Total Variation

When incorporating the Total Variation prior into ADMM, the objective function g(z) includes the TV term, which is expressed as:

$$g(z) = \|Dz\|_{\text{TV}}$$

Reformulating the Lagrangian, we get:

$$\mathcal{L}_\rho(x, z, \lambda) = f(x) + \|Dz\|_{\text{TV}} + \lambda^T(Ax + Bz - c) + \frac{2\rho}{2}\|Ax + Bz - c\|_2^2$$

It can be observed that g(z) has simply been replaced with the TV regularization term.

4 Results and Discussion

4.1 Metrics

Two metrics were used to evaluate the results - Peak Signal to Noise Ratio and Structural Similarity.

Peak Signal to Noise Ratio (PSNR) - It is the ratio between the maximum possible power of a signal and the power of any noise that might distort it.

$$MSE = \frac{\sum_{X,Y}(I_1(x, y) - I_2(x, y))^2}{XY}$$

$$PSNR = 10log_{10}\frac{255^2}{MSE}$$

Structural Similarity (SSIM) - SSIM measures the difference between select properties (luminance, contrast and structure) of pixels in a pair of images.

$$SSIM(I_1, I_2) = \frac{(2\mu_1\mu_2 + C_1)(2\sigma(I_1, I_2) + C_2)}{(\mu_1^2 + \mu_2^2 + C_1)(\sigma(I_1)^2 + \sigma(I_2)^2 + C_2)}$$

4.2 Results on Synthetic Images

An image of stars was used and a blur was applied in the polar domain to generate star trails. The displayed results are only for a blur equivalent to 30 degrees (2 hours) of exposure. The pre-processing is shown in 2.

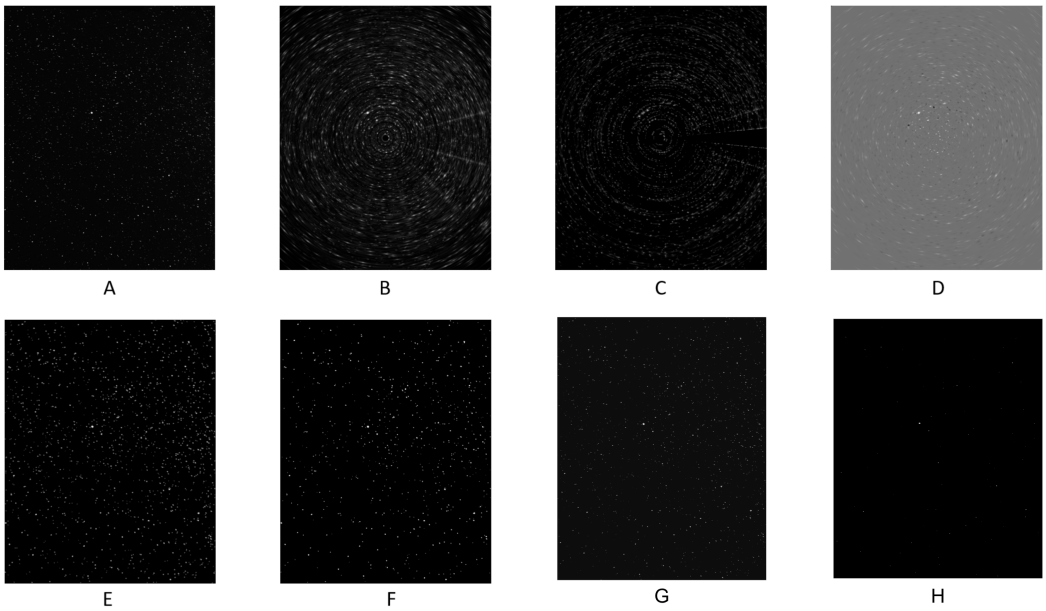

Figure 3: Deconvolution Results. A - Original Image, B - Wiener Deconvolution, C - Richardson-Lucy Deconvolution, D - Vanilla Gradient Descent, E - FISTA, F - FISTA + TV, G - ADMM, H - ADMM + TV

Table 1: Results on Various Deconvolution Algorithms

Degrees	**5 deg**		**10 deg**		**15 deg**		**30 deg**	
Method	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
Wiener Deconvolution	27.383	0.24991	28.054	0.36394	27.666	0.3724	27.521	0.38981
Richardson-Lucy	25.408	0.15976	26.164	0.16864	25.307	0.15363	25.765	0.19441
Gradient Descent	17.023	0.25442	18.028	0.26792	17.696	0.25074	16.72	0.21872
FISTA	28.623	0.23082	28.625	0.23089	27.867	0.2159	27.967	0.21601
FISTA + TV	27.228	0.20726	27.043	0.23189	26.966	0.25377	26.967	0.31941
ADMM	19.328	0.16726	21.043	0.15689	22.456	0.12771	22.656	0.13671
ADMM+TV	21.453	0.14078	22.847	0.13975	25.853	0.19462	24.676	0.11601

4.3 Results on a Real Image

Section 4.2 shows quantitative results on synthetically generated images. To evaluate the qualitative results of our proposed method, we aim to recover a visually appealing real image. Figures 4-7 show the image recovery pipeline and the recovered output image. While this result is visually appealing and can be considered suitable qualitatively, we cannot evaluate the accuracy of star location due to the lack of a ground truth.

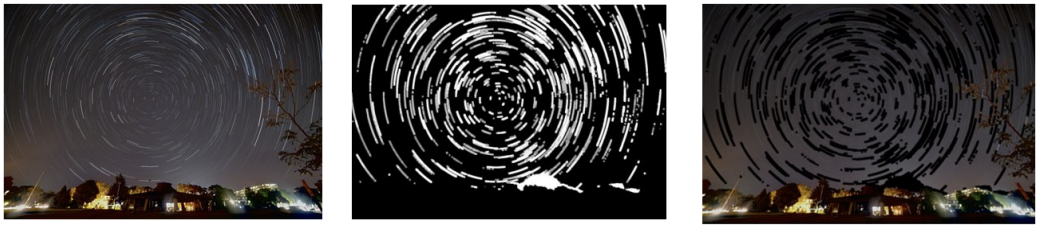

Figure 4: Left: Original image. Center: Dilated star trail mask. Right: Original image with star trails subtracted.

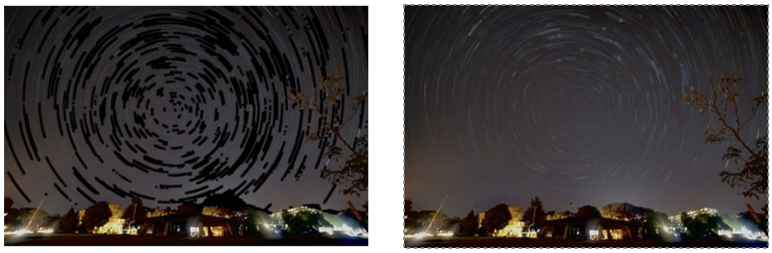

Figure 5: Left: Original image with star trails subtracted. Right: Inpainted star trails.

4.4 Challenges

- **Finding Good Input Images** - most star trail images are composites of multiple shorter exposure images. Often, these images have incomplete star trails, where a small segment of the trail is missing. This makes deconvolution challenging, as the algorithm may interpret the broken trails as separate stars.
- **Generating an Accurate Foreground Mask** - generating a mask that accurately isolates the star trails is crucial for effective deconvolution. Classical methods struggled to create precise segmentation masks for the stars.
- **Finding the Center of Rotation** - Even a slight error in identifying the coordinates of the center of rotation can result in a polar transform where the star trails are not properly aligned. This misalignment significantly impacts the deconvolution performance.

4.5 Drawbacks of the Proposed Method

4.5.1 Reliance on Concentric Trails

- Inaccurate Center Estimation: Any minor error in locating the center of rotation can cause deviations in the expected circular patterns, leading to artifacts in the reconstructed image.
- Real-World Variability: In real-world scenarios, slight deviations from perfect concentricity can occur due to factors like atmospheric distortion or slight camera movements, which the method cannot accommodate, reducing the robustness of the approach.

4.5.2 Poor Performance on Dense Trails

- Overlapping Trails: The deconvolution process becomes less effective as overlapping trails can be misinterpreted, resulting in blurred or merged star trails in the recovered image.
- Noise Amplification: Dense trails can amplify noise in the image, making it harder to distinguish between actual star trails and noise, leading to less accurate deconvolution results.

Figure 6: Left: Inpainted star trails. Center: Foreground mask. Right: Gaussian blur applied to inpainted image, excluding the foreground.

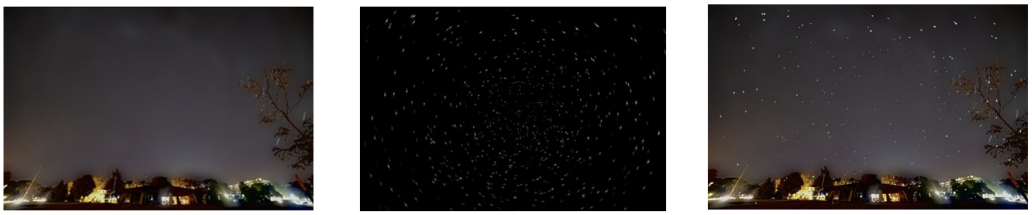

Figure 7: Left: Blurred image. Center: Reconstructed point stars. Right: Composite output image.

5 Conclusion

In this paper, we explored various deconvolution techniques for removing star trails from long-exposure astrophotography images. Our investigation included the application of methods such as Wiener Deconvolution, Richardson-Lucy Algorithm, Gradient Descent, FISTA and ADMM. The results from our synthetic and real image experiments demonstrate that while our methods can effectively reduce or remove star trails, challenges remain, particularly with dense star fields and real-world variabilities. The reliance on accurately estimating the center of rotation and dealing with overlapping trails are also significant issues that need further refinement.

Some potential further directions may include:

- Improving the robustness of our methods against real-world variabilities, such as slight deviations from perfect concentricity in star trails.
- Exploring advanced noise reduction techniques and more robust deconvolution algorithms that can accurately separate overlapping trails without amplifying noise.
- Integrating machine learning approaches, such as convolutional neural networks (CNNs) or generative adversarial networks (GANs), to potentially enhance the accuracy and efficiency of star trail removal.
- Enhancing the computational efficiency of our methods to enable real-time processing by optimizing the existing algorithms or leveraging hardware acceleration technologies such as GPUs to speed up the processing time.

By addressing these future directions, we can further improve the practicality and effectiveness of computational techniques for star trail removal, making high-quality astrophotography accessible to more enthusiasts regardless of star tracker availability.

6 Contributions

Avanti worked on the image preprocessing pipeline and wrote code for the ADMM and ADMM with TV algorithms. She was also responsible for the entire real image deconvolution and recovery pipeline.

Nikhil worked on the synthetic image pipeline and wrote code for most of our deconvolution methods and algorithm evaluation. He also cleaned up our codebase and converted it into functions.

Avanti and Nikhil contributed equally to the presentation and the report.

References

[1] Tai, Yu-Wing, Ping Tan, and Michael S. Brown. "Richardson-lucy deblurring for scenes under a projective motion path." IEEE Transactions on Pattern Analysis and Machine Intelligence 33.8 (2010): 1603-1618.

[2] Liu, Di, Xiyuan Chen, and Xiao Liu. "An improved Richardson-Lucy algorithm for star image deblurring." 2019 IEEE International Instrumentation and Measurement Technology Conference (I2MTC). IEEE, 2019.

[3] Jiang, Jie, Junnan Huang, and Guangjun Zhang. "An accelerated motion blurred star restoration based on single image." IEEE Sensors Journal 17.5 (2016): 1306-1315.

[4] Mourya, Rahul, et al. "A blind deblurring and image decomposition approach for astronomical image restoration." 2015 23rd European Signal Processing Conference (EUSIPCO). IEEE, 2015.

[5] Liu, Peiyu, et al. "Blind deblurring using space target features." IEEE Access 7 (2019): 131818-131829.

[6] Zhang, Cundu, et al. "Fast restoration of star image under dynamic conditions via lp regularized intensity prior." Aerospace Science and Technology 61 (2017): 29-34.

[7] Wang, Shiqiang, et al. "Motion blurred star image restoration based on MEMS gyroscope aid and blur kernel correction." Sensors 18.8 (2018): 2662.

[8] Whisler, David and Adam Dai. “Computational Star-Tracking Compensation in Astrophotography.” (2020).

[9] Hou, Yaxian, et al. "A real-time star tailing removal method based on fast blur kernel estimations." Mathematical Problems in Engineering 2021 (2021): 1-13.