Sentinel-2 Sharpening Using a Reduced-Rank Method With Modified Roughness Regularization

Final Report

A. Kikkeri, H. Do, J. Gearig, K. Rauscher

December 27, 2021

Our team will give attribution for any figures used in our documents and will cite all code sources (beyond those already built into Julia Base or Matlab toolboxes)

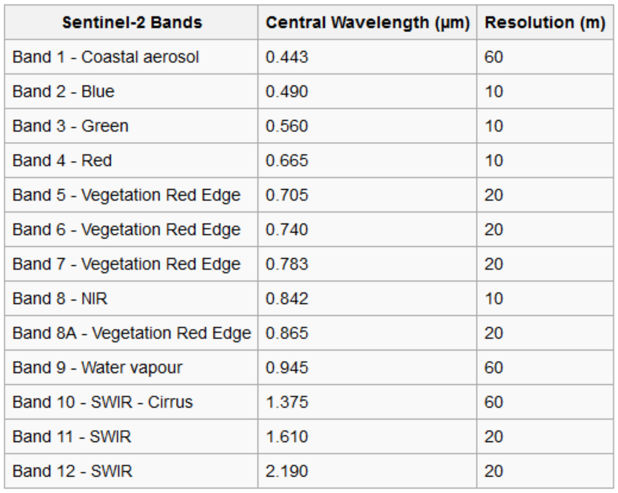
1 Abstract

Pansharpening, the process of improving resolution of some bands of an image through information from higher-resolution bands, is important for high-fidelity satellite images. In the following paper, we propose improvements to S2Sharp, an optimization based sharpening method similar to pansharpening, which sharpens low-resolution bands of images collected from the Sentinel-2 Satellite. We observe modest improvements in error metrics, partially convert S2Sharp from Matlab to Julia, and mathematically describe possible changes to the regularization used for our cost function.

2 Introduction

Optical satellite imagery of the earth is typically collected in multiple bands, where each band is an image containing measurements from a certain visible/near-visible region of the electromagnetic spectrum. Because of physical limitations of sensors, the spatial resolution of images of each band generally differs. Bands with larger spectral bandwidth have finer spatial resolution than bands with smaller spectral bandwidth. This results in a trade-off between spatial resolution and spectral resolution. It is desirable to combine the spatial resolution of wide-spectrum bands with the spectral resolution of narrow-spectrum bands. There are various sharpening techniques that carry out this super-resolution task.

The primary super-resolution technique in the literature is known as *pansharpening*. It is generally viewed as a task of fusing data between two images. One image, the *panspectral* (PAN) image, contains light captured from the entire visible portion of the electromagnetic spectrum, hence the prefix *pan*. The other image is known as a *multispectral* (MS) image, and it contains multiple bands comprising smaller portions of the light spectrum (typically red, green, and blue bands). The PAN image has finer spatial resolution



Sentinel-2 Bands	Central Wavelength (µm)	Resolution (m)
Band 1 - Coastal aerosol	0.443	60
Band 2 - Blue	0.490	10
Band 3 - Green	0.560	10
Band 4 - Red	0.665	10
Band 5 - Vegetation Red Edge	0.705	20
Band 6 - Vegetation Red Edge	0.740	20
Band 7 - Vegetation Red Edge	0.783	20
Band 8 - NIR	0.842	10
Band 8A - Vegetation Red Edge	0.865	20
Band 9 - Water vapour	0.945	60
Band 10 - SWIR - Cirrus	1.375	60
Band 11 - SWIR	1.610	20
Band 12 - SWIR	2.190	20

Table 1: Spatial Resolution of Sentinel-2 Bands [5]

than the MS image. The pansharpening process fuses the high spatial resolution of the PAN image with the high spectral resolution of the MS image, producing a high resolution colored product. Pansharpening is an established technique for processing satellite imagery and has been explored very frequently over the last 30 years. There are multiple works devoted to surveying the topic, including a textbook [1] and many papers, such as [2] [3] and [4].

Although many satellites produce imagery that neatly fits into pansharpening algorithms, some have more variety in the bands they produce. One prominent example of this is the Sentinel-2 satellite. Sentinel-2 has 13 bands total, and four bands at its sharpest resolution level of 10 meters: blue, green, red, and near-infrared [5]. Details of spectral wavelength and spatial resolution of Sentinel-2 bands are shown in Table 1. Though sharpening Sentinel-2 data is not able to improve the resolution of RGB image outputs, it can use the 10m bands to sharpen the other bands, facilitating exploitation for applications requiring spectral information beyond visible light.

Some authors have designed methods specifically focused on Sentinel-2 imagery. [6] designs a framework using the 10m bands to sharpen the other low-resolution bands of varying resolution, and this method is reproduced for typical pansharpening in [4]. We worked to reproduce [6] using Sentinel-2 imagery, and explore the idea of expanding the method to WorldView-3 imagery, which has higher spatial resolution and fewer bands than Sentinel-2 [7]. Additionally, we converted the currently available MATLAB code [8] into Julia, and evaluated changes in speed.

After reproducing the results, we investigated methods to improve the sharpening quality of the current algorithm. The original paper, [6], formulates the optimization problem to be solved using a cost function, so our group modified the cost function to improve error metrics. A regularization term in the cost function attempts to smooth out uniform areas while maintaining structure in presence of edges, and we explored techniques to better detect and capture the roughness information in uniform areas and/or edges.

3 Quantitative Performance Prediction

We have assessed the quantitative performance of our pansharpening algorithm as follows. Our implementation of the algorithm (including any extensions) was compared to the original implementation to test the effect of our extensions.

One data set we used for testing is Sentinel-2 data. Sentinel-2 data is publicly available on Copernicus Open Access Hub [9]. Sentinel 2 band and resolution info is shown in Table 1. We also attempted to extend the method to WorldView-3 data, which is available as a limited set of sample products [10]. WorldView-3 data has 8 spectral bands, with PAN resolution of 30cm and MS resolution of 1.2m [7].

When using Sentinel-2 data, the original image served as the reference image when calculating performance metrics; the algorithm instead ran on a downgraded version of the image with lower resolution and attempt to recreate the original image. As per the original paper [6], when using non Sentinel-2 data, in this case from WorldView-3, we applied the S2 spectral response and point spread function to better simulate S2 data; this modified image will then be used as a reference. Similarly as before, this new reference image was downgraded for the purpose of testing our pansharpening algorithm.

The quantitative metrics for comparison are given from the original paper [6]:

- spectral angle mapper (SAM)
- signal-to-reconstruction error (SRE)
- root mean squared error (RMSE)
- structural similarity (SSIM).

SAM measures the similarity between two spectral by calculating the angle between them:

$$\text{SAM} = \frac{1}{n}\sum_{p=1}^{n} \arccos\left(\frac{\mathbf{x}_{(p)}^T \hat{\mathbf{x}}_{(p)}}{\|\mathbf{x}_{(p)}\|\|\hat{\mathbf{x}}_{(p)}\|}\right)\frac{180}{\pi}. \tag{1}$$

SRE (in dB) measure the error between the reconstructed and reference image at a given band i:

$$\text{SRE}_i = 10\log_{10}\left(\frac{\|\mathbf{x}_i\|^2}{\|\mathbf{x}_i - \hat{\mathbf{x}}_i\|^2}\right). \tag{2}$$

RMSE is the average root squared error over all the bands given in the image:

$$\text{RMSE} = \frac{1}{n}\sum_{i=1}^{L} \|\mathbf{x}_i - \hat{\mathbf{x}}_i\|. \tag{3}$$

NRMSE is similar to RMSE but normalized to the pixel values of the reference image:

$$\text{NRMSE} = \frac{\sum_{i=1}^{L} \|\mathbf{x}_i - \hat{\mathbf{x}}_i\|}{\sum_{i=1}^{L} \|\mathbf{x}_i\|}. \tag{4}$$

The SSIM metric for one band i and one pixel j is

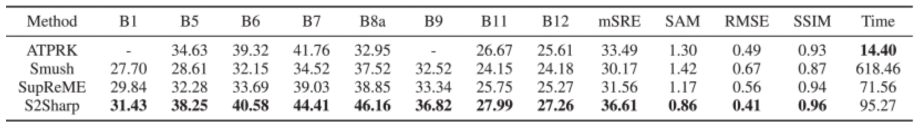
$$\text{SSIM}_{i,j} = \frac{(2\mu_{x_{i,j}}\mu_{\hat{x}_{i,j}} + c_1)(2\sigma_{x_{i,j}\hat{x}_{i,j}} + c_2)}{(\mu^2_{x_{i,j}} + \mu^2_{\hat{x}_{i,j}} + c_1)(\sigma^2_{x_{i,j}} + \sigma^2_{\hat{x}_{i,j}} + c_2)}. \tag{5}$$

As explained in [6], $\mu_{x_{i,j}}$ and $\sigma_{x_{i,j}}$ represent the mean and standard deviation of the original image, and $\mu_{\hat{x}_{i,j}}$ and $\sigma_{x_{i,j}\hat{x}_{i,j}}$ represent the mean and standard deviation of the sharpened image, both at band i and at a window around pixel j. c_1 and c_2 are constants based on the dynamical range of the image. We average over all bands and pixels to get the final SSIM metric.

Finally, our project involves converting the existing MATLAB code [8] to Julia. We also compare our runtime against the original MATLAB implementation.

3.1 Performance Prediction

With all of the quantitative metric models above, in [6] the authors have created a table comparing the results with other state-of-the-art methods, like SupReME, Smush and ATPRK. The table is shown below.



Method	B1	B5	B6	B7	B8a	B9	B11	B12	mSRE	SAM	RMSE	SSIM	Time
ATPRK	-	34.63	39.32	41.76	32.95	-	26.67	25.61	33.49	1.30	0.49	0.93	**14.40**
Smush	27.70	28.61	32.15	34.52	37.52	32.52	24.15	24.18	30.17	1.42	0.67	0.87	618.46
SupReME	29.84	32.28	33.69	39.03	38.85	33.34	25.75	25.27	31.56	1.17	0.56	0.94	71.56
S2Sharp	**31.43**	**38.25**	**40.58**	**44.41**	**46.16**	**36.82**	**27.99**	**27.26**	**36.61**	**0.86**	**0.41**	**0.96**	95.27

Figure 1: Evaluation for the ARVIS S2 simulation. Best results shown in bold [6]

Since we were proposing modifications to an existing method, we expected our improvements to be modest. Additionally, since we converted sections of code to Julia, we expect performance improvements from the original Matlab implementation in [8].

4 Methods

A variety of pansharpening methods exist, requiring different underlying technology. Some involve spatial frequency transforms, some involve formulated optimization problems, and others use deep learning. We have decided to focus on the best performing optimization paper (as evaluated in source [4]) as the base of our project. This section discusses the method from the paper, and outlines the steps we will take to attempt to improve it.

4.1 Summary of Sentinel-2 Reduced Rank Sharpening Method

Source [6] proposes a novel optimization method to produce pansharpened images. The algorithm, S2Sharp, finds a low-rank approximation to a set of aligned images of different resolutions, refining the approximation using a cyclic descent method. MATLAB code is provided as reference [8].

4.2 Forward Blur Model

Before showing the full optimization model, there is preliminary terminology to discuss, starting with the Sensor Model [6]. $\boldsymbol{B}_i$ is a circulant blur matrix, which the spatial point spread function known for each S2 Band is used to construct. $\boldsymbol{M}_i$ is the downsampling matrix. $\boldsymbol{\epsilon}_i$ is the sensor noise. $\boldsymbol{y}_i$ refers to the noisy vectorized images, $\boldsymbol{x}_i$ refers to the vectorized target images, and L is the number of images to sharpen.

$$\boldsymbol{y}_i = \boldsymbol{M}_i \boldsymbol{B}_i \mathbf{x}_i + \boldsymbol{\epsilon}_i, \quad i = 1, 2, \ldots, L \tag{6}$$

For the reduced-rank approximation, using the L vectorized images, we can create an $n \times L$ matrix $\boldsymbol{X}$, which is assumed to be represented in r-dimensional space where $r \leq \min(n, L)$. This assumption implies there is a matrix factorization $\boldsymbol{X} = \boldsymbol{G}\boldsymbol{F}^T$ where $\boldsymbol{G}$ is a full rank $n \times r$ matrix and $\boldsymbol{F}$ is a full rank $L \times r$ matrix with orthonormal columns that spans r-dimensional subspace in $\mathbb{R}^L$ [6].

4.3 Sentinel-2 Reduced Rank Sharpening Method

For the estimation of the sharpened image, we hope to minimize the cost function

$$J(\boldsymbol{F}, \boldsymbol{G}) = \sum_{i=1}^{L} \frac{1}{2} \|\boldsymbol{y}_i - \boldsymbol{M}_i \boldsymbol{B}_i \boldsymbol{G} \boldsymbol{f}_{(i)}\|^2 \ + \sum_{j=1}^{r} \lambda_j \phi_w(\boldsymbol{g}_j), \tag{7}$$

where the first term is the l_2 norm between the observation and the approximation, $\boldsymbol{y}_i$ and $\boldsymbol{x}_i$ respectively, and the second term is aimed to impose *a priori* structure where $\lambda_j > 0, j = 1, ..., r$ are weights [6]. $\boldsymbol{g}_j$ is defined as each column vector in $\boldsymbol{G}$, and $\boldsymbol{f}_{(i)}$ are columns of $\boldsymbol{F}^T$.

4.3.1 Regularization Term

Expanding on the weighted roughness of the optimization function, the weighted roughness term $\phi_w(g_j)$ is defined to be

$$\phi_w(\boldsymbol{g}_j) = \boldsymbol{g}_j^T \left(\boldsymbol{H}_h^T \boldsymbol{W} \boldsymbol{H}_h + \boldsymbol{H}_v^T \boldsymbol{W} \boldsymbol{H}_v \right) \boldsymbol{g}_j, \tag{8}$$

where $\boldsymbol{H}_v$ and $\boldsymbol{H}_h$ matrices are the vertical and horizontal $n \times n$ finite difference matrices and $\boldsymbol{W} = [w_{ij}]$ is a diagonal spatial weighting matrix. The idea behind $\boldsymbol{W}$ is that without it, the difference matrices will blur the edges, which is a useful part in pansharpening. This is why the $\boldsymbol{W}$ matrix is required to mitigate this effect and down-weight pixels at edges relative to other pixels. The values of this weight matrix are calculated by the following equation:

$$\max(0.5, e^{z_{max}^2 / 2\sigma_s^2}). \tag{9}$$

where z_{max} is the maximum gradient at that pixel over all the high-resolution bands and σ_s^2 is the scaling factor [6].

4.3.2 Optimization Method

The next step is the Cyclic Descent, which solves a non-convex optimization problem without a closed-form solution. The initial solution is calculated by the compact SVD of the interpolated bands, considering the r largest singular values [6].

The minimization problem is separated into two steps: G-Step and the F-Step. These two steps are where one of the values is fixed and estimating the other value by solving the minimization function. Because these are not easily solved in Euclidean space, a transformation to Stiefel manifold allows the problem to be a relatively small quadratic optimization problem. This problem [6] is

$$\phi_f(\boldsymbol{F}) = \sum_{i=1}^{L} \frac{1}{2} \boldsymbol{f}_{(i)}^T \boldsymbol{K}_{(i)} \boldsymbol{f}_{(i)} - \boldsymbol{z}_i^T \boldsymbol{f}_{(i)}. \tag{10}$$

The variables K_i and z_i are defined as:

$$\begin{aligned} \boldsymbol{K}_i &= (\boldsymbol{G}^{k+1})^T \boldsymbol{B}_i^T \boldsymbol{M}_{(i)}^T \boldsymbol{M}_{(i)} \boldsymbol{B}_{(i)} \boldsymbol{G}^{k+1} \\ \boldsymbol{z}_i &= (\boldsymbol{G}^{k+1})^T \boldsymbol{B}_{(i)}^T \boldsymbol{M}_{(i)}^T \boldsymbol{y}_i. \end{aligned}$$

4.4 Implementation

4.4.1 Code Implementation

Algorithm wise, the team will use the code written in MATLAB and understand the process using Algorithm 1 of S2Sharp to construct a Julia version. This will allow reproduction of the outputs of the original paper.

As the team chose to develop the Sentinel-2 Sharpening Using a Reduced-Rank (S2 sharpening) method as the base algorithm with different optimizations, looking at different loss functions to either reduce the cost while maintaining accuracy or increasing the accuracy with similar computational cost, we found the MATLAB code for the S2 sharpening, released by the authors [8]. One of the main tasks will be to understand the manopt, a MATLAB toolbox for optimization on manifolds, and integrate to Julia language: `https://github.com/mou12/S2Sharp/tree/master/manopt` [11]. This is also relatively simple because the library is available in Julia as well : `https://www.manopt.org` [12].

Using the above library and the methods introduced in the papers, the team will reconstruct the base sensor model and the reduced rank approximation, where the input image is assumed to be perturbed by a circulant blur matrix, and downsampled, with noise added during the sensing procedure.

4.4.2 Data Sets

The primary data set the team will use is the Sentinel-2 data set. There are couple of benefits using this data set. One of them is that it will easily be comparable to the author's original result. Another benefit is that the Sentinel-2 data set is co-registered so there is no need for pre-processing the input data. The data is available at: `https://scihub.copernicus.eu` [9], where there are tens of thousands of images available for use. Another source of Sentinel-2 and other satellite images are `https://earthdata.nasa.gov` [13], where there are thousands of images available for use. For the WorldView-3 data, we plan to use approximately 5 sets of images available as samples from Maxar [10]. WorldView-3 pan and MS data is also co-registered.

4.4.3 Dataset

We will use data from the Copernicus Open Access Hub [9] as our primary data set. From this resource, we have first started with a Sentinel-2 Level-1C dataset of Ann Arbor taken on September 26, 2020 with (Fig. 6). This contains the typical 13 spectral bands of Sentinel-2 imagery (of which we used all except 10), and the image subset we selected is 600×600 pixels at 10m resolution. The images are stored and downloaded in JPEG 2000 format.

To test our Sentinel-2 data, we downsampled the 10m and 20m bands to have 20m and 40m resolution. The pansharpening process will then increase the resolution of all the images to 20m, and the original 20m bands will serve as the reference images. The downside of this approach is that the 60m bands are not used in the final analysis. [14] proposes the following method to generate our test images: we first applied a Gaussian filter, then moving average, and then downsampled by a factor of 2. (Fig. 2).

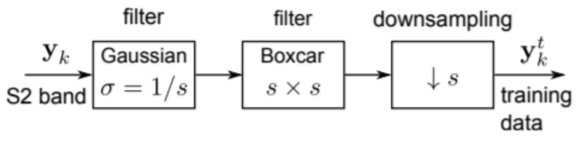


Figure 2: Flowchart describing the process of downsampling data for testing data (not training data). For us, $s = 2$.

We also have a second dataset included in the example code. It is an AVIRIS image taken over Sacramento, California from August 2013, with the spectral response of S2 imagery applied. The image is 408×408 pixels, and contains all S2 bands except for 10. This dataset includes both the degraded test data and the optimal true data for comparison ([6]).

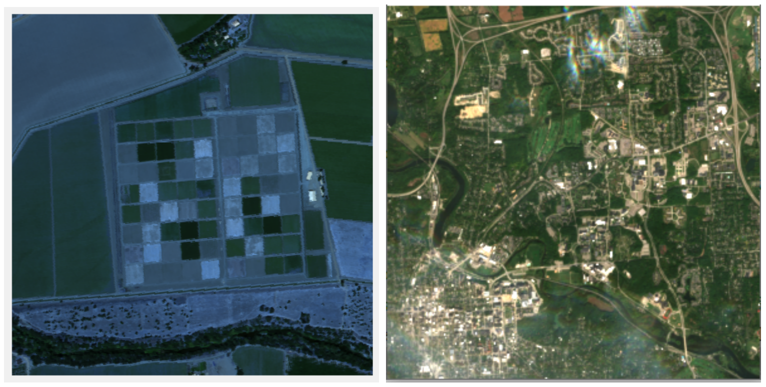


Figure 3: On the left, the AVIRIS dataset, where bands 2-4 represent the channels B,G, R respectively. On the right, the Sentinel-2 dataset as displayed on the Copernicus Open Access Hub.

4.5 Modifications to Original Algorithm

We attempted several modifications to improve the pansharpening accuracy of S2Sharp. The following sections give mathematical details of those modifications.

4.5.1 Reformulation of Optimization Problem

As mentioned by Professor Fessler, the team investigated to change the conversion of updating the $\boldsymbol{F}$ as an orthonormal Procrustes problem rather than converting the equation to a Stiefel manifold, which requires the Manopt library and would have higher computational cost. The idea of converting $\boldsymbol{F}$ as a orthonormal Procrustes problem follows these steps. First, the to make the $\boldsymbol{F}$ the only variable, fix $\boldsymbol{G}$ first by conducting the G-step. Then using the property that all norms are equivalent within constant factors, the l_2 norm can be converted as a Frobenius norm. If there existed a matrix M and B that stays constant throughout all frequencies, when capturing the images, then this would have been possible to be converted to a Procrustes problem. If this was the case the formulation of the optimization problem would change using (7) to remove the summation and change the column vector f_i to the matrix $\boldsymbol{F}$. Then, realizing that this is essentially minimizing the (7) with respect to $\boldsymbol{F}$,

$$\hat{\boldsymbol{F}} = \min_{F,G=G^{k+1}} J(\boldsymbol{F},\boldsymbol{G}) = \operatorname*{argmin}_{F,G=G^{k+1}} \|\boldsymbol{Y} - \boldsymbol{W}\boldsymbol{F}\|_F^2 \; + \sum_{j=1}^{r} \lambda_j \phi_w(\boldsymbol{g}_j), \tag{11}$$

where $\boldsymbol{W} = \boldsymbol{M} * \boldsymbol{B} * \boldsymbol{G}^{k+1}$. However, in the formulation of the sentinel-2 data, we see that the $\boldsymbol{M}$ and $\boldsymbol{B}$ matrices are different for each bands, with different down-sampling factors. Some frequencies need factor of 6 while others need only factor of 2. In the case where we consider all of the input data to have the same resolution and parameters, so the $\boldsymbol{M}$ and $\boldsymbol{B}$ matrices are the same for every band, we can formulate as a procrustes problem by combining it to $\boldsymbol{W}$ shown in eq 11. If this was the case, noting that the orthonormal Procrustes problem is scale invariant, we would change the norm to Frobenius norm. This means that whatever scale is applied between the output and input values, the solution $\hat{\boldsymbol{F}}$ will be the same. We would like to change it to the Frobenius norm to compute the equation in a simpler fashion using the trace properties of the Frobenius norm. Solving for the $\boldsymbol{F}$ that will minimize the $\boldsymbol{J}$, we can represent it as the following equation:

$$\hat{\boldsymbol{F}} = \operatorname*{argmin}_{\boldsymbol{F}}(\text{trace}\{(\boldsymbol{Y} - \boldsymbol{W}\boldsymbol{F})'(\boldsymbol{Y} - \boldsymbol{W}\boldsymbol{F})\} = -2\,\text{real}(\text{trace}\{\boldsymbol{C}\boldsymbol{F}\})), \tag{12}$$

where $\boldsymbol{C} = \boldsymbol{W}'\boldsymbol{Y}$ and the SVD of it is $\boldsymbol{C} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}'$. Then the optimal $\hat{\boldsymbol{F}}$ will be $\hat{\boldsymbol{F}} = \boldsymbol{V}\boldsymbol{U}'$, so the $\boldsymbol{F}$-update could be re-written as

$$\boldsymbol{F}^{k+1} = \boldsymbol{V}\boldsymbol{U}', \tag{13}$$

where $\boldsymbol{C} = (\boldsymbol{M}\boldsymbol{B}\boldsymbol{G}^{k+1})'\boldsymbol{Y})$, and $\boldsymbol{U}$ and V are from the SVD of $\boldsymbol{C}$; i.e., $\boldsymbol{C} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}'$

4.5.2 Cost Function Changes

We have investigated two modifications to the original cost function. The first modification involves changes in image gradient calculation as the $\boldsymbol{W}$ matrix is formed, and the second involves modifying the roughness regularization term (8) to incorporate a nonquadratic edge-preserving potential. We implemented both modifications as changes to the publicly provided MATLAB code, and measured the sharpening accuracy of the algorithms using established metrics.

4.5.2.1 Investigation of Total Variation We have also looked into different regularization terms from different papers, mainly [15], to see whether different regularization term could improve the results. In [6], the reason of having the regularization term is to impose a *priori* structure on the estimated image by considering the roughness of the components to stop possible blurring of the edges. Knowing this the TV regularization seemed to be a reasonable replacement because it aims to encourage noise removal while preserving edges[15]. By constructing smooth estimations without penalizing the sharp boundaries the total variance method which is formulated below in equation(15) seemed very reasonable. With the cost function using total variance formulated as

$$J(\mathbf{x}) = \|\mathbf{y} - \mathbf{M}\mathbf{x}\|_2^2 + \lambda\,\mathrm{TV}(\mathbf{x}) \tag{14}$$

where TV(x) is defined as the following

$$\mathrm{TV}(\mathbf{x}) = \left\| \sqrt{(\mathbf{D}_H\mathbf{x})^2 + (\mathbf{D}_V\mathbf{x})^2} \right\|_1, \tag{15}$$

and the input x is the vectorized MS images and $D_H = I_l \otimes D_h$, with l being the number of MS images, and the D_h returns a first order difference in the horizontal and vertical direction when multiplied with a vectorized image. By taking different materials from the reduced rank method, a reformulation of the regularization could be conducted. First, D_h and D_v can be replaced with H_h and H_v. Another component that could be replaced is the λ in the total variation cost function. A good substitution would be the weights, λ_j, diagonalized. This is because both terms exist to scale the roughness components to estimate the reconstructed image. While the reduced rank method does not specify the calculation of the weighted terms, the total variance method calculates the weights using the scaled inverse total variance of each band. However, the reduced rank paper does go deeper with the calculation of weights with $\boldsymbol{W}$ by calculating the weights for each pixel using the maximum gradient over all high-resolution bands.

In the actual calculation, in the G-step of the update, the regularization can be viewed as stacking the horizontal and vertical components and conducting an inner product with the diagonalized λ weights. This is very similar to the total variation where $D_H * x$ and $D_v * x$ will return the gradients of the image. Conducting the rest of the operations, the sum of the magnitude of the gradient, with respect to xy-plane, is calculated to be the total variation.

While the regularization term was not changed in the algorithm due to complex reformulation of the update steps, we saw that the change of regularization from reduced rank to total variation is reasonable and quite similar.

4.5.2.2 Modification of W Formulation The $\boldsymbol{W}$ matrix is originally formed using (9). z_{max} is defined as the pixel-wise maximum of the gradient over all bands. Expressed using indices i and j of 2D images, it is formulated as:

$$z_{max}[i,j] = \max_{k\in\{1,\dots,K\}} \sqrt{((\boldsymbol{Y}_k ** \boldsymbol{h}_v)[i,j])^2 + ((\boldsymbol{Y}_k ** \boldsymbol{h}_h)[i,j])^2}, \tag{16}$$

where $\boldsymbol{Y}_k$ is each image band at full-resolution and $\boldsymbol{h}_h$ and $\boldsymbol{h}_v$ are horizontal and vertical difference kernels, respectively. The difference kernels used in the original implementation are the following:

$$\boldsymbol{h}_h = \begin{bmatrix} \underline{1} & -1 \end{bmatrix} \tag{17}$$

$$\boldsymbol{h}_v = \begin{bmatrix} \underline{1} \\ -1 \end{bmatrix} \tag{18}$$

The initial modifications which we have made involve replacing the finite difference kernels with Sobel and Prewitt kernels which are frequently used for simple edge detection operations.

The Prewitt kernels are defined as follows [16]:

$$\boldsymbol{h}_h = \begin{bmatrix} 1 & 0 & -1 \\ 1 & \underline{0} & -1 \\ 1 & 0 & -1 \end{bmatrix} \tag{19}$$

$$\boldsymbol{h}_v = \begin{bmatrix} 1 & 1 & 1 \\ 0 & \underline{0} & 0 \\ -1 & -1 & -1 \end{bmatrix} \tag{20}$$

The Sobel kernels are defined as follows [16]:

$$\boldsymbol{h}_h = \begin{bmatrix} 1 & 0 & -1 \\ 2 & \underline{0} & -2 \\ 1 & 0 & -1 \end{bmatrix} \tag{21}$$

$$\boldsymbol{h}_v = \begin{bmatrix} 1 & 2 & 1 \\ 0 & \underline{0} & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{22}$$

In the code implementation, the kernels were normalized to have responses of similar magnitude. This was accomplished by dividing the Prewitt kernels by 3 and the Sobel kernels by 4.

4.5.2.3 Replacement of Finite Difference Matrices in Regularization Term

Early in the project, we attempted to replace the finite difference matrices $\boldsymbol{H}_h$ and $\boldsymbol{H}_v$ in (8) with matrices applying the Sobel and Prewitt kernels to a vectorized image. This made all of the metrics much worse, as shown in the later results section, so we abandoned this idea early on.

4.5.2.4 Edge Preserving Potential Term

The regularization term as originally expressed in the paper, restating (8), is:

$$\phi_w(\boldsymbol{g}_j) = \boldsymbol{g}_j^T \left(\boldsymbol{H}_h^T \boldsymbol{W} \boldsymbol{H}_h + \boldsymbol{H}_v^T \boldsymbol{W} \boldsymbol{H}_v \right) \boldsymbol{g}_j, \tag{23}$$

and can be rewritten using the Frobenius norm:

$$\phi_w(\boldsymbol{g}) = ||\boldsymbol{W}^{1/2} \boldsymbol{H}_h \boldsymbol{g}_j||_F^2 + ||\boldsymbol{W}^{1/2} \boldsymbol{H}_v \boldsymbol{g}_j||_F^2. \tag{24}$$

We define the k-th vectorized image band as $\boldsymbol{\gamma} \triangleq \boldsymbol{g}_k$ for simplicity of indexing.

The vertical portion of the regularization is equivalent (except for boundary conditions) to the following sum of pixelwise operations applied to the (column-major) vectorized image:

$$\sum_{i=2}^{n} w_i(|\gamma_i - \gamma_{i-1}|^2), \tag{25}$$

and the horizontal term can be written similarly using strided indexing.

This can be rewritten using the quadratic potential function $\psi(x) = |x|^2$,

$$\sum_{i=2}^{n} w_i \psi(\gamma_i - \gamma_{i-1}). \tag{26}$$

which is the same formulation (again, except for boundary conditions) as 1.10.17 in [17], a spatially weighted penalty function part of a discussion on edge-preserving roughness penalties. [17] mentions that the quadratic penalty function tends to blur edges. Putting the cost function in the formulation of (26) means that we can replace the current quadratic penalty function with an edge-preserving penalty function. [17] contains a compilation of several alternative penalty functions, and we looked into and tested these alternatives.

Modifying the regularization term requires deriving a gradient of the modified term, because the optimization method used in S2Sharp requires taking the gradient of $\boldsymbol{J}(\boldsymbol{F}, \boldsymbol{G})$ with respect to G. With the original quadratic regularization term, computing the gradient is simple, and is implemented in the provided MATLAB code as:

$$\frac{\partial \phi_w(\boldsymbol{g})}{\partial \boldsymbol{g}} = (\boldsymbol{H}_h^T \boldsymbol{W} \boldsymbol{H}_h + \boldsymbol{H}_v^T \boldsymbol{W} \boldsymbol{H}_v)\boldsymbol{g}. \tag{27}$$

This is equivalent to the following formula using the derivative of the quadratic potential applied element-wise:

$$\frac{\partial \phi_w(\boldsymbol{g})}{\partial \boldsymbol{g}} = \boldsymbol{H}_h^T \boldsymbol{W}^{1/2} \dot{\psi}_q(\boldsymbol{W}^{1/2} \boldsymbol{H}_h \boldsymbol{g}) + \boldsymbol{H}_v^T \boldsymbol{W}^{1/2} \dot{\psi}_q(\boldsymbol{W}^{1/2} \boldsymbol{H}_v \boldsymbol{g});\ \dot{\psi}_q(\boldsymbol{x}) = \boldsymbol{x}, \tag{28}$$

where, generally, the gradient of a potential used in this term is

$$\frac{\partial \psi(\boldsymbol{W}^{1/2} \boldsymbol{H} \boldsymbol{g})}{\partial \boldsymbol{g}} = \boldsymbol{H}^T \boldsymbol{W}^{1/2} \dot{\psi}(\boldsymbol{W}^{1/2} \boldsymbol{H} \boldsymbol{g}). \tag{29}$$

Thus, any differentiable potential can be swapped in for the quadratic potential and will still fit the mathematical framework of this problem. The required modification to the code only involves separating the multiplications in the regularization term, which is simple because each of the matrix multiplications (implemented as convolutions) is implemented separately, and taking the matrix square root of $\boldsymbol{W}$, which is trivial because it is a diagonal matrix.

We replaced the quadratic potential with multiple different edge-preserving potentials with different parameters, and the results of these modifications are shown in the next section.

4.6 Results

This section contains quantitative evaluations of the pansharpening accuracy of our method, as compared to the original S2Sharp method. It also shows qualitative visual results of the sharpening performance on both Sentinel-2 and WorldView-3 data. Finally, this section goes over the quantitative results and challenges of converting the code from MATLAB to Julia.

Method	SAM	Avg. SRE	aSSIM	RMSE
Original	0.8834	36.594	0.9582	0.4145
Prewitt	0.8615	36.754	0.9589	0.4013
Sobel	**0.8609**	**36.760**	**0.9589**	**0.4009**

Table 2: Results with new kernels in W only

Method	SAM	Avg. SRE	aSSIM	RMSE
Original	**0.8834**	**36.594**	**0.9582**	**0.4145**
Prewitt	0.9600	35.746	0.9542	0.4556
Sobel	1.0534	34.387	0.9384	0.5175

Table 3: Results with new kernels in weight function only

4.6.1 Results of Initial Modifications

Tables 2, 3 and 5 show our initial results from sharpening the AVIRIS data used in [6]. We replaced the kernels independently in two places: the formation of the $\boldsymbol{W}$ matrix, and the convolutions of $\boldsymbol{H}_h$ and $\boldsymbol{H}_v$ with the images $\boldsymbol{g}_j$ in the weighting function. Table 2 shows the results with only the first replacement, Table 3 only the second, and Table 5 both replacements. We have highlighted the most desirable results for each metric.

The results show that replacing the kernel used to calculate the image gradient magnitude seems to be beneficial, but extending that to the later roughness calculations makes the results worse. In every metric for the $\boldsymbol{W}$-only case, Sobel performs the best, followed by Prewitt and then the original. For the cases with all kernels or only weight function kernels replaced, the order is reversed, with the original version performing the best. The replacement in the weight function negates any benefit introduced by the replacement in the creation of $\boldsymbol{W}$. This shows that the two replacements can be performed independently, and that there is no need for the kernels in both steps to be the same. Because of the poor performance of replacing $\boldsymbol{H}_h$ and $\boldsymbol{H}_v$, we abandoned that idea in further testing.

4.6.2 Results of Final Modifications

While doing the initial modifications, we found that the Sobel kernel was clearly superior as an edge detection kernel for forming $\boldsymbol{W}$, and that modifying the finite difference kernels used in the regularization term made all of the metrics worse. For the next modifications, we only modify $\boldsymbol{W}$, and we restrict ourselves to the original $\boldsymbol{W}$, the Sobel method, or no weighting.

The final modifications were replacement of the quadratic potential function with edge-preserving potentials,

Method	SAM	Avg. SRE	aSSIM	RMSE
Original	**0.8834**	**36.594**	**0.9582**	**0.4145**
Prewitt	0.9381	35.936	0.9550	0.4418
Sobel	1.0394	34.462	0.9388	0.5093

Table 4: Results with new kernels in all usages

Method	SAM	Avg. SRE	aSSIM	RMSE
Original	**0.8834**	**36.594**	**0.9582**	**0.4145**
Prewitt	0.9381	35.936	0.9550	0.4418
Sobel	1.0394	34.462	0.9388	0.5093

Table 5: Results with new kernels in all usages

and they are evaluated in conjunction with modifications to the $\boldsymbol{W}$matrix.

After experimentation with several potential functions in [17], we found that the Cauchy potential with $\delta = 0.5$,

$$\psi(z) = \psi_\delta(z) \triangleq \frac{\delta^2}{2} log(1 + |z/\delta|^2), \tag{30}$$

provided the best metrics on the AVIRIS dataset. Our final results compare the original potential function (quadratic) with this edge-preserving potential. We additionally include three separate methods of calculating weights: no weighting ($\boldsymbol{W} = \boldsymbol{I}$), the original weighting using the "intermediate" kernel, and weighting using the Sobel kernel. These newer results include the additional NRMSE metric to aid in interpretability, although the older results have not been updated to add it. Table 6 shows the results of all of our modifications. Figure 4 shows the resultant image compared to the downsampling done in Figure 2. Figure 5 shows a comparison of our final improved sharpening method to the source [8] method, and a computed difference between the images.

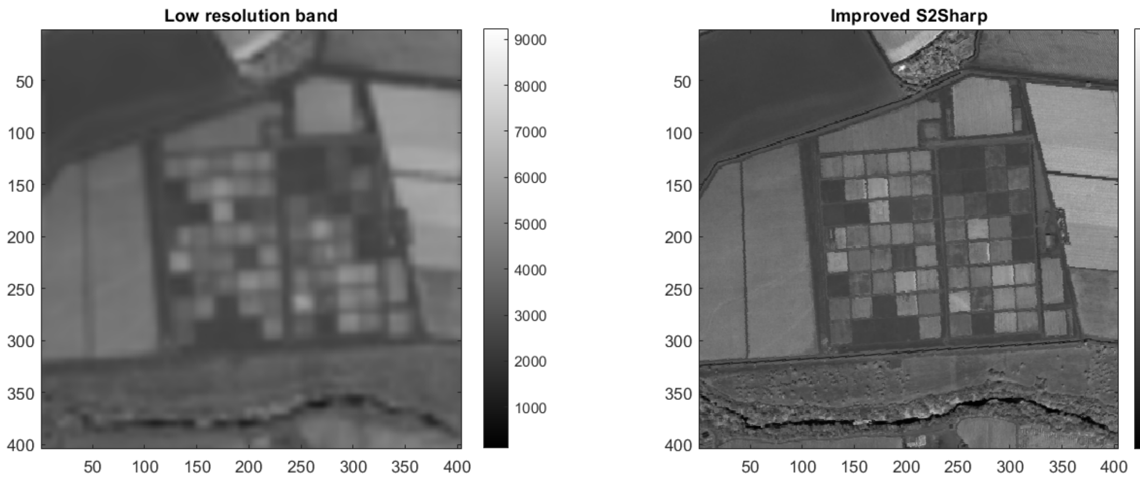


Figure 4: A comparison between the low-resolution band 9 (upsampled using bicubic interpolation), and the pansharpened version.

The result in Band 6, which was pansharpened from 40m resolution to 20m resolution indicates a higher level of detail. Additionally, there was a slight improvement in the error metrics as compared to the original pansharpening proposal.

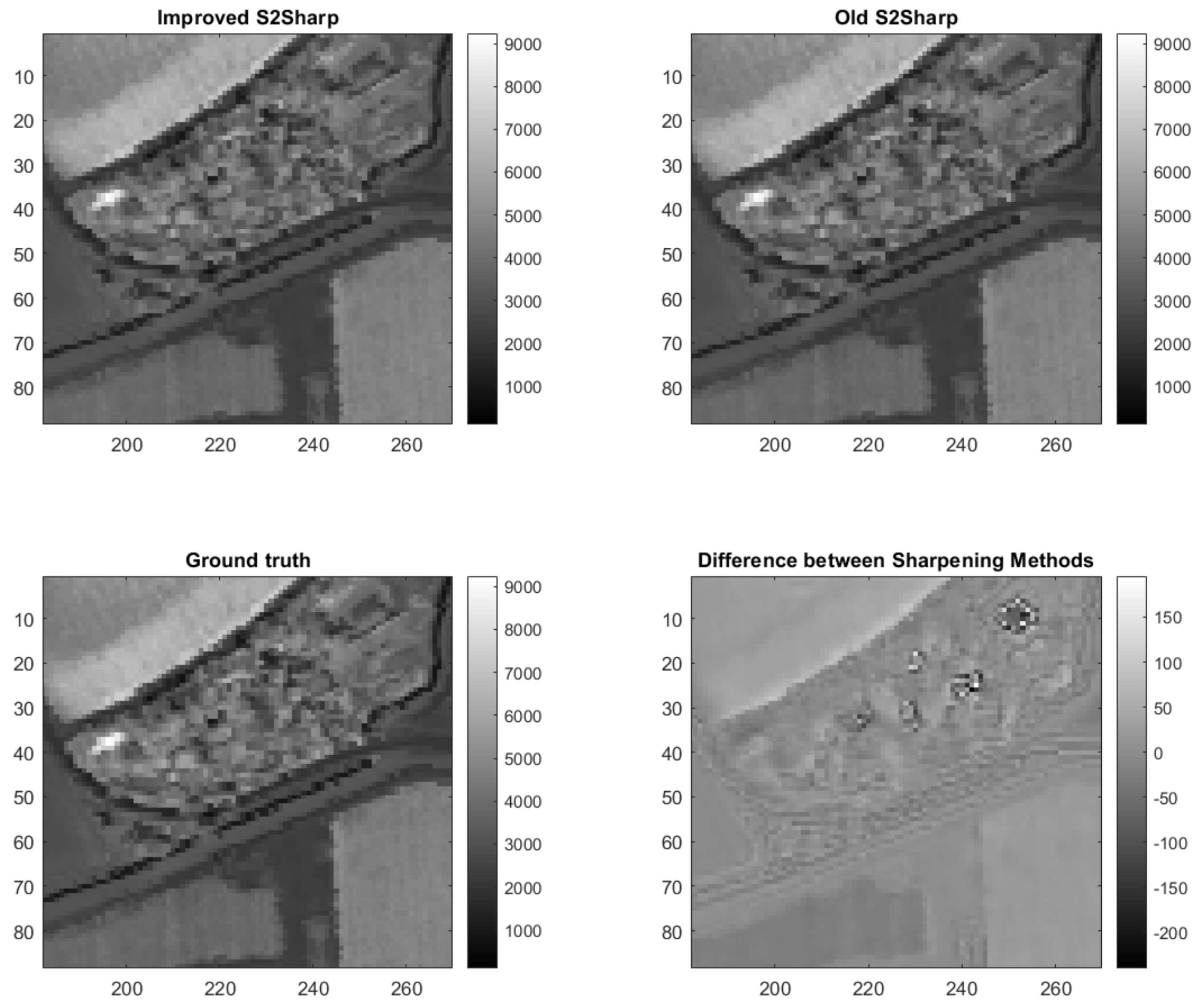


Figure 5: A comparison of our improved S2Sharp with Sobel weighting and Cauchy penalty versus the originally proposed method, showing sharpening differences in a region of band 9 of the AVIRIS dataset

4.6.3 Extension to WorldView Data

The Pansharpening Toolbox [18] includes an adaptation of the S2Sharp code (RRPansharp) to perform pansharpening on 8-band images such as those from WorldView-3. We attempted to add our S2Sharp modifications to RRPansharp, and found that several potential function replacements seemed to cause the gradient descent in the G-step to diverge. Additionally, the result of the original code seemed to have severely distorted colors. Because of a lack of time to dig deeper into these issues, we chose not to go into the WV3 data extension any further.

$\boldsymbol{W}$ Method	Potential	SAM	Avg. SRE	aSSIM	RMSE	NRMSE
None	Quad	0.876	36.626	0.958	0.407	0.0216
Original	Quad	0.883	36.594	0.958	0.414	0.0220
Sobel	Quad	0.860	36.760	0.958	0.401	0.0212
None	Cauchy	0.853	36.764	0.959	0.396	0.0210
Original	Cauchy	0.860	36.733	0.958	0.402	0.0213
Sobel	Cauchy	**0.844**	**36.873**	**0.959**	**0.392**	**0.0208**

Table 6: Results with different Kernels and Potentials on AVIRIS Image

$\boldsymbol{W}$ Method	Potential	SAM	Avg. SRE	aSSIM	RMSE
Original	Quad	2.6413	34.938	0.887	1.3595
Sobel	Cauchy	**2.6377**	**35.13**	**0.891**	**0.1.362**

Table 7: Results on Sentinel-2 image of Ann Arbor before and after improvements

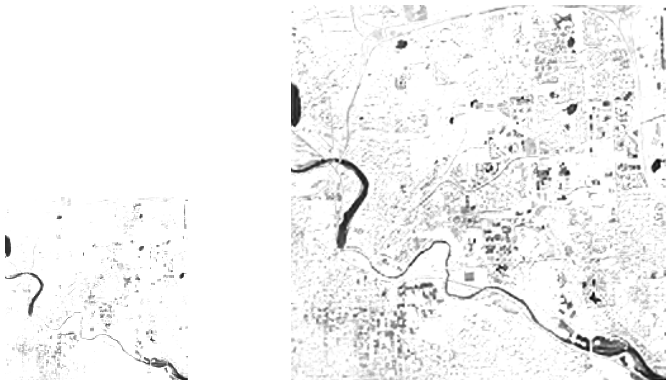


Figure 6: On the left, a grayscale image of Band 6. On the right, the same band after pansharpening. Note the smaller size of the image on the left before pansharpening.

4.6.4 Conversion of Code to Julia

The entire algorithm was not able to be ported to Julia. The Manopt.jl library was missing several functionalities required by the algorithm, notably the parallel vector transport method for the Stiefel manifold, and the ability to specify a gradient in Euclidean space rather than Riemannian space. While it may have been possible to mathematically work around these issues, the team chose to prioritize working on the pansharpening algorithms over this potentially very difficult task. This meant that we could not implement the Fstep portion of the problem in Julia.

To work around the lack of features in Manopt.jl, the team used the MATLAB.jl package to call the MATLAB implementation of the F-step from Julia. This allowed us to ensure that the rest of the Julia code worked properly, and pave the way for a possibly complete Julia implementation.

The Julia implementation was found to be initially much slower than the MATLAB implementation. To account for any overhead in calling MATLAB from Julia, we only discuss the time taken during the Z-Step. Table 8 shows the time taken in processing the 408 x 408 AVIRIS image through the S2Sharp algorithm, using the Sobel gradient and the original roughness potential function. The ”original” Julia version performed

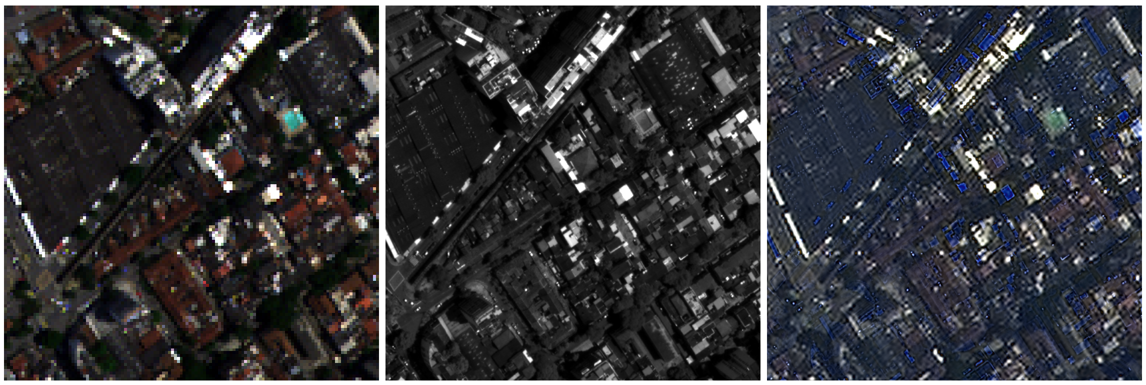


Figure 7: The multispectral, panspectral, and pansharpened WV3 data [10]. The pansharpened data was produced by the original RRPansharp algorithm. The images were displayed using viewing code in the Pansharpening Toolbox, which incorporated linear stretching and fixing saturation of RGB data to produce colors suitable for viewing. [18].

Implementation	Z-Step Time (seconds)
MATLAB	6.674
Julia - Original	38.2
Julia - MT-FFT	14.57
Julia - Optimized	10.034

Table 8: Julia/MATLAB timing

the same operations as the MATLAB, the ”MT-FFT” included telling the FFTW library to run multithreaded (as MATLAB does), and the ”optimized” Julia version replaced some FFT-based image-kernel convolutions with direct versions. For reference, the total time taken by the MATLAB implementation was 20 seconds.

The timing results show that naively copying MATLAB code to Julia can make code run much more slowly. MATLAB uses multiple threads by default. However, for Julia, multithreading manually needs to be enabled from the command line and FFTW multithreading needs to also be enabled by a function call (although ImageFiltering.jl uses threads by default). Even with multithreading enabled, Julia is still than MATLAB by a factor of 2, and optimizing out the FFTs to use more efficient image domain filtering still does not match the performance of MATLAB. With a few day’s time spent dedicated to optimizing the Julia code, it would almost certainly be possible to beat the current MATLAB performance. With careful attention to extra memory allocations and copies, as well as using compiled for loops when appropriate, the Julia code could achieve performance that can only be reached in MATLAB by writing MEX functions in C/C++.

4.7 Comparing Results with other pan-sharpening methods

With the modified Sentinel-2 pan-sharpening method using reduced-rank method for the test dataset, we compared it with different papers conducting pan-sharpening using a lower resolution MS images and PAN images. With papers comparing different pan-sharpening algorithms[4], we have selected few of the papers referenced to compare with our results, both quantitatively and qualitatively.
One of the selected method is GS method[19], exploiting the Gram-Schmidt transformation of the data. This

Algorithm Name	NRMSE red	NRMSE green	NRMSE blue	SSIM
AWLP	0.3002	0.0068	0.2276	0.3031
MTF-GLP	0.2862	0.2177	0.2893	0.8032
GS	0.2923	0.2143	0.2870	0.6788
S2 Sharp	0.0016	0.0031	0.0021	0.9999

Table 9: Quantitative analysis with different Algorithms using S2 data

method first changes the MS data to be a zero mean matrix and subtract the same value for the higher resolution Pan image. The weights of each Ms bands are calculated and using the Gram-Schmidt transformation is used to orthogonalize the bands of the digital images.
Another method that the team explored was MTF-tailored MS fusion with HR MS and pan Imagery[20]. The approach here is re-sampling the MS band at a finer scale of the PAN image, which lacks high-frequency details. The high frequency details can be inferred from the high resolution Pan image to supplement the re-sampled Ms band images. Using the GLP-based fusion scheme, we can combine the high pass details of the HR panchromatic images with the expanded low resolution of the panchromatic image and the expanded low resolution multi-spectral image to create a high resolution multi-spectral image.

One other method that was visited was the AWLP method[21], focusing on the additive wavelet Luminance proportions. By proportionally injecting the spectral signature preserved in the HR panchromatic image to each of the LR MS images, relation between the LR MS images are kept.

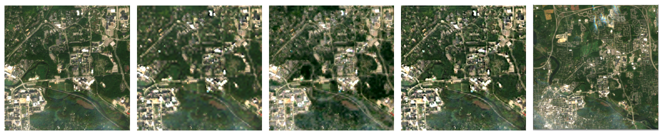


Figure 8: The Pansharpened Ann Arbor image using AWLP, GSm and MTF-GLP, and reduced rank methods, respectively, with the leftmost image being the original image. The images are generated using the AWLP[21], GS[19], and MTF-GLP[20] reduced-rank[6] algorithms, respectivly. The images are displayed using the R,G,B channels of the pansharpened MS images.

To compare the results of the S2 reduced rank method [6] with the methods mentioned above, we calculated the RMSE and SSIM values shown in the table 9 below.

For qualitative analysis comparison of the pansharpened images are shown below in figure**??**.

One possible reason why GS method returned high error rate may be because the GS method required a specified sensor model as the input. While this could return better model when the model is customed to the sensor. However, the GS method did not have a specified sensor for Sentinel-2 data. That is why the input of the GS method was the WorldView2 sensor, which uses the R,G,B, and NIR1 wavelengths, which is similar to Band 2, Band 3, Band 4, and Band 8 of the Sentinel-2 sensor.

5 Conclusion and Future Work

For this project, we improved the accuracy of the S2Sharp sharpening method for multispectral images from the Sentinel-2 satellite. Our modifications focused on edge-preserving properties of the regularization term of the cost function. We modified the edge detection algorithm used to downweight edges in roughness regularization, and we altered the potential function used in the regularization function to an edge-preserving potential. Our modifications showed a clear improvement in the evaluation metrics over the original S2Sharp method.

Our code is stored on a GitHub repository at https://github.com/konradrauscher/eecs556-sharp. This includes modified MATLAB code as well as new Julia code.

Future work would involve optimizing the Julia code to be faster, likely surpassing the speed of the MATLAB implementation. However, since pansharpening is a mature research field, it is generally easier to develop projects in MATLAB, since the majority of the literature relies on the language and affiliated tools.

In addition to programming improvements, we could explore algorithm improvements, possibly looking at options for tuning the potential functions used and/or their parameters, using some form of hyperparameter optimization as used in [6]. We could additionally formulate ways to bring methods such as Total Variation into the context of our optimization problem, and explore alternatives to Manifold optimization for performing the F-Step.

6 Bibliography

[1] Christine Pohl and John van Genderen. *Remote Sensing Image Fusion: A Practical Guide*. CRC Press, 2017. DOI: 10.1201/9781315370101 (cit. on p. 2).

[2] Hassan Ghassemian. "A review of remote sensing image fusion methods". In: *Information fusion* 32 (2016), pp. 75–89. DOI: 10.1016/j.inffus.2016.03.003 (cit. on p. 2).

[3] C Thomas et al. "Synthesis of Multispectral Images to High Spatial Resolution: A Critical Review of Fusion Methods Based on Remote Sensing Physics". In: *IEEE transactions on geoscience and remote sensing* 46.5 (2008), pp. 1301–1312. DOI: 10.1109/TGRS.2007.912448 (cit. on p. 2).

[4] Gemine Vivone et al. "A New Benchmark Based on Recent Advances in Multispectral Pansharpening: Revisiting pansharpening with classical and emerging pansharpening methods". In: *IEEE geoscience and remote sensing magazine* (2020). DOI: 10.1109/MGRS.2020.3019315 (cit. on pp. 2, 4, 16).

[5] *Sentinel-2A Satellite Sensor — Satellite Imaging Corp.* URL: https://www.satimagingcorp.com/satellite-sensors/other-satellite-sensors/sentinel-2a/ (cit. on p. 2).

[6] M. O. Ulfarsson et al. "Sentinel-2 Sharpening Using a Reduced-Rank Method". In: *IEEE Transactions on Geoscience and Remote Sensing* 57.9 (2019), pp. 6408–6420. DOI: 10.1109/TGRS.2019.2906048 (cit. on pp. 2–7, 9, 12, 17, 18).

[7] *WorldView-3 Satellite Sensor — Satellite Imaging Corp.* URL: https://www.satimagingcorp.com/satellite-sensors/worldview-3/ (cit. on pp. 2, 3).

[8] *GitHub - mou12/S2Sharp: Matlab code to sharpen Sentinel-2 satellite images*. URL: https://github.com/mou12/S2Sharp/tree/master/ (cit. on pp. 2, 4, 6, 13).

[9] *Open Access Hub*. URL: https://scihub.copernicus.eu/ (cit. on pp. 3, 6, 7).

[10] *Product Samples — Maxar*. URL: https://www.maxar.com/product-samples (cit. on pp. 3, 6, 16).

[11] *S2Sharp/Data at master · mou12/S2Sharp · GitHub*. URL: https://github.com/mou12/S2Sharp/tree/master/manopt (cit. on p. 6).

[12] *Manopt, a matlab toolbox for optimization on manifolds*. URL: https://www.manopt.org/ (cit. on p. 6).

[13] *Earthdata*. URL: https://earthdata.nasa.gov/ (cit. on p. 6).

[14] Charis Lanaras et al. "Super-resolution of Sentinel-2 images: Learning a globally applicable deep neural network". In: *ISPRS Journal of Photogrammetry and Remote Sensing* 146 (2018), pp. 305–319. ISSN: 0924-2716. DOI: https://doi.org/10.1016/j.isprsjprs.2018.09.018. URL: https://www.sciencedirect.com/science/article/pii/S0924271618302636 (cit. on p. 7).

[15] F. Palsson, J. R. Sveinsson, and M. O. Ulfarsson. "A New Pansharpening Algorithm Based on Total Variation". In: *IEEE Geoscience and Remote Sensing Letters* 11.1 (2014), pp. 318–322. DOI: 10.1109/LGRS.2013.2257669 (cit. on p. 9).

[16] Jeffrey A Fessler. *Chapter 9: Image analysis basics*. URL: https://web.eecs.umich.edu/~fessler/course/556/l/ (cit. on p. 10).

[17] J A Fessler. "Image reconstruction: Algorithms and analysis". URL: https://web.eecs.umich.edu/~fessler/book/ (cit. on pp. 11, 13).

[18] *Pansharpening Toolbox*. DOI: `https://doi.org/10.21982/2q1x-5x14`. URL: `https://rscl-grss.org/coderecord.php?id=541` (cit. on pp. 14, 16).

[19] Craig A Laben and Bernard V Brower. *Process for enhancing the spatial resolution of multispectral imagery using pan-sharpening*. US Patent 6,011,875. Jan. 2000 (cit. on pp. 16, 17).

[20] Bruno Aiazzi et al. “MTF-tailored Multiscale Fusion of High-resolution MS and Pan Imagery”. In: *Photogrammetric Engineering and Remote Sensing* 72 (May 2006), pp. 591–596. DOI: `10.14358/PERS.72.5.591` (cit. on p. 17).

[21] X. Otazu et al. “Introduction of sensor spectral response into image fusion methods. Application to wavelet-based methods”. In: *IEEE Transactions on Geoscience and Remote Sensing* 43.10 (2005), pp. 2376–2385. DOI: `10.1109/TGRS.2005.856106` (cit. on p. 17).

Henry Do hdo	John Gearig johnge	Anusha Kikkeri kikkeria	Konrad Rauscher krausch	Task
15	45	25	15	leadership
25	25	25	25	planning/design
8	30	7	55	programming
25	25	25	25	testing
25	20	25	30	analysis
25	25	25	25	proposal
28	16	28	28	progress report
25	25	25	25	presentation
25	25	25	25	report
15	30	15	35	code repo

Table 10: Group Project Effort