Implementing Continuous High-Resolution Image Reconstruction using Patch Priors

Bachelor Thesis

Sebastian Konietzny

Department of Computer Science
Machine Learning
Heinrich-Heine-Universität Düsseldorf

October 2019

Supervisor:	Prof. Dr. S. Harmeling
Co-Supervisor:	Prof. Dr. A. Schädle
Advisor:	T. Uelwer

Declaration

I hereby confirm that this thesis is my own work. I have documented all sources and tools used. Any direct or indirect quote has been marked as such clearly with specification of the source.

Düsseldorf, 15. October 2019

Sebastian Konietzny

Abstract

In order to observe a distant radio source, a very long baseline interferometry (VLBI) array of disjoint regular-sized radio telescopes around the Earth enables to emulate measurements from a computational telescope with a diameter equal to the distance between these telescopes. These measurements provide only a few noisy samples of constraints, which are approximated as the Fourier transform of the emission's intensity distribution. Therefore, the task is to complete all the missing Fourier transforms in the spartial frequency plane and reconstruct the underlying image using these sparse measurements. However, since there are endless possible images that will fit to the given data, this inverse problem is highly ill-posed.

To focus on solving this task K. L. Bouman et al. designed the algorithm *continuous high-resolution image reconstruction using patch priors* (CHIRP) [3], which simplifies the handling of inhomogeneities in the atmosphere through a problem formulation based on a triple product of measurements and provides an improved forward model to approximate the missing Fourier transforms by maximizing an expected patch log likelihood while still being close to the given measurements.

In this bachelor thesis we work through the concepts of VLBI and CHIRP in detail and provide an implementation of CHIRP demonstrated on synthetically generated and real VLBI measurements.

Contents

1 Introduction

For the purpose of high resolution imaging of distant astronomical radio sources the Event Horizon Telescope (EHT) was created, which simultaneously collects data using an array of disjoint radio telescopes. However, the measurements obtained in this way provide only a few noisy constraints on the emission's intensity distribution, thus requiring particularly designed reconstruction algorithms.
The development of K. L. Bouman et al.'s method, called *continuous high-resolution image reconstruction using patch priors* (CHIRP), made it possible for the EHT Collaboration et al. to evaluate the first Global EHT observations collected at a wavelength of 1.3 mm from 2017 [7]. Their recent achievement yields an image of the event horizon of the black hole in the center of the elliptical galaxy Messier 87 for the first time. Refer to Figure 1.

Objective Building on [3] we start with a detailed introduction into very long baseline interferometry and its corresponding inverse problem. We are giving ideas for solving this problem by summarizing related methods for image reconstruction and subsequently work in depth through the algorithm CHIRP. Finally, we describe the tricks necessary for implementing the algorithm and provide an implementation of CHIRP demonstrated on synthetically generated and real measurements.

Figure 1: First direct visual evidence of the supermassive black hole inside Messier 87 captured by the EHT (10. April 2019)[1].

[1] `https://eso.org/public/images/eso1907a/` (visited on 15. October 2019)

2 Very Long Baseline Interferometry

To provide the necessary background knowledge for this thesis we need to gain a brief insight into the concept of interferometry in radio astronomy.

Definition 1 (Angular Resolution)**.** The *angular resolution* θ (measured in radians) characterizes the capability of a radio telescope to perceive two objects with a small angular distance as separate objects, i.e. differentiate between distant objects that seem to become indistinct. It can be specified by the formula

$$\theta = 1.2197 \frac{\lambda}{D}, \tag{1}$$

where λ is the wavelength of light, and D is the diameter of the telescope [11].

The problem of visualizing distant astronomical radio sources like the emissions surrounding the black hole inside Messier 87 is primarly its required angular resolution, which for this task is substantially smaller than for previous imagings of radio sources [10]. Due to the inverse dependency between telescope diameter and angular resolution in Equation (1), an immense radio telescope could solve this problem. However, to stay with the example of the black hole inside Messier 87, visualizing these emissions requires a 38 μas $\approx 1.824 \cdot 10^{-10}$ rad resolution [7], hence for an observation at a 1.3 mm wavelength a single-dish telescope with a diameter of approximately 8700 km would be nessecary. Nevertheless, even though creating a telescope this size is not feasable, using an array of disjoint regular-sized radio telescopes, called an *interferometer*, enables to emulate measurements from a computational telescope with a diameter equal to the distance between these telescopes. The technique of building such an array around the Earth to simultaneously observe a stellar source is referred to as *very long baseline interferometry* (VLBI) [18]. Two of said VLBI arrays, the EHT and the Global mm-VLBI Array (GMVA), are shown in Figure 2.

2.1 Fourier Transform Relationship

As just mentioned, an interferometer allows to excel the capabilities of a single-dish telescope. To grant an intuitive explanation to the functionality, consider the geometry in Figure 3.

Definition 2 (Baseline & Projected Baseline)**.** The *baseline* between two interferometric radio telescopes, T_i and T_j, located at r_i and r_j, respectively, is given by its distance vector $b_{i,j} = r_j - r_i$. The opposite of the resulting right triangle from T_i, T_j and the source's direction $\hat{s}$ is called the *projected baseline* and is given by $b_{i,j}^{\text{proj}} = b_{i,j} \sin\theta$, where θ is the angle between $b_{i,j}$ and $\hat{s}$.

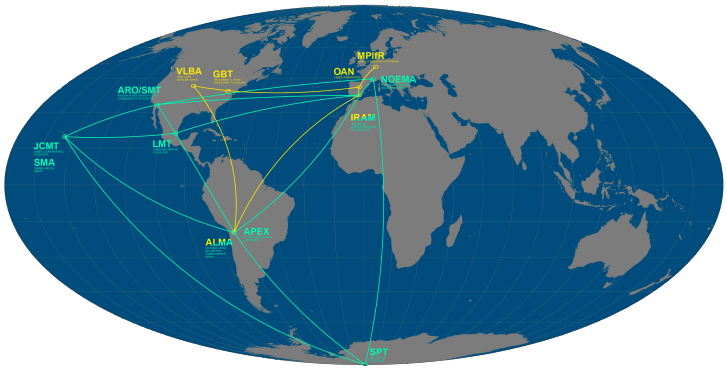

Figure 2: Locations of the involved telescopes of the EHT (cyan) and the GMVA (yellow).[2]

A distant stellar source in the normalized direction $\hat{s}$ radiates light as a spherical wavefront, but due to its sufficient distance, it can be considered that the signal arrives at the telescopes T_i and T_j as a plane orthogonal to the line of sight. The telescopes are geographically separated by the baseline $b_{i,j}$, hence according to the trigonometric functions and the dot product, the plane wave has to travel a supplementary path of

$$|b_{i,j}\cos\theta| = \left|b_{i,j}\cos\arccos\frac{b_{i,j}\cdot\hat{s}}{|b_{i,j}|\,|\hat{s}|}\right| = \left|\frac{b_{i,j}}{|b_{i,j}|}b_{i,j}\cdot\hat{s}\right| = |b_{i,j}\cdot\hat{s}|$$

to reach the farther telescope.

Definition 3 (Geometric Delay)**.** The *geometric delay* τ_g between the received signals of two radio telescopes T_i and T_j is given by their time delay, i.e.

$$\tau_g = \frac{|b_{i,j}\cdot\hat{s}|}{c},$$

where $b_{i,j}$ is the baseline between T_i and T_j, $\hat{s}$ is the normalized source's direction, and c is the velocity of light.

Definition 4 (Time-Averaged Correlation)**.** For continuous real-valued functions f and g with period T , the *time-averaged correlation* is defined as

$$\langle f(\tau)\star g(\tau)\rangle = \lim_{T\to\infty}\frac{1}{T}\int_{t_0}^{t_0+T} f\left(t-\tau\right)g\left(t\right)dt,$$

where $t_0\in\mathbb{R}$.

[2] `https://eso.org/public/images/ann17015a/` (visited on 15. October 2019)

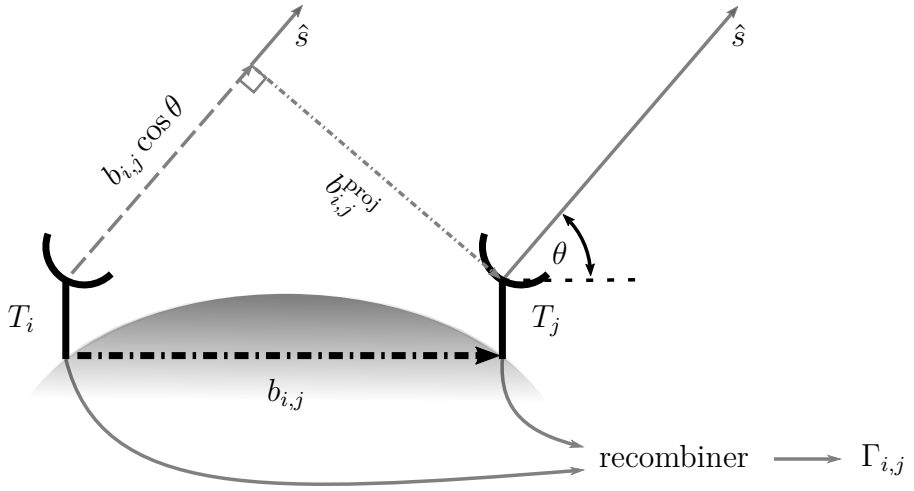

Figure 3: A simplified interferometer of two telescopes, T_i and T_j, measuring visibilities, $\Gamma_{i,j}$, from the source's direction $\hat{s}$. The baseline and projected baseline between T_i and T_j are denoted by $b_{i,j}$ and $b_{i,j}^{\text{proj}}$, respectively [18].

Radio telescopes receive cosmic signals, that are observed as sinusoidal functions of time t with a Gaussian amplitude distribution A [18]. Thus, the outputs of T_i and T_j can be written as $A\cos(\omega(t-\tau_g))$ and $A\cos(\omega t)$, respectively, where $\omega = \frac{2\pi}{T}$ is the angular frequency with respect to the period T. After receiving the measurements the time-average of the multiplicated outputs filters out high frequencies leaving a sinusoidal fringe function with respect to the source's direction $\hat{s}$

$$
\begin{aligned}
&\lim_{T\to\infty}\frac{1}{T}\int_{t_0}^{t_0+T} A\cos(\omega(t-\tau_g))A\cos(\omega t)dt \\
=\ &\lim_{T\to\infty}\frac{1}{T}\int_{t_0}^{t_0+T}\frac{A^2}{2}\left[\cos(\omega t-\omega(t-\tau_g))+\cos(\omega t+\omega(t-\tau_g))\right]dt \\
=\ &\frac{A^2}{2}\left[\cos(\omega\tau_g)+\lim_{T\to\infty}\frac{1}{T}\frac{1}{2\omega}\left[\sin(2\omega(t_0+T)-\omega\tau_g)-\sin(2\omega t_0-\omega\tau_g)\right]\right] \\
=\ &\frac{A^2}{2}\left[\cos(\omega\tau_g)+\lim_{T\to\infty}\frac{1}{T}\frac{1}{2\omega}\left[\sin(2\omega t_0-\omega\tau_g)-\sin(2\omega t_0-\omega\tau_g)\right]\right] \\
=\ &\frac{A^2}{2}\cos(\omega\tau_g) = \frac{A^2}{2}\cos\left(\frac{2\pi}{\lambda}b_{i,j}\cdot\hat{s}\right).
\end{aligned}
\tag{2}
$$

Furthermore, the following theorem generalizes the resulting variation in Equation (2) to extended emissions and uses it to form the time-averaged correlation of the observations to a continuous image of the intensity distribution [18].

Definition 5 (Spatial Frequency Plane)**.** The *spatial frequency plane* is defined as the (u, v)-plane orthogonal to the source's direction where the coordinates u and v (measured in wavelengths) are the east and the north component of the projected baseline, respectively.

Remark. Refer to Figure 4 for an improved insight into the spatial frequency plane.

Theorem 1 (van Cittert-Zernike Theorem [18])**.** For two ideal telescopes, T_i and T_j, the time-averaged correlation of the measured signals at position (u, v) on the spatial frequency plane can be approximated as

$$\Gamma_{i,j}(u, v) \approx \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \mathrm{e}^{-\mathrm{i}2\pi(ul+vm)} I_\lambda(l, m) dl dm, \tag{3}$$

where $I_\lambda(l, m)$ is the emission's intensity distribution of wavelength λ at right ascension l and declination m (measured in degrees).

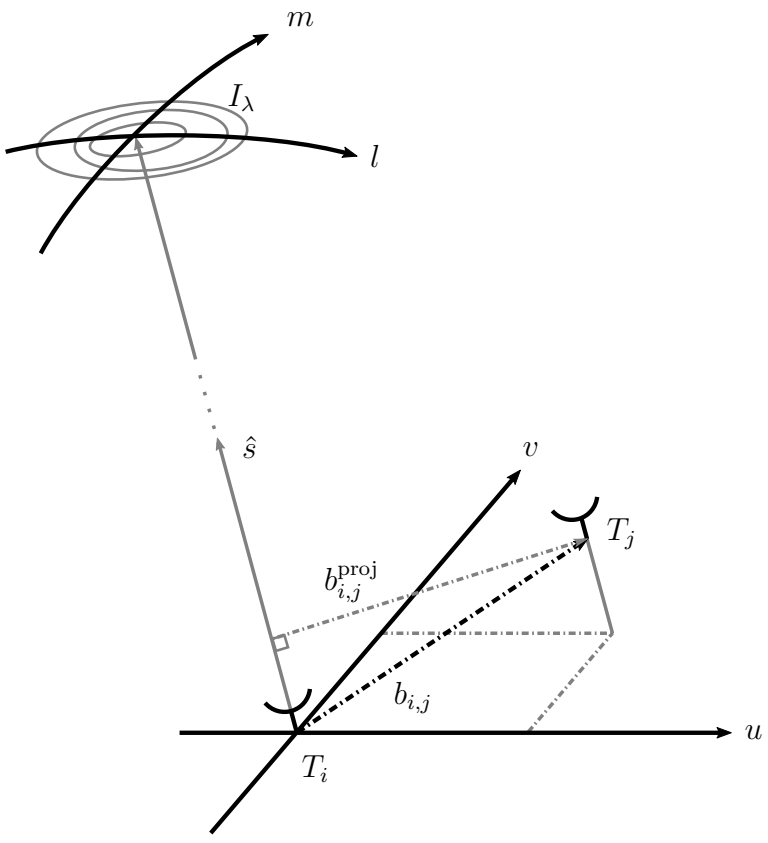

Figure 4: Illustrated relation between the frequency plane of a two-sized interferometry array and the emission's intensity distribution I_λ in the direction $\hat{s} = (l, m, \sqrt{1 - l^2 - m^2})$ [18].

Definition 6 (Visibility)**.** We refer to the measurements of an interferometer $\Gamma_{i,j}$, which are represented by the van Cittert-Zernike theorem as approximations of the Fourier transform of the emissons's intensity distribution, as *visibilities*.

Due to the Fourier transform relationship in Equation (3) and the proportionality between the spatial frequency (u, v) and the baseline $b_{i,j}$ a larger distance between the corresponding telescopes is leading to an expanded coverage of the frequency plane. Therefore, the measured visibility contains finer details of the observed intensity distribution providing an increased resolving power, i.e. a improved angular resolution of the interferometer.

2.2 Frequency Plane Coverage

Reconstructing an expressive image just using a single measured visibility is clearly impossible. Since we are using an array of telescopes, each pair of distinct telescopes creates an individual interferometer generating a single measurement depending on the corresponding baseline. Thus, the VLBI array should contain as many spatially incoherent telescopes as possible with baselines orientated in different directions to cover the spatial frequency plane. However, because of the fixed size of Earth, improving the angular resolution by sufficient large baselines limits the domain of potential telescope locations. If all corresponding N telescopes have a clear view of the source, the array obtains $\binom{N}{2} = \frac{N(N-1)}{2}$ independent samples at a single time, which is for realistic array sizes an extremely sparse number of measurements for image reconstruction.
To increase the number of measurements the source is additionaly observed over time. During Earth's rotation the alignments of the telescopes are constantly changing yielding slightly transformed projected baselines. Thus, the measurements are obtained on elliptical paths in the frequency plane, as shown in Figure 5.

2.3 Termed Phase Closure

Due to the limitation of the maximum distances between the telescopes, to further improve angular resolution, we have to decrease the observing wavelength to millimeter and sub-millimeter wavelengths, which requires to overcome a new issue. The theory behind the measured visibilities thus far disregarded that varying atmospheric noise induces the spherical wavefront to travel at several velocities and therefore introduce an additional delay in each signal. For wavelengths on millimeter scale these phase shifts scramble the observations making them unusable for the reconstruction task [14].
Nevertheless, a simple algebraic trick, called *termed phase closure*, allows to avoid the effect of atmosperic inhomogeneities and obtain some information from the mea-

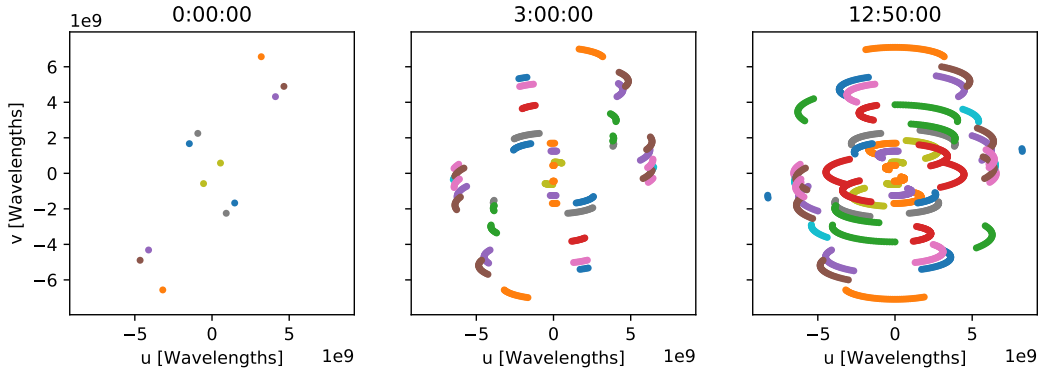

Figure 5: Frequency plane coverage of synthetic VLBI data over time. Note that $\Gamma_{i,j}(-u,-v) = \overline{\Gamma_{i,j}(u,v)}$, i.e. the additional measurement at $(-u,-v)$ does not add information and therefore the observed data is reflected in the origin.

surements. Let ϕ_i and ϕ_j be the unknown phase delays at the telescopes T_i and T_j, respectively, then we can formally represent the disordered, measured visibility $\Gamma_{i,j}^{\text{meas}}$ by the ideal visibility $\Gamma_{i,j}^{\text{ideal}}$ by means of an additional phase term, i.e. $\Gamma_{i,j}^{\text{meas}} = \mathrm{e}^{\mathrm{i}(\phi_i-\phi_j)}\Gamma_{i,j}^{\text{ideal}}$. Hence, the product of three visibilities from different telescopes shortens to an expression without the unknown delays

$$\begin{aligned}\Gamma_{i,j}^{\text{meas}}\Gamma_{j,k}^{\text{meas}}\Gamma_{k,i}^{\text{meas}} &= \mathrm{e}^{\mathrm{i}(\phi_i-\phi_j)}\Gamma_{i,j}^{\text{ideal}}\mathrm{e}^{\mathrm{i}(\phi_j-\phi_k)}\Gamma_{j,k}^{\text{ideal}}\mathrm{e}^{\mathrm{i}(\phi_k-\phi_i)}\Gamma_{k,i}^{\text{ideal}} \\ &= \Gamma_{i,j}^{\text{ideal}}\Gamma_{j,k}^{\text{ideal}}\Gamma_{k,i}^{\text{ideal}}.\end{aligned} \tag{4}$$

Definition 7 (Bispectrum)**.** We refer to the triple product of visibilities from three different telescopes as the *bispectrum*.

In a certain way the usage of bispectrum values instead of visibilities makes the invariance of the atmospheric noise possible, but at the expense of the already sparse number of measurements. Furthermore, even though a N-sized VLBI array consists of $\binom{N}{3}$ different triples, not all of the corresponding bispectrum values are independent, as some can be derived from other bispectrum values [6]. An example of this behavior is shown in Figure 6. In order to construct an independent set of triples, we select a reference telescope and consider only those triples that include the reference [18]. In this way, there are only $\binom{N-1}{2}$ independent bispectrum values. Thus, we restrict ourselves to $\binom{N-1}{2}\Big/\binom{N}{3} = 1 - \frac{2}{N}$ data points, which for small arrays, like the EHT, is an immense drawback. Consequently, the main purpose of the reconstruction algorithm has to be an useful application on sparse measurements.

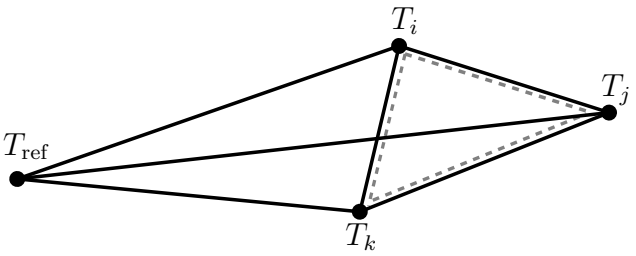

Figure 6: Possible triples in a VLBI array consisting of telescopes T_i, T_j, T_k and T_{ref}. The three triples that include T_{ref} are independent. The bispectrum measurement corresponding to the fourth triple consisting of T_i, T_j and T_k can be derived from the independent ones [18].

2.4 Reconstruction Task

Our intention is to complete all the missing Fourier transforms in the spartial frequency plane and reconstruct the underlying image using the sparse visibilities, or rather the bispectrum values. If the given measurements were noiseless and would cover the entire frequency plane, this task would be solved by a simple inverse Fourier transform. However, VLBI measurements provide only few noisy samples of constraints on the emission's intensity distribution and since there are endless possible images that will fit to this sparse data, the reconstruction task is highly ill-posed.
At this point we can keep the radio astronomy in mind and consider this reconstruction task as a common computational imaging problem.

3 Related Methods

Before we get a detailed insight into the main algorithm of this work, we summarize a few state-of-the-art methods for VLBI image reconstruction.

3.1 CLEAN

One of the most common methods so far is a deconvolution algorithm called *CLEAN*, which was developed by J. Högbom (1974) [8]. With the assumption that the underlying intensity distribution is represented by a discrete number of point sources, the algorithm generates a noisy artifact-heavy initialization image, the so called *dirty image*, by setting all missing data in the spatial frequency plane to zero followed by a simple application of an inverse Fourier transform. Starting from this image, an iterative approach is used to search for the brightest point sources and clean the found areas by removing artifacts that occur due to the missing data. After many iterations, the resulting image will be blurred to scale down the usually incorrectly deduced higher visibilities [16, 18].
However, for millimeter and sub-millimeter wavelengths that the EHT operates at, CLEAN starts to break down since it requires time-consuming data calibration from knowledgeable users to handle the phase delays introduced in Section 2.3.

3.2 SQUEEZE

Although, we differentiate between optical interferometry which operates at visible wavelengths and millimeter and sub-millimeter interferometry, the methods used share many similarities [14].
An optical algorithm capable of the usage of bispectrum measurements and multi-spectral reconstruction, called *SQUEEZE*, was developed by F. Baron, J. D. Monnier and B. Kloppenborg (2010) [1]. It makes use of a Markov chain Monte Carlo approach and multiple gradient-based optimization engines to sample the image's posterior probability distribution. These image samples are not explicitly obtained by searching for the most likely image, but result of averaging statistics under the posterior. After sampling a certain number of images the final image is then calculated as the average of them [13, 17].
Due to its impressive optimization approach, SQUEEZE is not limited in its choice of regularizers or other prior constraints [12]. However, this freedom comes at the cost of a large number of parameter choices that may be hard for an unknowledgeable user to select and tune [3].

3.3 BSMEM

Another optical interferometry reconstruction algorithm, *MEM* (Maximum Entropy Method), has been developed into *BSMEM* (BiSpectrum Maximum Entropy Method) by D. F. Buscher (1994) to regularize the reconstruction problem by means of the bispectrum [5].
Both algorithms apply a fully Bayesian approach to the inverse problem of finding the most likely image given the evidence, making use of a maximum entropy prior to optimize the posterior probability of an image [13, 17].
Although BSMEM is often able to achieve impressive high-resolution results on simple celestial images, it often struggles on complex, extended emissions [3].

4 CHIRP

As addressed in Section 2.4 we want to construct an image that fits the given sparse VLBI measurements, which should also remain restricted to our prior assumption about how the universe looks like. To focus on solving this ill-posed inverse problem various ideas from computer vision were assembled by K. L. Bouman et al. to design a new imaging algorithm, called *continuous high-resolution image reconstruction using patch priors* (CHIRP), by a Bayesian approach. CHIRP simplifies the handling of inhomogeneities in the atmosphere through a problem formulation based on the bispectrum and eventually provides an improved forward model to approximate the missing visibilities in the (u, v) plane [3].

4.1 Continuous Image Representation

In many further imaging techniques the optimization strategy is modeled by assuming the intensity distribution as a discrete set of point sources [16]. Since $I_\lambda(l, m)$ is defined over a continuous space this discretization significantly impairs the optimization in fitting higher visibilities, which mostly contain the desired fine details of the image [3].
To reduce these modeling errors we use a sum of $N_l \cdot N_m$ scaled and shifted pulse functions to parameterize a continuous representation of $I_\lambda(l, m)$ for our optimization method. Let $\left[-\frac{FOV_l}{2}, \frac{FOV_l}{2}\right] \times \left[-\frac{FOV_m}{2}, \frac{FOV_m}{2}\right]$ be the domain of the scene we are interested in and $h(l - l_i, m - m_j)$ a continuous pulse function with a closed-form Fourier transform shifted to

$$\begin{aligned} l_i &= \left(\frac{2i+1}{N_l} - 1\right) \cdot \frac{FOV_l}{2} \;=\; i\Delta_l + \frac{\Delta_l}{2} - \frac{FOV_l}{2} \\ m_j &= \left(\frac{2j+1}{N_m} - 1\right) \cdot \frac{FOV_m}{2} \;=\; j\Delta_m + \frac{\Delta_m}{2} - \frac{FOV_m}{2} \end{aligned} \tag{5}$$

for $i = 0, ..., N_l - 1$ and $j = 0, ..., N_m - 1$, where $\Delta_l = \frac{FOV_l}{N_l}$ and $\Delta_m = \frac{FOV_m}{N_m}$, respectively. We then can represent the continuous image of the intensity distribution as a discrete number of terms, more exactly

$$I_\lambda(l, m) \approx \sum_{i=0}^{N_l-1} \sum_{j=0}^{N_m-1} X[i, j] h\left(l - l_i, m - m_j\right) \tag{6}$$

for a sufficient scaling matrix $X \in \mathbb{R}^{N_l \times N_m}$. In this context, the chosen pulse function plays a crucial role on the interpretation of these image coefficients. For an appropriate choice the function values of the continuous image representation at the pulse centers equal the corresponding coefficients such that these can be interpreted as a discretization of $I_\lambda(l, m)$. Such an appropriate pulse function is described in

more detail in Section 5.2. Therefore, by considering the right-hand side of Equation (6) as a function in terms of $\mathrm{x} = \mathrm{vec}(X)$, hereafter referred to as $\hat{I}_\lambda(\mathrm{x}, l, m)$, we can model our optimization strategy to be aimed at estimating these coefficients. Back to the context of the measured visibilities, replacing the intensity distribution of Equation (3) in the van Cittert-Zernike theorem with the continuous image representation results in

$$
\begin{aligned}
\Gamma(u,v)_{i,j} &\approx \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \mathrm{e}^{-\mathrm{i}2\pi(ul+vm)} \hat{I}_\lambda(\mathrm{x}, l, m) dl dm \\
&= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \mathrm{e}^{-\mathrm{i}2\pi(ul+vm)} \sum_{i=0}^{N_l-1}\sum_{j=0}^{N_m-1} X[i,j] h\left(l-l_i, m-m_j\right) dl dm \\
&= \sum_{i=0}^{N_l-1}\sum_{j=0}^{N_m-1} \Bigg[X[i,j] \\
&\quad \cdot \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \mathrm{e}^{-\mathrm{i}2\pi(ul+vm)} \underbrace{\mathrm{e}^{\mathrm{i}2\pi(ul_i+vm_j)}\mathrm{e}^{-\mathrm{i}2\pi(ul_i+vm_j)}}_{=1} h\left(l-l_i, m-m_j\right) dl dm \Bigg] \\
&= \sum_{i=0}^{N_l-1}\sum_{j=0}^{N_m-1} \Big[X[i,j] \mathrm{e}^{-\mathrm{i}2\pi(ul_i+vm_j)} \\
&\quad \cdot \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \mathrm{e}^{-\mathrm{i}2\pi(u(l-l_i)+v(m-m_j))} h\left(l-l_i, m-m_j\right) dl dm \Big] \\
&= \sum_{i=0}^{N_l-1}\sum_{j=0}^{N_m-1} \Big[X[i,j] \mathrm{e}^{-\mathrm{i}2\pi(ul_i+vm_j)} \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \mathrm{e}^{-\mathrm{i}2\pi(us+vt)} h\left(s,t\right) ds dt \Big] \\
&= \sum_{i=0}^{N_l-1}\sum_{j=0}^{N_m-1} \Big[X[i,j] \mathrm{e}^{-\mathrm{i}2\pi(ul_i+vm_j)} H(u,v) \Big] = \gamma(u,v)^T \mathrm{x}, \qquad (7)
\end{aligned}
$$

where $H(u,v)$ is the Fourier transform of $h(l,m)$ and

$$
\gamma(u,v) = H(u,v) \begin{bmatrix}
\mathrm{e}^{-\mathrm{i}2\pi(ul_0+vm_0)} \\
\mathrm{e}^{-\mathrm{i}2\pi(ul_1+vm_0)} \\
\vdots \\
\mathrm{e}^{-\mathrm{i}2\pi(ul_{N_l-1}+vm_0)} \\
\mathrm{e}^{-\mathrm{i}2\pi(ul_0+vm_1)} \\
\mathrm{e}^{-\mathrm{i}2\pi(ul_1+vm_1)} \\
\vdots \\
\mathrm{e}^{-\mathrm{i}2\pi(ul_{N_l-1}+vm_{N_m-1})}
\end{bmatrix}
$$

for $l_{0,\ldots,N_l-1}$ and $m_{0,\ldots,N_m-1}$ as denoted in Equation (5). Note that if we are aware of the image coefficients X, Equation (7) will provide a closed-form solution to the entire spatial frequency plane.

4.2 Model

The basic idea behind CHIRP is to estimate the recently introduced vectorized image coefficients x by maximizing an expected patch log likelihood (EPLL) while still being close to the given M complex bispectrum measurements, y. More exactly, the cost function we want to minimize in order to reconstruct the image is

$$f_p(\mathrm{x}|\mathrm{y}) = D(\mathrm{y}|\mathrm{x}) - \mathrm{EPLL}_p(\mathrm{x}) \tag{8}$$

for an Gaussian mixture model (GMM) patch prior p. The data term D and prior term EPLL_p are described in more detail below.
Equation (8) has the familar form of a likelihood and prior term from a Bayesian posterior probability, but note that, as suggested in Section 2.3, the data term is expressed with respect to the bispectrum measurements, which are not independent and therefore D is not a log-likelihood. Additionally, since EPLL_p sums over the log probabilities of all overlapping patches, it is not the log-likelihood of $\hat{I}_\lambda$, rather the expected log likelihood of a randomly chosen patch [20].

4.2.1 Data Term

Let y_k be a complex bispectrum measurement of the itensity distribution I_λ corresponding to telescopes T_{k_1}, T_{k_2} and T_{k_3} with observed visibilities at positions (u_{k_1}, v_{k_1}), (u_{k_2}, v_{k_2}) and (u_{k_3}, v_{k_3}) on the spatial frequency plane. Using Equation (7) we can derive an approximated bispectrum ξ_k regarding our image representation $\hat{I}_\lambda$ by

$$\begin{aligned} \mathrm{y}_k &= \Gamma_{k_1,k_2}(u_{k_1}, v_{k_1}) \cdot \Gamma_{k_2,k_3}(u_{k_2}, v_{k_2}) \cdot \Gamma_{k_3,k_1}(u_{k_3}, v_{k_3}) \\ &\approx \gamma(u_{k_1}, v_{k_1})^T \mathrm{x} \cdot \gamma(u_{k_2}, v_{k_2})^T \mathrm{x} \cdot \gamma(u_{k_3}, v_{k_3})^T \mathrm{x} =: \xi_k(\mathrm{x}), \end{aligned} \tag{9}$$

yielding a polynomial equation for the measured bispectrum with respect to the image coefficients x. Although the bispectrum is invariant to the phase delays discussed in Section 2.3, it is not free from residual errors. Thermal noise and gain fluctuation introduce Gaussian noise on the visibilities [18] leading to a Gaussian bounding of the bispectrum's first-order noise terms [2]. Therefore, by approximating each bispectrum measurement's noise distribution with a Gaussian noise Σ_k the data term is evaluated as

$$\begin{aligned} D(\mathrm{y}|\mathrm{x}) &= \frac{\lambda}{2} \sum_{k=1}^{M} \alpha_k \begin{bmatrix} \xi_k^{\Re}(\mathrm{x}) - \mathrm{y}_k^{\Re} \\ \xi_k^{\Im}(\mathrm{x}) - \mathrm{y}_k^{\Im} \end{bmatrix}^T \Sigma_k^{-1} \begin{bmatrix} \xi_k^{\Re}(\mathrm{x}) - \mathrm{y}_k^{\Re} \\ \xi_k^{\Im}(\mathrm{x}) - \mathrm{y}_k^{\Im} \end{bmatrix} \\ &= \frac{\lambda}{2} \sum_{k=1}^{M} \alpha_k \Big[\sigma_{k1} \left(\xi_k^{\Re}(\mathrm{x}) - \mathrm{y}_k^{\Re} \right)^2 + \sigma_{k2} \left(\xi_k^{\Im}(\mathrm{x}) - \mathrm{y}_k^{\Im} \right)^2 \\ &\qquad + \; 2\sigma_{k3} \left(\xi_k^{\Re}(\mathrm{x}) - \mathrm{y}_k^{\Re} \right) \left(\xi_k^{\Im}(\mathrm{x}) - \mathrm{y}_k^{\Im} \right) \Big] \end{aligned} \tag{10}$$

for $\Sigma_k^{-1} = \begin{bmatrix} \sigma_{k1} & \sigma_{k3} \\ \sigma_{k3} & \sigma_{k2} \end{bmatrix}$, where λ is an adjustable parameter and $\alpha_{1,\dots,K}$ are weights to reinforce the independence between the terms, more exactly

$$\alpha_k = \frac{(A_k - 1)(A_k - 2)/2}{\binom{A_k}{3}} = \frac{3}{A_k},$$

where A_k is the number of telescopes observing by the time y_k was measured.

4.2.2 Prior Term

With the aim to model our solution to a reasonable image in terms of our visual assumption we regularize it by maximizing the probabilities all N overlapping patches of fixed size in $\hat{I}_\lambda(\mathrm{x}, l, m)$ using a GMM patch prior trained on a real-world image dataset. Given the vectorized image coefficients x we define the EPLL under prior p as

$$\mathrm{EPLL}_p(\mathrm{x}) = \sum_{i=1}^{N} \log p(P_i \mathrm{x}), \tag{11}$$

where P_i is a matrix that extracts the i-th vectorized patch from x, while $p(P_i\mathrm{x})$ is the probability of that patch [20], i.e.

$$p(P_i\mathrm{x}) = \sum_{j=1}^{K} \pi_j \mathcal{N}(P_i\mathrm{x}|\mu_j, \Sigma_j)$$

for mixing weights $\pi_{1,\dots,K}$, means $\mu_{1,\dots,K}$ and covariance matrices $\Sigma_{1,\dots,K}$ learned through an expectation–maximization (EM) optimization.

4.3 Optimization

Due to the usage of a GMM patch prior, a direct optimization of our model in Equation (8) would be highly challenging. Alternatively, we use an iterative framework, called *half quadratic splitting*, building on [20] to efficiently optimize the cost function f_p.

4.3.1 Half Quadratic Splitting

To avoid the patch prior's dependence on x, we introduce a set of auxiliary variables $Z = \{\mathrm{z}_i\}_{i=1}^N$ corresponding to the N overlapping patches $\{P_i\mathrm{x}\}_{i=1}^N$ in the image

and modify Equation (8) in terms of these variables to an approximated cost function

$$\begin{aligned} c_{p,\beta}(\mathrm{x}, Z|\mathrm{y}) = \frac{\lambda}{2} \sum_{k=1}^{M} \alpha_k \Big[& \sigma_{k1} \left(\xi_k^{\Re}(\mathrm{x}) - \mathrm{y}_k^{\Re} \right)^2 + \sigma_{k2} \left(\xi_k^{\Im}(\mathrm{x}) - \mathrm{y}_k^{\Im} \right)^2 \\ & + \; 2\sigma_{k3} \left(\xi_k^{\Re}(\mathrm{x}) - \mathrm{y}_k^{\Re} \right) \left(\xi_k^{\Im}(\mathrm{x}) - \mathrm{y}_k^{\Im} \right) \Big] \\ & + \sum_{i=1}^{N} \left[\frac{\beta}{2} \| P_i \mathrm{x} - \mathrm{z}_i \|_2^2 - \log p(\mathrm{z}_i) \right] \end{aligned} \tag{12}$$

for a fixed β. In this way we are able to monotonically decrease our original model (8), since the minimization of Equation (12) ensures that the auxiliary variables z_i have to conform to the patches $P_i\mathrm{x}$ for $\beta \to \infty$.
In order to optimize the approximated cost function $c_{p,\beta}$ we start with initialized image coefficients x_0 and iterate between the following steps for a successively increasing β value.

(a) Solve for Z while keeping x constant:

We estimate each auxiliary patch z_i as the most likely patch under the prior, given the current corresponding patch $P_i\mathrm{x}$ and weighting parameter β. However, due to the mixture of Gaussians in Equation (12), these estimates can not readily be specified in closed-form solutions. Therefore, we restrict the GMM to the mixture component which has the highest probability of containing $P_i\mathrm{x}$ to calculate an approximation of the minima with respect to z_i.

(b) Solve for x while keeping Z constant:

The usage of the bispectrum measurements in Equation (12) leads us to optimize a 6th order polynomial in terms of x whose minima cannot be solved in closed-form. Although using gradient descent to search for a local minimum is generally possible, this approach is expensive at runtime. Alternatively, we approximate $\xi_k(\mathrm{x})$ as a function linear in x by means of performing a second order Taylor expansion around our previous solution $\tilde{\mathrm{x}}$ to simplify $c_{p,\beta}$ as a quadratic function which provides a local closed-form solution for x.

Remark. Refer to Section 5.3 for the specific analysis of Z and x.

In conclusion, we frequently solve for z and x, both given their counterparty and the current β, then increasing its value after each iteration. Due to the increase of β, these two steps not only improve the approximated cost $c_{p,\beta}$ from Equation (12) but also the original cost f_p from Equation (8) [20].

4.3.2 Multi-scale Framework

Since half quadratic splitting only approximates our solution x of the cost function $f_p(\mathrm{x}|\mathrm{y})$, we utilize a multi-scale framework to slowly build up our continuous image representation $\hat{I}_\lambda$. After several iterations of optimizing our discretized image, we frequently increase the number of pulse functions, N_l and N_m, used to describe $\hat{I}_\lambda$ and repeat the whole process. As a result, we obtain the desired finer details of the image by optimizing a higher-resolution image on the basis of a already optimized lower-resolution partial solution [3].

5 Implementation

The source code associated with this bachelor thesis is written in Python 3.7.3 and can be found in Appendix A. For orientation purposes we used an implementation[3] of CHIRP, which was simultaneously published with [3]. Refer to Code 1 for an exemplary execution of the algorithm.

5.1 Dataset

In addition to the publication of [3] K. L. Bouman et al. provide a website[4] including a large standardized dataset designed to evaluate VLBI image reconstruction algorithms. For the most part, this dataset consists of synthetic VLBI measurements generated by the MAPS (MIT Array Performance Simulator) package [12]. By selecting a simulated VLBI array configuration one can choose between different targets of calibration examples and static sources consisting of natural, celestial and modelled black hole images [4] at various total flux densities, i.e. noise levels, to obtain the corresponding data.
Furthermore, the website includes real data from the VLBA Boston University Blazar (VLBA-BU-BLAZAR) program, which has been collecting data using the Very Long Baseline Array (VLBA) [9]. Note that since this data consists of real VLBI measurements there are no ground truth images to evaluate the corresponding reconstruction on, only reference images produced by the Boston Univerity Blazar Group.

5.1.1 OIFITS

All the presented measurement data is allocated in the same standardized exchange format, called *OIFITS* (Optical Interferometry Flexible Image Transport System), which expands the Flexible Image Transport System (FITS) to support the storage of optical interferometric observables, such as the bispectrum values and the corresponding uv-coordinates [15]. In order to extract the necessary information for image reconstruction we are using a OIFITS Python module[5] written by Paul Boley.

It is important to note that unlike Definition 5 the measurements' uv-coordinates are stored in OIFITS in meters. For this reason we have to divide each coordinate by the measured wavelength to appropriately convert the unit in wavelengths.

[3] `https://github.com/achael/eht-imaging/tree/960a79557b4de7f2776bcfa1aef2c37cea487ab7` (visited on 15. October 2019)
[4] `http://vlbiimaging.csail.mit.edu/` (visited on 15. October 2019)
[5] `https://github.com/pboley/oifits` (visited on 15. October 2019)

Additionally, since the *uv*-coordinates of a triple product's third visibility are implicit, they are not included in the corresponding bispectrum value of the data format. Thus, for the purpose of correctly transferring the visibilities from OIFITS to our model special attention should be given to the format's order of telescopes in the calculation of the triple product. In comparison to our representation of the termed phase closure the format utilizes an uniform orientation of the interferometric baselines and therefore considers the complex conjugation of the third visibility for building up the bispectrum, i.e. in OIFITS the k-th bispectrum value is calculated as

$$\begin{aligned}
\mathrm{y}_k &= \Gamma_{k_1,k_2}(u_{k_1}, v_{k_1}) \cdot \Gamma_{k_2,k_3}(u_{k_2}, v_{k_2}) \cdot \overline{\Gamma_{k_1,k_3}(u_{k_3}, v_{k_3})} \\
&= \Gamma_{k_1,k_2}(u_{k_1}, v_{k_1}) \cdot \Gamma_{k_2,k_3}(u_{k_2}, v_{k_2}) \cdot \overline{\Gamma_{k_1,k_3}(u_{k_1} + u_{k_2}, v_{k_1} + v_{k_2})} \\
&= \Gamma_{k_1,k_2}(u_{k_1}, v_{k_1}) \cdot \Gamma_{k_2,k_3}(u_{k_2}, v_{k_2}) \cdot \Gamma_{k_1,k_3}(-(u_{k_1} + u_{k_2}), -(v_{k_1} + v_{k_2})) \\
(&= \Gamma_{k_1,k_2}(u_{k_1}, v_{k_1}) \cdot \Gamma_{k_2,k_3}(u_{k_2}, v_{k_2}) \cdot \Gamma_{k_3,k_1}(u_{k_3}, v_{k_3})).
\end{aligned}$$

Hence, we have to additionally negate the third components to adapt the data to our representation. Refer to the method `getData` in Code 3 (line 306-308).

5.1.2 Gaussian Noise

In the publication of OIFITS [15] it is assumed that the bispectrum's Gaussian noise Σ_k introduced in Section 4.2.1 is fully characterized by a noise ellipse in the complex plane. More precisely, this ellipse is described in terms of a variance $\sigma_{\parallel_k}$, denoted as the *amplitude error*, parallel and a variance $\sigma_{\perp k}$ perpendicular to the bispectrum measurement y_k. Further, the so-called *phase error* $\sigma_{\theta k}$ is defined as a parameterization of the perpendicular variance via

$$\sigma_{\theta k} = \frac{180}{\pi} \frac{\sigma_{\perp k}}{|\mathrm{y}_k|},$$

which is stored together with the amplitude error $\sigma_{\parallel_k}$ in the exchange format's bispectrum data type.
However, the given data fulfils the equation

$$\sigma_{\theta k} = \frac{180}{\pi} \frac{\sigma_{\parallel_k}}{|\mathrm{y}_k|}$$

leading to the assumption that the perpendicular and parallel variances are equal and therefore the Gaussian noise is circular in the complex plane, i.e.

$$\left(\Leftrightarrow \quad \Sigma_k^{-1} = \begin{bmatrix} \frac{1}{\sigma_{\parallel_k}^2} & 0 \\ 0 & \frac{1}{\sigma_{\parallel_k}^2} \end{bmatrix} \right) \Sigma_k = \begin{bmatrix} \sigma_{\parallel_k}^2 & 0 \\ 0 & \sigma_{\parallel_k}^2 \end{bmatrix}.$$

Thus, the approximated cost function $c_{p,\beta}(\mathrm{x}, Z|\mathrm{y})$ from Equation (12) simplifies to

$$\begin{aligned} c_{p,\beta}(\mathrm{x}, Z|\mathrm{y}) = \frac{\lambda}{2} \sum_{k=1}^{M} \frac{\alpha_k}{{\sigma_\parallel}_k^2} \left[\left(\xi_k^{\Re}(\mathrm{x}) - \mathrm{y}_k^{\Re} \right)^2 + \left(\xi_k^{\Im}(\mathrm{x}) - \mathrm{y}_k^{\Im} \right)^2 \right] \\ + \sum_{i=1}^{N} \left[\frac{\beta}{2} \| P_i \mathrm{x} - \mathrm{z}_i \|_2^2 - \log p(\mathrm{z}_i) \right] \end{aligned} \tag{13}$$

5.2 Parameter Selection

In the following, we summarize the still pending parameters chosen at the suggestion of K. L. Bouman et al. in [3]. However, all set parameters discussed below can be manually modified through the constructer `CHIRP.__init__`, refer to Code 2 (line 46-65).

Pulse Function and Fourier Transform As addressed in Section 4.1 the chosen pulse in the continuous image representation from Equation (6) plays a crucial role on the interpretation of the image coefficients X to be reconstructed during our optimization. The choice of a two-dimensional triangular function with a base of width $(2\Delta_l, 2\Delta_m)$, i.e.

$$\mathrm{tri}\left(\frac{l}{\Delta_l}\right) \cdot \mathrm{tri}\left(\frac{m}{\Delta_m}\right),$$

where

$$\mathrm{tri}(x) = \max(1 - |x|, 0) = \begin{cases} 1 - |x| & \text{if } |x| < 1 \\ 0 & \text{otherwise} \end{cases},$$

remains the image coefficients unaffected at the pulse centers and interpolates the intermediate points between them. Although the image coefficients can be interpreted in this way as a discretization of the emission's intensity distribution, the overall continuous image representation is not properly approximating the distribution, since we have disregarded its unit (janskys). Each pixel in the discretization shall contain information about an area of $\Delta_l \cdot \Delta_m$ in the continuous image representation, due to our choice of pulse width. Therefore, in order to properly respect the intensity distribution's unit we adjust the chosen pulse function to

$$h(l, m) = \frac{1}{\Delta_l} \mathrm{tri}\left(\frac{l}{\Delta_l}\right) \cdot \frac{1}{\Delta_m} \mathrm{tri}\left(\frac{m}{\Delta_m}\right)$$

with the closed-form Fourier transform

$$\begin{aligned}
H(u,v) &= \mathcal{F}\{h(l,m)\}(u,v) = \mathcal{F}\left\{\frac{1}{\Delta_l}\mathrm{tri}\left(\frac{l}{\Delta_l}\right)\cdot\frac{1}{\Delta_m}\mathrm{tri}\left(\frac{m}{\Delta_m}\right)\right\}(u,v) \\
&= \frac{1}{\Delta_l}\mathcal{F}\left\{\mathrm{tri}\left(\frac{l}{\Delta_l}\right)\right\}(u)\cdot\frac{1}{\Delta_m}\mathcal{F}\left\{\mathrm{tri}\left(\frac{m}{\Delta_m}\right)\right\}(v) \\
&= \frac{1}{\Delta_l^2}\mathcal{F}\left\{\mathrm{rect}\left(\frac{l}{\Delta_l}\right)*\mathrm{rect}\left(\frac{l}{\Delta_l}\right)\right\}(u) \\
&\quad\cdot\frac{1}{\Delta_m^2}\mathcal{F}\left\{\mathrm{rect}\left(\frac{m}{\Delta_m}\right)*\mathrm{rect}\left(\frac{m}{\Delta_m}\right)\right\}(v) \\
&= \frac{1}{\Delta_l^2}\mathcal{F}\left\{\mathrm{rect}\left(\frac{l}{\Delta_l}\right)\right\}^2(u)\cdot\frac{1}{\Delta_m^2}\mathcal{F}\left\{\mathrm{rect}\left(\frac{m}{\Delta_m}\right)\right\}^2(v) \\
&= \mathrm{sinc}\left(\Delta_l u\right)^2\cdot\mathrm{sinc}\left(\Delta_m v\right)^2,
\end{aligned}$$

where

$$\mathrm{rect}(x) = \begin{cases} 1 & \text{if } |x| < \frac{1}{2} \\ 0 & \text{otherwise} \end{cases}$$

and

$$\mathrm{sinc}(x) = \begin{cases} 1 & \text{if } x = 0 \\ \frac{\sin(\pi x)}{\pi x} & \text{otherwise} \end{cases}.$$

Consequently, the scaled image coefficients $\Delta_l \cdot \Delta_m \cdot X$ provide a discretized image in the required unit (janskys per pixel).

Gaussian Mixture Model and Patch Size For the patch prior used in Equation (11) to regularize our model, we choose the pre-trained Gaussian Mixture Model `naturalPrior.mat` provided on the given implementation's GitHub repository. This GMM is trained with respect to 200 mixture components on real-world natural image patches of size 8×8. Therefore, to alter the patch size from 8×8 we have to train an individual GMM.

β Values Although in theory half quadratic splitting only ensures a monotonically decrease of our original model (8) for $\beta \to \infty$, we have to restrict ourselves to a finite number of increasing β values. Following the suggestion from [20], we iterate between the optimization steps (a) and (b) for $\beta = 1, 4, 8, 16, 32, 64, 128, 256, 512$ in order to minimize each sub-problem.

Multi-Scaling As addressed in Section 4.3.2, we slowly build up our continuous image representation by frequently increasing its discretization size. After each run of half quadratic splitting we scale the resulting image coefficients up and start over again. Refer to the method `upscaleImage` in Code 3 (line 62-95). We start with an initialisation scale of 20 and frequently increase to 64 over 10 scales, more precisely $N_l = N_m = 20, 23, 26, 29, 34, 39, 44, 50, 56, 64$.

5.3 Optimization Steps

Finally, we discuss the necessary analysis for the individual optimization steps in detail.

5.3.1 Initialization

To start with a reasonable approximation of the to be reconstructed image we take a look back at Equation (7). For the given measured visibilities $\Gamma \in \mathbb{C}^M$ we then have the following system of equations, which is linear in x

$$\Gamma \approx \underbrace{\begin{bmatrix} \gamma(u_1, v_1)^T \\ \vdots \\ \gamma(u_M, v_M)^T \end{bmatrix}}_{=:G} \mathrm{x}.$$

Building on ideas from [17] we want to find the minimal image x_0 such that $G\mathrm{x}_0 = \Gamma$, i.e. solve the constrained optimization problem

$$\begin{aligned} &\min_{\mathrm{x}} \ \frac{1}{2}\|\mathrm{x}\|_2^2 \\ &\text{s.t. } G\mathrm{x} - \Gamma = 0. \end{aligned}$$

The corresponding Lagrangian is

$$\begin{aligned} \mathcal{L}(\mathrm{x}, \alpha) &= \frac{1}{2}\|\mathrm{x}\|_2^2 - \alpha\left(G\mathrm{x} - \Gamma\right) \\ &= \frac{1}{2}\|\mathrm{x}\|_2^2 - \langle \alpha, G\mathrm{x} - \Gamma \rangle \\ &= \frac{1}{2}\|\mathrm{x}\|_2^2 - \left\langle \overline{G}^T\alpha, \mathrm{x} \right\rangle + \langle \alpha, \Gamma \rangle, \end{aligned}$$

where its gradient with respect to x

$$\frac{\partial}{\partial \mathrm{x}}\mathcal{L}(\mathrm{x}, \alpha) = \mathrm{x} - \overline{G}^T\alpha$$

is leading to the solution $\mathrm{x}_0 = \overline{G^T}\alpha$ in terms of the constraint holding Langrange multiplier $\alpha \in \mathbb{R}^M$. Through the equality constraint we then obtain $G\overline{G^T}\alpha = \Gamma$, where

$$\begin{aligned}\left(G \cdot \overline{G^T}\right)_{k,k'} &= H(u_k, v_k)\overline{H(u_{k'}, v_{k'})} \sum_{i=0}^{N_l-1} \sum_{j=0}^{N_m-1} \mathrm{e}^{-\mathrm{i}2\pi(u_k l_i + v_k m_j)} \overline{\mathrm{e}^{-\mathrm{i}2\pi(u_{k'} l_i + v_{k'} m_j)}} \\ &= H(u_k, v_k)H(-u_{k'}, -v_{k'}) \sum_{i=0}^{N_l-1} \sum_{j=0}^{N_m-1} \mathrm{e}^{-\mathrm{i}2\pi(l_i(u_k - u_{k'}) + m_j(v_k - v_{k'}))}.\end{aligned} \tag{14}$$

In order to solve this problem for α, we approximate the sum in the right-hand side of Equation (14) to be equal to zero for $k \neq k'$, i.e.

$$\alpha_k \approx \frac{\Gamma_k}{H(u_k, v_k)H(-u_k, -v_k)N_l N_m} =: \tilde{\Gamma}_k,$$

resulting in

$$\mathrm{x}_0 \approx \overline{G^T}\tilde{\Gamma}. \tag{15}$$

Refer to the method `initImage` in Code 3 (line 32-60).

5.3.2 Most Likely Patches under the Prior

As addressed in Section 4.3.1 we want to simplify the prior term (11) by restricting the GMM to the mixture component which has the highest probability of containing $P_i\mathrm{x} \in \mathbb{R}^n$. Thus, we determine each patch's mixture component, i.e.

$$\begin{aligned}j_i^\star = \underset{j\in\{1,\dots,K\}}{\arg\max}\, p(c_j = 1|P_i\mathrm{x}) &= \underset{j\in\{1,\dots,K\}}{\arg\max} \frac{p(P_i\mathrm{x}|c_j = 1)p(c_j = 1)}{\sum_{k=1}^K p(P_i\mathrm{x}|c_k = 1)p(c_k = 1)} \\ &= \underset{j\in\{1,\dots,K\}}{\arg\max}\, p(P_i\mathrm{x}|c_j = 1)p(c_j = 1) \\ &= \underset{j\in\{1,\dots,K\}}{\arg\max}\, \pi_j \mathcal{N}(P_i\mathrm{x}|\mu_j, \Sigma_j) \\ &= \underset{j\in\{1,\dots,K\}}{\arg\max}\, \log\left(\pi_j \mathcal{N}(P_i\mathrm{x}|\mu_j, \Sigma_j)\right) \\ &= \underset{j\in\{1,\dots,K\}}{\arg\max} \Big[\log(\pi_j) - \frac{n}{2}\log(2\pi) - \frac{1}{2}\log\left(\det\left(\Sigma_j\right)\right) \\ &\qquad -\frac{1}{2}\left(\mathrm{z}_i - \mu_j\right)^T \Sigma_j^{-1}\left(\mathrm{z}_i - \mu_j\right)\Big]\end{aligned} \tag{16}$$

and replace each sum of Gaussians $\sum_{j=1}^K \pi_j \mathcal{N}(P_i\mathrm{x}|\mu_j, \Sigma_j)$ by its determined maximal term $\pi_{j_i^\star}\mathcal{N}(P_i\mathrm{x}|\mu_{j_i^\star}, \Sigma_{j_i^\star})$. In combination with Equation (13) we then obtain the

following approximated cost function

$$\begin{aligned} c_{p,\beta}(\mathrm{x}, Z|\mathrm{y}) \approx \frac{\lambda}{2} \sum_{k=1}^{M} \frac{\alpha_k}{\sigma_{\|k}^2} \left[\left(\xi_k(\mathrm{x})^{\Re} - \mathrm{y}_k^{\Re} \right)^2 + \left(\xi_k(\mathrm{x})^{\Im} - \mathrm{y}_k^{\Im} \right)^2 \right] \\ + \sum_{i=1}^{N} \left[\frac{\beta}{2} \|P_i \mathrm{x} - \mathrm{z}_i\|_2^2 - \log \left(\pi_{j_i^\star} \mathcal{N}(P_i \mathrm{x} | \mu_{j_i^\star}, \Sigma_{j_i^\star}) \right) \right] \\ =: c_{p,\beta}^\star(\mathrm{x}, Z|\mathrm{y}). \end{aligned}$$

For the purpose of estimating the auxiliary patches $Z = \{\mathrm{z}_i\}_{i=1}^N$ we derive $c_{p,\beta}^\star$ with respect to z_i

$$\begin{aligned} \frac{\partial}{\partial \mathrm{z}_i} c_{p,\beta}^\star(\mathrm{x}, Z|\mathrm{y}) &= \frac{\lambda}{2} \sum_{k=1}^{M} \frac{\alpha_k}{\sigma_{\|k}^2} \left[\frac{\partial}{\partial \mathrm{z}_i} \left(\xi_k(\mathrm{x})^{\Re} - \mathrm{y}_k^{\Re} \right)^2 + \frac{\partial}{\partial \mathrm{z}_i} \left(\xi_k(\mathrm{x})^{\Im} - \mathrm{y}_k^{\Im} \right)^2 \right] \\ &\quad + \sum_{n=1}^{N} \left[\frac{\partial}{\partial \mathrm{z}_i} \frac{\beta}{2} \|P_n \mathrm{x} - \mathrm{z}_n\|_2^2 - \frac{\partial}{\partial \mathrm{z}_i} \log \left(\pi_{j_n^\star} \mathcal{N}(P_n \mathrm{x} | \mu_{j_n^\star}, \Sigma_{j_n^\star}) \right) \right] \\ &= \frac{\partial}{\partial \mathrm{z}_i} \frac{\beta}{2} \|P_i \mathrm{x} - \mathrm{z}_i\|_2^2 - \frac{\partial}{\partial \mathrm{z}_i} \log \left(\pi_{j_i^\star} \mathcal{N}(P_i \mathrm{x} | \mu_{j_i^\star}, \Sigma_{j_i^\star}) \right) \\ &= -\beta (P_i \mathrm{x} - \mathrm{z}_i)^T - \frac{1}{\pi_{j_i^\star} \mathcal{N}(P_i \mathrm{x} | \mu_{j_i^\star}, \Sigma_{j_i^\star})} \frac{\partial}{\partial \mathrm{z}_i} \pi_{j_i^\star} \mathcal{N}(P_i \mathrm{x} | \mu_{j_i^\star}, \Sigma_{j_i^\star}) \\ &= -\beta (P_i \mathrm{x} - \mathrm{z}_i)^T + \left(\mathrm{z}_i - \mu_{j_i^\star} \right)^T \Sigma_{j_i^\star}^{-1}, \end{aligned}$$

where

$$\begin{aligned} \frac{\partial}{\partial \mathrm{z}_i} \mathcal{N}(P_i \mathrm{x} | \mu_j, \Sigma_j) &= \frac{\partial}{\partial \mathrm{z}_i} \frac{1}{\sqrt{(2\pi)^n \det(\Sigma_j)}} \exp \left(-\frac{1}{2} (\mathrm{z}_i - \mu_j)^T \Sigma_j^{-1} (\mathrm{z}_i - \mu_j) \right) \\ &= \frac{1}{\sqrt{(2\pi)^n \det(\Sigma_j)}} \exp \left(-\frac{1}{2} (\mathrm{z}_i - \mu_j)^T \Sigma_j^{-1} (\mathrm{z}_i - \mu_j) \right) \\ &\quad \cdot \frac{\partial}{\partial \mathrm{z}_i} \left[-\frac{1}{2} (\mathrm{z}_i - \mu_j)^T \Sigma_j^{-1} (\mathrm{z}_i - \mu_j) \right] \\ &= \mathcal{N}(P_i \mathrm{x} | \mu_j, \Sigma_j) \left[-(\mathrm{z}_i - \mu_j)^T \Sigma_j^{-1} \right] \end{aligned}$$

and solve for z_i

$$\begin{aligned} & & 0 &= -\beta(P_i\mathrm{x} - \mathrm{z}_i)^T + \left(\mathrm{z}_i - \mu_{j_i^\star}\right)^T \Sigma_{j_i^\star}^{-1} \\ \Leftrightarrow & & \beta(P_i\mathrm{x} - \mathrm{z}_i) &= \Sigma_{j_i^\star}^{-1}\left(\mathrm{z}_i - \mu_{j_i^\star}\right) \\ \Leftrightarrow & & \Sigma_{j_i^\star}^{-1}\mathrm{z}_i + \beta\mathrm{z}_i &= \beta P_i\mathrm{x} + \Sigma_{j_i^\star}^{-1}\mu_{j_i^\star} \\ \Leftrightarrow & & \Sigma_{j_i^\star}\mathrm{z}_i + \frac{1}{\beta}\mathrm{z}_i &= \Sigma_{j_i^\star}P_i\mathrm{x} + \frac{1}{\beta}\mu_{j_i^\star} \\ \Leftrightarrow & & \mathrm{z}_i &= \left(\Sigma_{j_i^\star} + \frac{1}{\beta}I\right)^{-1}\left(\Sigma_{j_i^\star}P_i\mathrm{x} + \frac{1}{\beta}\mu_{j_i^\star}\right). \end{aligned} \tag{17}$$

For reasons of efficiency we combine afterwards the auxiliary patches into a single prior image. Refer to the method `mostLikelyPatches` in Code 3 (line 97-174).

5.3.3 Taylor Expansion

In order to avoid optimizing a 6th order polynomial, we perform a second order Taylor expansion around our previous solution $\tilde{\mathrm{x}}$ to approximate $\xi_k^{\Re}(\mathrm{x})$ and $\xi_k^{\Im}(\mathrm{x})$ as functions linear in x, i.e.

$$\xi_k(\mathrm{x}) \approx \xi_k(\tilde{\mathrm{x}}) + \left(\frac{\partial}{\partial \mathrm{x}}\xi_k(\tilde{\mathrm{x}})\right)(\mathrm{x} - \tilde{\mathrm{x}}) = \underbrace{\left(\frac{\partial}{\partial \mathrm{x}}\xi_k(\tilde{\mathrm{x}})\right)}_{=:a_k}\mathrm{x} + \underbrace{\xi_k(\tilde{\mathrm{x}}) - \left(\frac{\partial}{\partial \mathrm{x}}\xi_k(\tilde{\mathrm{x}})\right)\tilde{\mathrm{x}}}_{=:b_k}. \tag{18}$$

Substituting these linear functions into Equation (13) results in the following approximated cost function quadratic in x

$$\begin{aligned} c_{p,\beta}(\mathrm{x}, Z|\mathrm{y}) \approx & \frac{\lambda}{2}\sum_{k=1}^{M}\frac{\alpha_k}{\sigma_{\|k}^2}\left[\left(a_k^{\Re}\mathrm{x} + b_k^{\Re} - \mathrm{y}_k^{\Re}\right)^2 + \left(a_k^{\Im}\mathrm{x} + b_k^{\Im} - \mathrm{y}_k^{\Im}\right)^2\right] \\ & + \sum_{i=1}^{N}\left[\frac{\beta}{2}\|P_i\mathrm{x} - \mathrm{z}_i\|_2^2 - \log p(\mathrm{z}_i)\right] \\ & =: \tilde{c}_{p,\beta}(\mathrm{x}, Z|\mathrm{y}). \end{aligned}$$

The derivative of $\tilde{c}_{p,\beta}(\mathrm{x}, Z|\mathrm{y})$ is then given by

$$\begin{aligned}
\frac{\partial}{\partial \mathrm{x}}\tilde{c}_{p,\beta}(\mathrm{x}, Z|\mathrm{y}) &= \frac{\lambda}{2}\sum_{k=1}^{M}\frac{\alpha_k}{\sigma_{\|k}^2}\left[\frac{\partial}{\partial \mathrm{x}}\left(a_k^{\Re}\mathrm{x} + b_k^{\Re} - \mathrm{y}_k^{\Re}\right)^2 + \frac{\partial}{\partial \mathrm{x}}\left(a_k^{\Im}\mathrm{x} + b_k^{\Im} - \mathrm{y}_k^{\Im}\right)^2\right] \\
&\quad + \sum_{i=1}^{N}\left[\frac{\partial}{\partial \mathrm{x}}\frac{\beta}{2}\|P_i\mathrm{x} - \mathrm{z}_i\|_2^2 - \frac{\partial}{\partial \mathrm{x}}\log p(\mathrm{z}_i)\right] \\
&= \lambda\sum_{k=1}^{M}\frac{\alpha_k}{\sigma_{\|k}^2}\left[\left(a_k^{\Re}\mathrm{x} + b_k^{\Re} - \mathrm{y}_k^{\Re}\right)a_k^{\Re,T} + \left(a_k^{\Im}\mathrm{x} + b_k^{\Im} - \mathrm{y}_k^{\Im}\right)a_k^{\Im,T}\right] \\
&\quad + \beta\sum_{i=1}^{N}P_i^T(P_i\mathrm{x} - \mathrm{z}_i) \\
&= \lambda\sum_{k=1}^{M}\frac{\alpha_k}{\sigma_{\|k}^2}\left[a_k^{\Re,T}a_k^{\Re}\mathrm{x} + a_k^{\Im,T}a_k^{\Im}\mathrm{x}\right] + \beta\sum_{i=1}^{N}P_i^TP_i\mathrm{x} \\
&\quad + \lambda\sum_{k=1}^{M}\frac{\alpha_k}{\sigma_{\|k}^2}\left[a_k^{\Re,T}\left(b_k^{\Re} - \mathrm{y}_k^{\Re}\right) + a_k^{\Im,T}\left(b_k^{\Im} - \mathrm{y}_k^{\Im}\right)\right] - \beta\sum_{i=1}^{N}P_i^T\mathrm{z}_i \\
&= \left[\lambda\sum_{k=1}^{M}\frac{\alpha_k}{\sigma_{\|k}^2}\left[a_k^{\Re,T}a_k^{\Re} + a_k^{\Im,T}a_k^{\Im}\right] + \beta\sum_{i=1}^{N}P_i^TP_i\right]\mathrm{x} \\
&\quad - \left[-\lambda\sum_{k=1}^{M}\frac{\alpha_k}{\sigma_{\|k}^2}\left[a_k^{\Re,T}\left(b_k^{\Re} - \mathrm{y}_k^{\Re}\right) + a_k^{\Im,T}\left(b_k^{\Im} - \mathrm{y}_k^{\Im}\right)\right] + \beta\sum_{i=1}^{N}P_i^T\mathrm{z}_i\right]
\end{aligned}$$

and we are able to solve for x in closed-form, i.e.

$$\begin{aligned}
\mathrm{x} = &\left[\lambda\sum_{k=1}^{M}\frac{\alpha_k}{\sigma_{\|k}^2}\left[a_k^{\Re,T}a_k^{\Re} + a_k^{\Im,T}a_k^{\Im}\right] + \beta\sum_{i=1}^{N}P_i^TP_i\right]^{-1} \\
&\cdot\left[-\lambda\sum_{k=1}^{M}\frac{\alpha_k}{\sigma_{\|k}^2}\left[a_k^{\Re,T}\left(b_k^{\Re} - \mathrm{y}_k^{\Re}\right) + a_k^{\Im,T}\left(b_k^{\Im} - \mathrm{y}_k^{\Im}\right)\right] + \beta\sum_{i=1}^{N}P_i^T\mathrm{z}_i\right].
\end{aligned} \tag{19}$$

However, we still need to determine the derivatives of $\xi_k^{\Re}(\mathrm{x})$ and $\xi_k^{\Im}(\mathrm{x})$ to calculate the constants a_k and b_k from Equation (18). For this purpose we take a look back

at Equation (9) and determine the real and imaginary part of $\xi_k(\mathrm{x})$ in detail, i.e.

$$\begin{aligned}
\xi_k(\mathrm{x}) &= \gamma(u_{k_1}, v_{k_1})^T\mathrm{x} \cdot \gamma(u_{k_2}, v_{k_2})^T\mathrm{x} \cdot \gamma(u_{k_3}, v_{k_3})^T\mathrm{x} \\
&= \left(\gamma_{k_1}^{\Re} + \mathrm{i}\gamma_{k_1}^{\Im}\right)^T \mathrm{x} \cdot \left(\gamma_{k_2}^{\Re} + \mathrm{i}\gamma_{k_2}^{\Im}\right)^T \mathrm{x} \cdot \left(\gamma_{k_3}^{\Re} + \mathrm{i}\gamma_{k_3}^{\Im}\right)^T \mathrm{x} \\
&= \left(\gamma_{k_1}^{\Re,T}\mathrm{x} + \mathrm{i}\gamma_{k_1}^{\Im,T}\mathrm{x}\right) \cdot \left(\gamma_{k_2}^{\Re,T}\mathrm{x} + \mathrm{i}\gamma_{k_2}^{\Im,T}\mathrm{x}\right) \cdot \left(\gamma_{k_3}^{\Re,T}\mathrm{x} + \mathrm{i}\gamma_{k_3}^{\Im,T}\mathrm{x}\right) \\
&= \ \gamma_{k_1}^{\Re,T}\mathrm{x} \cdot \gamma_{k_2}^{\Re,T}\mathrm{x} \cdot \gamma_{k_3}^{\Re,T}\mathrm{x} - \gamma_{k_1}^{\Im,T}\mathrm{x} \cdot \gamma_{k_2}^{\Im,T}\mathrm{x} \cdot \gamma_{k_3}^{\Re,T}\mathrm{x} \\
&\quad - \gamma_{k_1}^{\Re,T}\mathrm{x} \cdot \gamma_{k_2}^{\Im,T}\mathrm{x} \cdot \gamma_{k_3}^{\Im,T}\mathrm{x} - \gamma_{k_1}^{\Im,T}\mathrm{x} \cdot \gamma_{k_2}^{\Re,T}\mathrm{x} \cdot \gamma_{k_3}^{\Im,T}\mathrm{x} \\
&\quad + \mathrm{i}\left[\gamma_{k_1}^{\Re,T}\mathrm{x} \cdot \gamma_{k_2}^{\Im,T}\mathrm{x} \cdot \gamma_{k_3}^{\Re,T}\mathrm{x} + \gamma_{k_1}^{\Im,T}\mathrm{x} \cdot \gamma_{k_2}^{\Re,T}\mathrm{x} \cdot \gamma_{k_3}^{\Re,T}\mathrm{x}\right. \\
&\quad \left. + \gamma_{k_1}^{\Re,T}\mathrm{x} \cdot \gamma_{k_2}^{\Re,T}\mathrm{x} \cdot \gamma_{k_3}^{\Im,T}\mathrm{x} - \gamma_{k_1}^{\Im,T}\mathrm{x} \cdot \gamma_{k_2}^{\Im,T}\mathrm{x} \cdot \gamma_{k_3}^{\Im,T}\mathrm{x}\right] \\
&= \xi_k^{\Re}(\mathrm{x}) + \mathrm{i}\xi_k^{\Im}(\mathrm{x})
\end{aligned} \tag{20}$$

for $\gamma_{k_i} = \gamma(u_{k_i}, v_{k_i})$. Therefore, we find

$$\begin{aligned}
\frac{\partial}{\partial \mathrm{x}}\xi_k^{\Re}(\mathrm{x}) &= \frac{\partial}{\partial \mathrm{x}}\left(\gamma_{k_1}^{\Re,T}\mathrm{x} \cdot \gamma_{k_2}^{\Re,T}\mathrm{x} \cdot \gamma_{k_3}^{\Re,T}\mathrm{x}\right) - \frac{\partial}{\partial \mathrm{x}}\left(\gamma_{k_1}^{\Im,T}\mathrm{x} \cdot \gamma_{k_2}^{\Im,T}\mathrm{x} \cdot \gamma_{k_3}^{\Re,T}\mathrm{x}\right) \\
&\quad - \frac{\partial}{\partial \mathrm{x}}\left(\gamma_{k_1}^{\Re,T}\mathrm{x} \cdot \gamma_{k_2}^{\Im,T}\mathrm{x} \cdot \gamma_{k_3}^{\Im,T}\mathrm{x}\right) - \frac{\partial}{\partial \mathrm{x}}\left(\gamma_{k_1}^{\Im,T}\mathrm{x} \cdot \gamma_{k_2}^{\Re,T}\mathrm{x} \cdot \gamma_{k_3}^{\Im,T}\mathrm{x}\right) \\
&= \gamma_{k_1}^{\Re,T}\left(\gamma_{k_2}^{\Re,T}\mathrm{x} \cdot \gamma_{k_3}^{\Re,T}\mathrm{x}\right) + \gamma_{k_2}^{\Re,T}\left(\gamma_{k_1}^{\Re,T}\mathrm{x} \cdot \gamma_{k_3}^{\Re,T}\mathrm{x}\right) + \gamma_{k_3}^{\Re,T}\left(\gamma_{k_1}^{\Re,T}\mathrm{x} \cdot \gamma_{k_2}^{\Re,T}\mathrm{x}\right) \\
&\quad - \left[\gamma_{k_1}^{\Im,T}\left(\gamma_{k_2}^{\Im,T}\mathrm{x} \cdot \gamma_{k_3}^{\Re,T}\mathrm{x}\right) + \gamma_{k_2}^{\Im,T}\left(\gamma_{k_1}^{\Im,T}\mathrm{x} \cdot \gamma_{k_3}^{\Re,T}\mathrm{x}\right) + \gamma_{k_3}^{\Re,T}\left(\gamma_{k_1}^{\Im,T}\mathrm{x} \cdot \gamma_{k_2}^{\Im,T}\mathrm{x}\right)\right] \\
&\quad - \left[\gamma_{k_1}^{\Re,T}\left(\gamma_{k_2}^{\Im,T}\mathrm{x} \cdot \gamma_{k_3}^{\Im,T}\mathrm{x}\right) + \gamma_{k_2}^{\Im,T}\left(\gamma_{k_1}^{\Re,T}\mathrm{x} \cdot \gamma_{k_3}^{\Im,T}\mathrm{x}\right) + \gamma_{k_3}^{\Im,T}\left(\gamma_{k_1}^{\Re,T}\mathrm{x} \cdot \gamma_{k_2}^{\Im,T}\mathrm{x}\right)\right] \\
&\quad - \left[\gamma_{k_1}^{\Im,T}\left(\gamma_{k_2}^{\Re,T}\mathrm{x} \cdot \gamma_{k_3}^{\Im,T}\mathrm{x}\right) + \gamma_{k_2}^{\Re,T}\left(\gamma_{k_1}^{\Im,T}\mathrm{x} \cdot \gamma_{k_3}^{\Im,T}\mathrm{x}\right) + \gamma_{k_3}^{\Im,T}\left(\gamma_{k_1}^{\Im,T}\mathrm{x} \cdot \gamma_{k_2}^{\Re,T}\mathrm{x}\right)\right]
\end{aligned} \tag{21}$$

and

$$\begin{aligned}
\frac{\partial}{\partial \mathrm{x}}\xi_k^{\Im}(\mathrm{x}) &= \frac{\partial}{\partial \mathrm{x}}\left(\gamma_{k_1}^{\Re,T}\mathrm{x} \cdot \gamma_{k_2}^{\Im,T}\mathrm{x} \cdot \gamma_{k_3}^{\Re,T}\mathrm{x}\right) + \frac{\partial}{\partial \mathrm{x}}\left(\gamma_{k_1}^{\Im,T}\mathrm{x} \cdot \gamma_{k_2}^{\Re,T}\mathrm{x} \cdot \gamma_{k_3}^{\Re,T}\mathrm{x}\right) \\
&\quad + \frac{\partial}{\partial \mathrm{x}}\left(\gamma_{k_1}^{\Re,T}\mathrm{x} \cdot \gamma_{k_2}^{\Re,T}\mathrm{x} \cdot \gamma_{k_3}^{\Im,T}\mathrm{x}\right) - \frac{\partial}{\partial \mathrm{x}}\left(\gamma_{k_1}^{\Im,T}\mathrm{x} \cdot \gamma_{k_2}^{\Im,T}\mathrm{x} \cdot \gamma_{k_3}^{\Im,T}\mathrm{x}\right) \\
&= \gamma_{k_1}^{\Re,T}\left(\gamma_{k_2}^{\Im,T}\mathrm{x} \cdot \gamma_{k_3}^{\Re,T}\mathrm{x}\right) + \gamma_{k_2}^{\Im,T}\left(\gamma_{k_1}^{\Re,T}\mathrm{x} \cdot \gamma_{k_3}^{\Re,T}\mathrm{x}\right) + \gamma_{k_3}^{\Re,T}\left(\gamma_{k_1}^{\Re,T}\mathrm{x} \cdot \gamma_{k_2}^{\Im,T}\mathrm{x}\right) \\
&\quad + \gamma_{k_1}^{\Im,T}\left(\gamma_{k_2}^{\Re,T}\mathrm{x} \cdot \gamma_{k_3}^{\Re,T}\mathrm{x}\right) + \gamma_{k_2}^{\Re,T}\left(\gamma_{k_1}^{\Im,T}\mathrm{x} \cdot \gamma_{k_3}^{\Re,T}\mathrm{x}\right) + \gamma_{k_3}^{\Re,T}\left(\gamma_{k_1}^{\Im,T}\mathrm{x} \cdot \gamma_{k_2}^{\Re,T}\mathrm{x}\right) \\
&\quad + \gamma_{k_1}^{\Re,T}\left(\gamma_{k_2}^{\Re,T}\mathrm{x} \cdot \gamma_{k_3}^{\Im,T}\mathrm{x}\right) + \gamma_{k_2}^{\Re,T}\left(\gamma_{k_1}^{\Re,T}\mathrm{x} \cdot \gamma_{k_3}^{\Im,T}\mathrm{x}\right) + \gamma_{k_3}^{\Im,T}\left(\gamma_{k_1}^{\Re,T}\mathrm{x} \cdot \gamma_{k_2}^{\Re,T}\mathrm{x}\right) \\
&\quad - \left[\gamma_{k_1}^{\Im,T}\left(\gamma_{k_2}^{\Im,T}\mathrm{x} \cdot \gamma_{k_3}^{\Im,T}\mathrm{x}\right) + \gamma_{k_2}^{\Im,T}\left(\gamma_{k_1}^{\Im,T}\mathrm{x} \cdot \gamma_{k_3}^{\Im,T}\mathrm{x}\right) + \gamma_{k_3}^{\Im,T}\left(\gamma_{k_1}^{\Im,T}\mathrm{x} \cdot \gamma_{k_2}^{\Im,T}\mathrm{x}\right)\right].
\end{aligned} \tag{22}$$

Refer to the method `taylorExpansion` in Code 3 (line 200-278).

6 Results

We will now finally look on our implementation's results using both synthetic and real measurements from the dataset website described in Section 5.1.

6.1 Synthetic Measurements

In order to reasonably compare our results to the original implementation of CHIRP we show the results from K. L. Bouman et al. [3] in Figure 7.
Building on [3] we are using the same array configuration realistic for the EHT pointed towards the black hole inside Messier 87, which is provided in their supplemental material [2], on generated source images with a total flux density of 1 jansky,

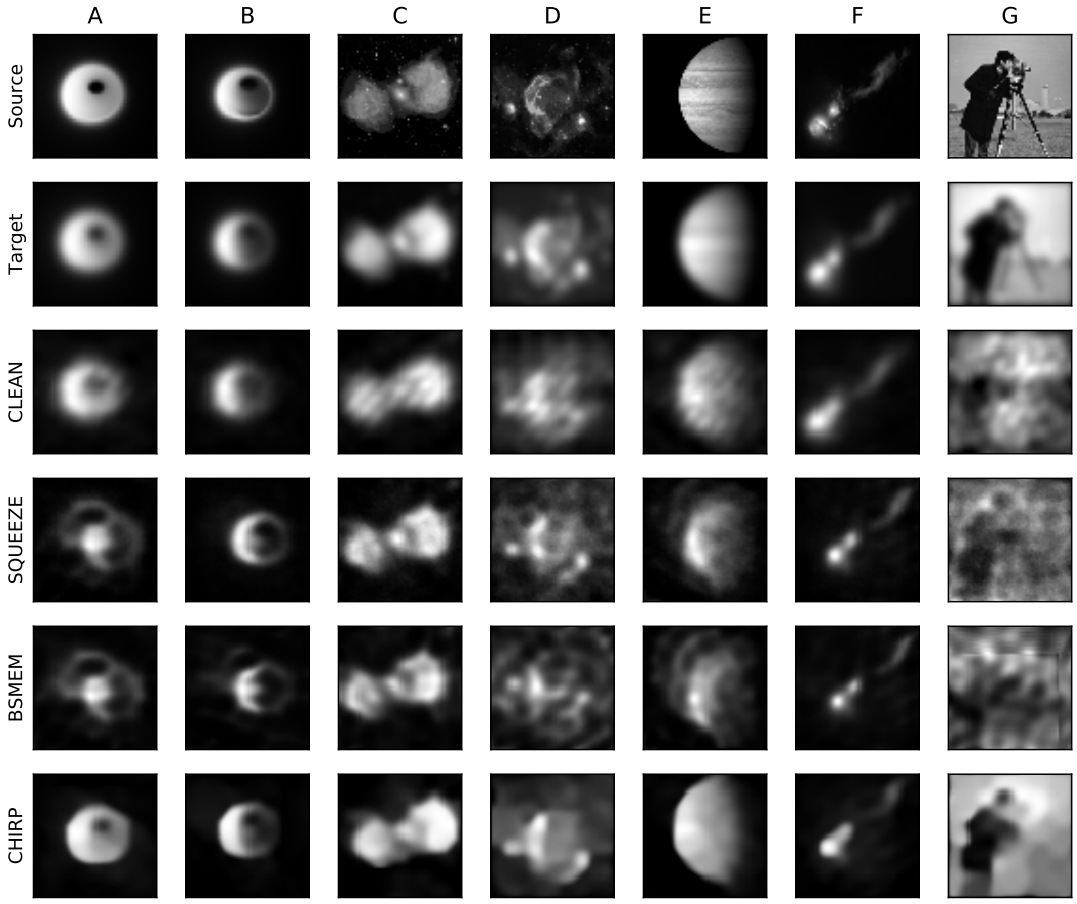

Figure 7: Normalized results from [3]. Comparison of CHIRP to the state-of-art methods CLEAN, SQUEEZE and BSMEM described in Section 3 of a variety of modelled black hole (A-B), celestial (C-F) and natural (G) source images.

a 71.1 μas field of view (FOV) for the black hole images 183.82 μas FOV for the celestial and natural images. Refer to Figure 5 for the corresponding frequency plane coverage.
Although the source images are given through the dataset, the ground truth emission's maximum resolution we can reconstruct is restricted by the longest baseline of the array configuration. Further, the target images in Figure 7 show the ground truth filtered to the maximum resolution what is possible to reliably reconstruct from the measurements [3]. For the purpose of evaluating our reconstruction we make use of the filtered ground truth images from [2] normalized to the corresponding source images. We align our results to these images using phase correlation and then evaluate the mean squared error (MSE) metric and the structural similarity (SSIM) index [19]. Refer to the method `CHIRP.score` in Code 2 (line 135-160).
In Figure 9 and 8 we demonstrate our implementation's results corresponding to the source images A-G from Figure 7. Notice the similiarities to the original results. For the choice of the data term weighting parameter λ from Equation (10) we ran our reconstruction on different values, as shown in Figure 10, and eventually set the default value to $\lambda = 0.0001$.
In addition, we examine the noise sensitivity corresponding to measurements at different total flux densities in Figure 11. Decreasing the total flux density, i.e. the sum of pixel intensities results in higher noise [3].

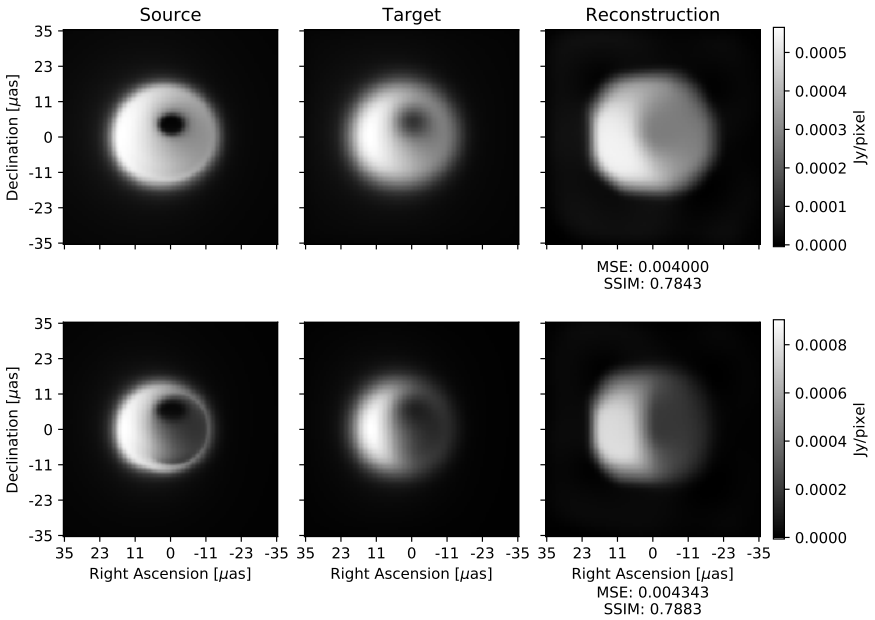

Figure 8: Our implementation's results corresponding to the black hole source images [4] A and B from Figure 7 with a 71.1 μas FOV and 1 jansky flux.

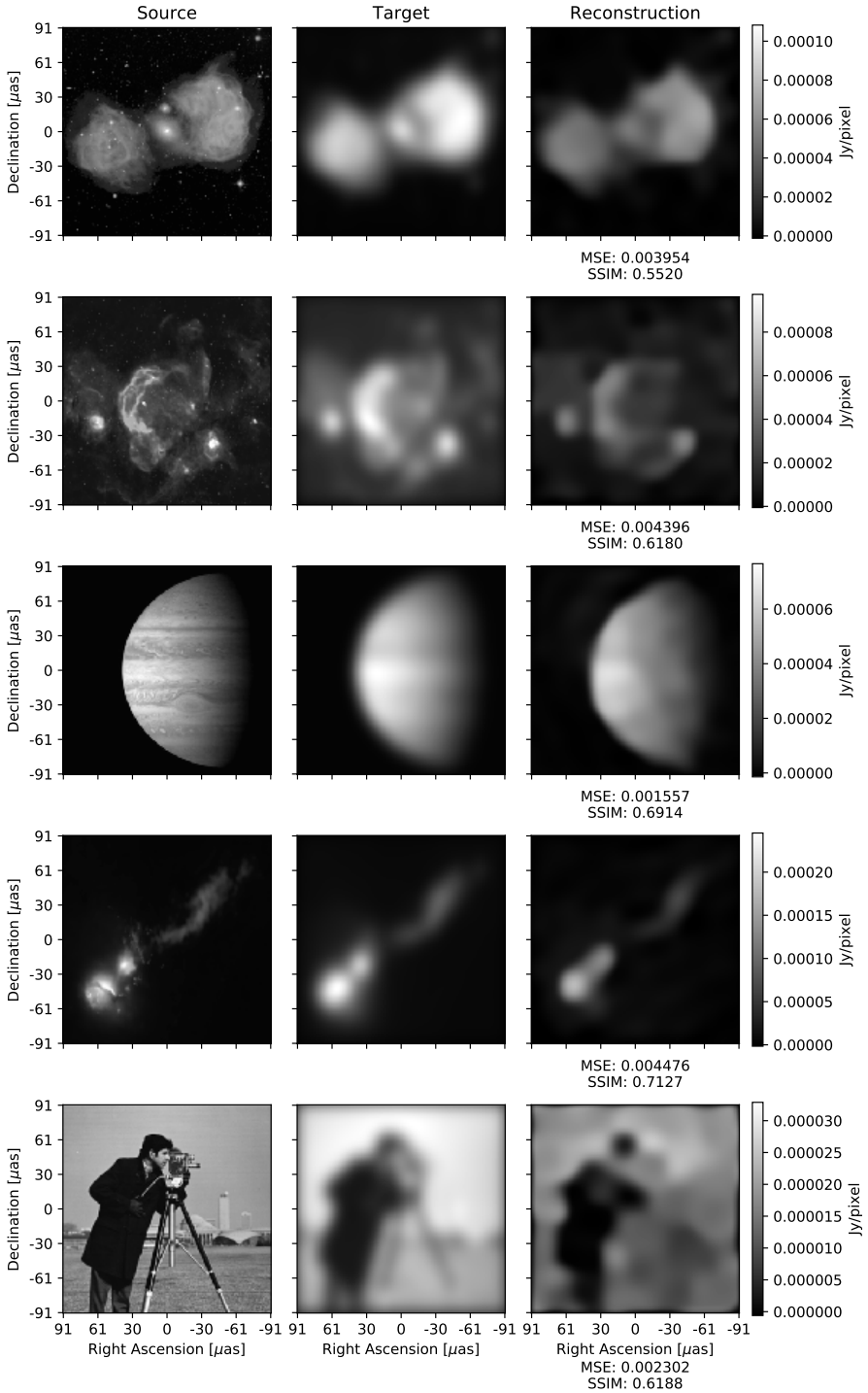

Figure 9: Our implementation's results corresponding to the celestial and natural source images C-G from Figure 7 with a 183.82 μas FOV and 1 jansky flux.

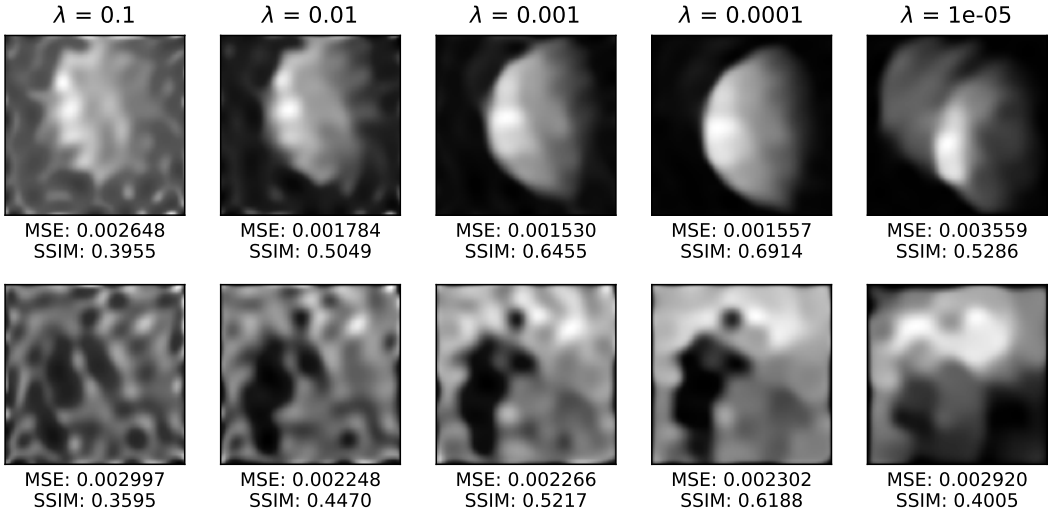

Figure 10: Effect of reconstructing the source images E and G from Figure 7 for varying weightings λ of the data term.

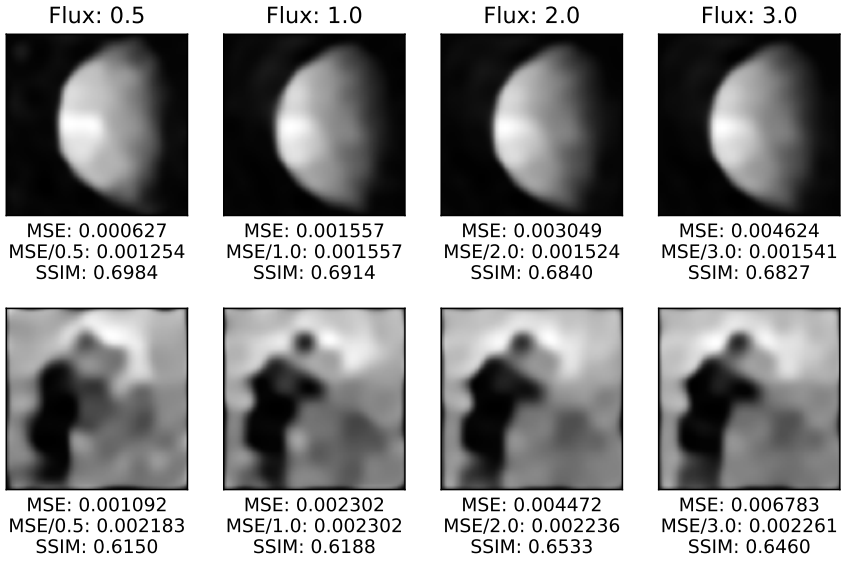

Figure 11: Effect of reconstructing the source images E and G from Figure 7 for varying noise levels. Note that due to the source images' difference in the sum of pixel intensities we additionally normalize each MSE to reansonably compare its value.

6.2 Real Measurements

In Figure 12 we show our implementation's results on three quasars 3C111, 3C446 and OJ287 using real VLBI data from the VLBA-BU-BLAZAR program [9]. As addressed in Section 5.1 there are no ground truth images to evaluate the corresponding reconstruction on, only reference images produced by the BU Blazar Group.

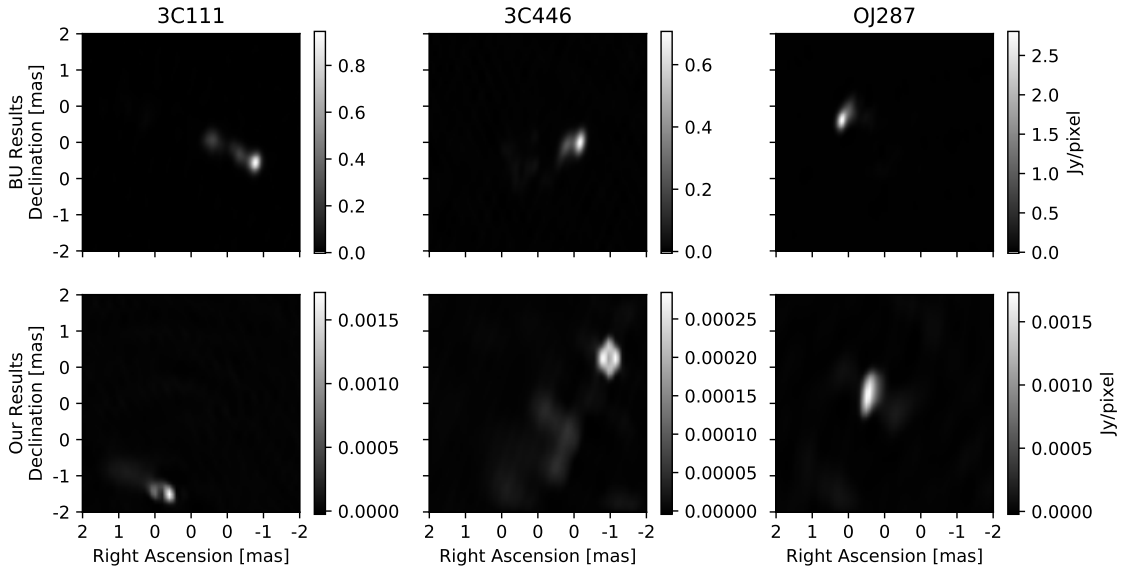

Figure 12: Our implementation's results in comparison to the BU Group's using CLEAN self-calibration[6] with a FOV of 5.12 μas.

[6]This study makes use of 43 GHz VLBA data from the VLBA-BU Blazar Monitoring Program (VLBA-BU-BLAZAR; `http://www.bu.edu/blazars/VLBAproject.html` (visited on 15. October 2019)), funded by NASA through the Fermi Guest Investigator Program. The VLBA is an instrument of the National Radio Astronomy Observatory. The National Radio Astronomy Observatory is a facility of the National Science Foundation operated by Associated Universities, Inc.

7 Conclusion

In this bachelor thesis we have extensively described the inverse problem to reconstruct the underlying image using sparse VLBI measurements and worked through K. L. Bouman et al.'s image reconstruction algorithm continuous high-resolution image reconstruction using patch priors.
Thanks to K. L. Bouman et al.'s website we got an easy access to a large, realistic VLBI dataset and thus were able to try out own implementations of VLBI imaging methods. In order to successfully implement CHIRP we had to gain a sufficient knowledge of its physical background and clarify many fine details omitted in their publication paper. During our implementation's development we had to face various difficulties, like specifying a concrete initialization, which may certainly be worthy of improvement, and additionally work on the unknown data term's weighting parameter, which seems to complicate reconstruction on blind VLBI image reconstruction. However, our results' similarities to those of K. L. Bouman et al. show a fundamental functionality of our final implementation of CHIRP.

References

[1] F. Baron, J. D. Monnier, and B. Kloppenborg. A novel image reconstruction software for optical/infrared interferometry. *Proceedings of SPIE - The International Society for Optical Engineering*, 7734, 2010.

[2] K. L. Bouman, M. D. Johnson, D. Zoran, V. L. Fish, S. S. Doeleman, and W. T. Freeman. Computational Imaging for VLBI Image Reconstruction Supplemental Material.

[3] K. L. Bouman, M. D. Johnson, D. Zoran, V. L. Fish, S. S. Doeleman, and W. T. Freeman. Computational Imaging for VLBI Image Reconstruction. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, page 913, Mar 2016.

[4] A. E. Broderick, V. L. Fish, S. S. Doeleman, and A. Loeb. Evidence for Low Black Hole Spin and Physically Motivated Accretion Models from Millimeter-VLBI Observations of Sagittarius A*. *Astrophysical Journal*, 735(2):110, Jul 2011.

[5] D. F. Buscher. *Direct Maximum-Entropy Image Reconstruction from the Bispectrum*, pages 91–93. Springer, 1994.

[6] A. A. Chael, M. D. Johnson, K. L. Bouman, L. L. Blackburn, K. Akiyama, and R. Narayan. Interferometric imaging directly with closure phases and closure amplitudes. *The Astrophysical Journal*, 857(1):23, Apr 2018.

[7] Event Horizon Telescope Collaboration, K. Akiyama, A. Alberdi, W. Alef, K. Asada, R. Azulay, A.-K. Baczko, D. Ball, M. Baloković, J. Barrett, and et al. First M87 Event Horizon Telescope Results. II. Array and Instrumentation. *Astrophysical Journal, Letters*, 875(1):L2, Apr 2019.

[8] J. A. Högbom. Aperture Synthesis with a Non-Regular Distribution of Interferometer Baselines. *Astronomy and Astrophysics, Supplement*, 1974.

[9] S. G. Jorstad, A. P. Marscher, M. L. Lister, A. M. Stirling, T. V. Cawthorne, W. K. Gear, J. L. Gómez, J. A. Stevens, P. S. Smith, J. R. Forster, and E. I. Robson. Polarimetric Observations of 15 Active Galactic Nuclei at High Frequencies: Jet Kinematics from Bimonthly Monitoring with the Very Long Baseline Array. *Astronomical Journal*, 130(4):1418–1465, Oct 2005.

[10] T. P. Krichbaum, D. A. Graham, M. Bremer, W. Alef, A. Witzel, J. A. Zensus, and A. Eckart. Sub-Milliarcsecond Imaging of Sgr A* and M 87. *Journal of Physics: Conference Series*, 54:328–334, dec 2006.

[11] F. R. S. Lord Rayleigh. Investigations in optics, with special reference to the spectroscope. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 8(49):261–274, 1879.

[12] R.-S. Lu, A. E. Broderick, F. Baron, J. D. Monnier, V. L. Fish, S. S. Doeleman, and V. Pankratius. Imaging the supermassive black hole shadow and jet base of m87 with the event horizon telescope. *The Astrophysical Journal*, 788(2):120, may 2014.

[13] F. Malbet, W. Cotton, G. Duvert, P. Lawson, A. Chiavassa, J. Young, F. Baron, D. Buscher, S. Rengaswamy, B. Kloppenborg, M. Vannier, and L. Mugnier. The 2010 interferometric imaging beauty contest. 7734, 2010.

[14] J. D. Monnier and R. J. Allen. *Radio and Optical Interferometry: Basic Observing Techniques and Data Analysis*, page 325. 2013.

[15] T. A. Pauls, J. S. Young, W. D. Cotton, and J. D. Monnier. A Data Exchange Standard for Optical (Visible/IR) Interferometry. *Publications of the Astronomical Society of the Pacific*, 117(837):1255–1262, nov 2005.

[16] G. B. Taylor, C. L. Carilli, and R. A. Perley, editors. *Synthesis Imaging in Radio Astronomy II*, volume 180 of *Astronomical Society of the Pacific Conference Series*, 1999.

[17] É. Thiébaut and J. Young. Principles of image reconstruction in optical interferometry: tutorial. *J. Opt. Soc. Am. A*, 34(6):904–923, Jun 2017.

[18] A. R. Thompson, J. M. Moran, and G. W. Swenson, Jr. *Interferometry and Synthesis in Radio Astronomy*. Springer, 2017.

[19] Z. Wang, A. C. Bovik, H. R. Sheikh, E. P. Simoncelli, et al. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[20] D. Zoran and Y. Weiss. From Learning Models of Natural Image Patches to Whole Image Restoration. In *2011 International Conference on Computer Vision*, pages 479–486, Nov 2011.

A Code

Code 1: example.py

```
import numpy as np
import matplotlib.pyplot as plt
from astropy.io import fits
import oifits as oi
import vlbi

# flux: 1.0
oidata = oi.open('./data/syntheticData/staticNatural/UVdata/' +\
                 'natural-03-10.oifits')
fitsdata = fits.open('./data/syntheticData/staticNatural/' +\
                     'targetImgs/natural-03-10.fits')[0]

src_im = fitsdata.data
target_im = plt.imread('./data/syntheticData/targets/natural-03.pgm')[:,:,0]
target_im = target_im.astype(np.float64)
target_im /= np.max(target_im)
target_im *= np.max(src_im)

fov = np.abs(fitsdata.header['CDELT1']) * np.pi/180 * fitsdata.header['NAXIS1']
print('fov:      %0.2f mas \n      %0.4e rad'
      %(fov * 1e6 * 3600 * 180 / np.pi, fov))
naxis = fitsdata.header['NAXIS1']

chirp = vlbi.CHIRP(oidata, fov, naxis)
res = chirp.reconstruct(display=True)

chirp.showResult(ref_im=src_im)
mse, ssim = chirp.score(target_im)
print('MSE:  %0.6f \nSSIM: %0.6f' %(mse, ssim))
```

Code 2: vlbi.py

```
import numpy as np
import matplotlib.pyplot as plt
import scipy.io as io
import vlbi_utils as utils
from astropy.io import fits
from skimage.feature import register_translation
from scipy.ndimage import fourier_shift
from skimage.measure import compare_ssim

class CHIRP(object):

    def __init__(self, oidata, fov, naxis, **kwargs):

        """
        Inputs:
        - oidata: OIFITS data format extraced by Paul Boley's OIFITS module
                  containing visibilities and bispectrum measurements.
        - fov: Field of view in radians (float).
        - naxis: Size of the source image (int).
        Optional arguments:
        - pulse: Pulse function used in the continuous image representation.
                Default is the two-dimensional triangular pulse.
        - pulse_ft: Closed-form Fourier transform of the pulse function used
                    in the continuous image representation. Default is the
                    Fourier transform of the two-dimensional triangular pulse.
        - gmm: A dictionary containing the mixture components of a pre-trained
              Gaussian Mixture Model. Default a GMM trained on natural images
              taken from https://github.com/achael/eht-imaging/tree/960a79557b
              4de7f2776bcfa1aef2c37cea487ab7.
        - patch_size: Size of the patches (int) for the EPLL. Default is 8.
        - betas: A sorted list in ascending order of weighting parameters (int)
                for half quadratic splitting. Default is
                [1, 4, 8, 16, 32, 64, 128, 256, 512].
        - scales : A sorted list in ascending order of the number of pulse
                   functions (int) in the continuous image representation.
                   Default is [20, 23, 26, 29, 34, 38, 43, 49, 56, 64].
        """
```

```
38
39          self.data = utils.getData(oidata)
40          self.fov = fov
41          self.naxis = naxis
42          self.history = {}
43          self.res = None
44          self.__fig, self.__axs, self.__cbar = None, 2*[None], None
45
46          pulse = lambda x, y, delta: \
47                  (np.maximum(1 - np.abs(x/delta), 0) / delta) *\
48                  (np.maximum(1 - np.abs(y/delta), 0) / delta)
49          self.pulse = kwargs.pop('pulse', pulse)
50          pulse_ft = lambda x, y, delta: \
51                     np.sinc(x * delta)**2 *\
52                     np.sinc(y * delta)**2
53          self.pulse_ft = kwargs.pop('pulse_ft', pulse_ft)
54          pdata = io.loadmat('naturalPrior.mat')
55          gmm = {}
56          gmm['n_components'] = pdata['nmodels'][0][0]
57          gmm['weights'] = pdata['mixweights'].flatten()
58          gmm['covs'] = pdata['covs']
59          gmm['means'] = pdata['means']
60          self.gmm = kwargs.pop('gmm', gmm)
61          self.patch_size = kwargs.pop('patch_size', 8)
62          self.betas = kwargs.pop('betas',
63                                  [1, 4, 8, 16, 32, 64, 128, 256, 512])
64          self.scales = kwargs.pop('scales',
65                                   [20, 23, 26, 29, 34, 39, 44, 50, 56, 64])
66
67          if len(kwargs) > 0:
68              raise Exception('Unrecognized arguments.')
69
70
71      def reconstruct(self, lam = 0.0001, display=False):
72
73          """
74          Reconstruct the image given the bispectrum measurements.
75
76          Inputs:
77          - lambda: Weighting parameter of the data term (float). Default
78                    is 0.0001.
```

```
79          - display: Flag for displaying partial solutions. Default is False.
80          Outputs:
81          - x: A numpy array of shape (N,N) contaning the reconstructed image
82               coefficients.
83          """
84
85          self.history = {}
86
87          x = utils.initImage(self.data, self.fov, self.scales[0], self.pulse_ft)
88
89          if display:
90              plt.ion()
91              plt.show()
92              self.__fig, self.__axs = plt.subplots(1, 2)
93              plt.subplots_adjust(wspace=-0.1)
94              self.__display(x, np.zeros_like(x), 'Initialization')
95
96          self.history['init'] = x
97
98          for scale in self.scales:
99
100             x = utils.upscaleImage(x, self.fov, scale, self.pulse)
101
102             gammas = (utils.ftVectors(self.data['bi_uvcoord1'], self.fov,
103                                       scale, self.pulse_ft),
104                       utils.ftVectors(self.data['bi_uvcoord2'], self.fov,
105                                       scale, self.pulse_ft),
106                       utils.ftVectors(self.data['bi_uvcoord3'], self.fov,
107                                       scale, self.pulse_ft))
108
109             for beta in self.betas:
110
111                 # (a) solve for Z while keeping x constant
112                 Z = utils.mostLikelyPatches(x, beta, self.data,
113                                             self.patch_size, self.gmm)
114
115                 # (b) solve for x while keeping Z constant
116                 x = utils.taylorExpansion(x, Z, beta, self.data, gammas,
117                                           self.patch_size, lam=lam)
118
```

```
119                 if display:
120                     self.__axs[0].clear()
121                     self.__axs[1].clear()
122                     self.__display(x, Z, 'Scale: ' + str(scale) + '\n' +\
123                                          r'$\beta$: ' + str(beta))
124             self.history[scale] = x
125 
126         if display:
127             plt.ioff()
128 
129         self.res = np.rot90(utils.upscaleImage(x, self.fov,
130                                                self.naxis, self.pulse),2)
131         return self.res
132 
133     def score(self, ref_im):
134 
135         """
136         Align the reconstruction using phase correlation and calculate the MSE
137         and SSIM.
138 
139         Inputs:
140         - ref_im: A numpy array of shape (N,N) contaning a reference image.
141         Outputs:
142         - mse: Mean squared error (float).
143         - ssim: Structural similarity index (float).
144         """
145 
146         if isinstance(self.res, type(None)):
147             raise Exception('Result is not yet aviable.')
148 
149         shift = register_translation(ref_im, self.res)[0]
150         shifted_res = fourier_shift(np.fft.fft2(self.res), shift)
151         shifted_res = np.real(np.fft.ifft2(shifted_res))
152 
153         mse = np.linalg.norm(shifted_res - ref_im)
154         drange = np.max(shifted_res) - np.min(shifted_res)
155         ssim = compare_ssim(ref_im, shifted_res, data_range=drange)
156 
157         return mse, ssim
158 
```

```
159     def saveFits(self, filename):
160
161         """
162         Save the reconstruction in FITS.
163
164         Inputs:
165         - filename: Name for the file to be created (String).
166         """
167
168         if isinstance(self.res, type(None)):
169             raise Exception('Result is not yet aviable.')
170
171         header = fits.Header()
172         header['NAXIS1'] = self.naxis
173         header['NAXIS2'] = self.naxis
174         header['CTYPE1'] = 'RA---SIN'
175         header['CTYPE2'] = 'DEC--SIN'
176         header['CDELT1'] = - self.fov/(np.pi/180 * self.naxis)
177         header['CDELT2'] = self.fov/(np.pi/180 * self.naxis)
178         header['BUNIT'] = 'JY/PIXEL'
179
180         hdu = fits.PrimaryHDU(self.res, header=header)
181         hdulist = fits.HDUList([hdu])
182         hdulist.writeto(filename, overwrite=True)
183
184         print("Saved as '%s'." %(filename))
185
186     def showResult(self, ref_im=None):
187
188         """
189         Show the reconstructed image (in comparison to the source image).
190
191         Inputs:
192         - ref_im: A numpy array of shape (N,N) contaning a reference image.
193                   Default is None.
194         """
195
196         if isinstance(self.res, type(None)):
197             raise Exception('Result is not yet aviable.')
198
```

```
199         fov_mas = self.fov * 1e6 * 3600 * 180 / np.pi
200         ticks = np.linspace(0, self.naxis-1, 7)
201         ticklabels = np.linspace(fov_mas/2, -fov_mas/2, 7, dtype=int)
202 
203         if not isinstance(ref_im, type(None)):
204 
205             fig, axs = plt.subplots(1, 2)
206             plt.subplots_adjust(wspace=-0.1)
207 
208 
209             minVal = np.min([ref_im,self.res])
210             maxVal = np.max([ref_im,self.res])
211 
212             im = axs[0].imshow(ref_im, cmap='gray', vmin=minVal, vmax=maxVal)
213             temp = fig.colorbar(im, ax=axs[0], shrink=0.575, label='Jy/pixel')
214             temp.remove()
215             axs[0].set_xticks(ticks)
216             axs[0].set_xticklabels(ticklabels)
217             axs[0].set_yticks(ticks)
218             axs[0].set_yticklabels(ticklabels)
219             axs[0].set_title('Reference')
220             axs[0].set_xlabel('Right Ascension [$\mu$as]')
221             axs[0].set_ylabel('Declination [$\mu$as]')
222 
223             im = axs[1].imshow(self.res, cmap='gray', vmin=minVal, vmax=maxVal)
224             axs[1].set_title('Reconstruction')
225             axs[1].set_xticks(ticks)
226             axs[1].set_xticklabels(ticklabels)
227             axs[1].set_yticks([])
228             axs[1].set_xlabel('Right Ascension [$\mu$as]')
229             fig.colorbar(im, ax=axs[1], shrink=0.575, label='Jy/pixel')
230 
231             plt.show()
232 
233         else:
234 
235             fig, ax = plt.subplots(1, 1)
236 
237             im = plt.imshow(self.res, cmap='gray')
238             ax.set_xticks(ticks)
```

```
239             ax.set_xticklabels(ticklabels)
240             ax.set_yticks(ticks)
241             ax.set_yticklabels(ticklabels)
242             ax.set_title('Reconstruction')
243             ax.set_xlabel('Right Ascension [$\mu$as]')
244             ax.set_ylabel('Declination [$\mu$as]')
245             fig.colorbar(im, ax=ax, label='Jy/pixel')
246 
247             plt.show()
248 
249     def __display(self, x, Z, title):
250 
251         """
252         Auxiliary method for displaying partial solutions.
253         """
254 
255         minVal = np.min([x,Z])
256         maxVal = np.max([x,Z])
257 
258         scale = x.shape[0]
259         fov_muas = self.fov * 1e6 * 3600 * 180 / np.pi
260         ticks = np.linspace(0, scale-1, 7)
261         ticklabels = np.linspace(fov_muas/2, -fov_muas/2, 7, dtype=int)
262 
263         self.__fig.suptitle(title)
264         im = self.__axs[0].imshow(np.rot90(Z,2), cmap='gray',
265                                   vmin=minVal, vmax=maxVal)
266         temp = self.__fig.colorbar(im, ax=self.__axs[0], shrink=0.575,
267                                    label='Jy/pixel')
268         temp.remove()
269         self.__axs[0].set_xticks(ticks)
270         self.__axs[0].set_xticklabels(ticklabels)
271         self.__axs[0].set_yticks(ticks)
272         self.__axs[0].set_yticklabels(ticklabels)
273         self.__axs[0].set_title('Combined \n Patch Priors')
274         self.__axs[0].set_xlabel('Right Ascension [$\mu$as]')
275         self.__axs[0].set_ylabel('Declination [$\mu$as]')
276 
277         im = self.__axs[1].imshow(np.rot90(x,2), cmap='gray',
278                                   vmin=minVal, vmax=maxVal)
```

```
279         self.__axs[1].set_title('Image Coefficients')
280         self.__axs[1].set_xticks(ticks)
281         self.__axs[1].set_xticklabels(ticklabels)
282         self.__axs[1].set_yticks([])
283         self.__axs[1].set_xlabel('Right Ascension [$\mu$as]')
284 
285         if(self.__cbar):
286             self.__cbar.remove()
287         self.__cbar = self.__fig.colorbar(im, ax=self.__axs[1], shrink=0.575,
288                                           label='Jy/pixel')
289 
290         self.__fig.canvas.draw()
291         plt.pause(0.001)
```

Code 3: vlbi_utils.py

```
1  import numpy as np
2  
3  def ftVectors(uvcoord, fov, scale, pulse_ft):
4  
5      """
6      Calculate the Fourier transform vectors from Eq.(7).
7  
8      Inputs:
9      - uvcoord: A numpy array of shape (M,2) contaning M uv-coordinates.
10     - fov: Field of view (float).
11     - scale: The number of pulse functions used in the continuous image
12              representation (int).
13     - pulse_ft: Closed-form Fourier transform of the pulse function used in
14                 the continuous image representation.
15     Outputs:
16     - gammas: A numpy array of shape (M,scale**2) containing the Fourier
17               transform vectors from Eq.(7) corresponding to the
18               uv-coordinates.
19     """
20 
21     delta = fov/scale
22     shift = delta * np.arange(scale) + delta/2 - fov/2
23     # Eq.(7)
```

```
24      gammas = np.array([pulse_ft(u, v, delta) *\
25                         np.outer(np.exp(-2j*np.pi * v*shift),
26                                  np.exp(-2j*np.pi * u*shift)
27                                  ).flatten(order='F')
28                         for u, v in uvcoord])
29
30      return gammas
31
32  def initImage(data, fov, scale, pulse_ft):
33
34      """
35      Initialize  the image coefficients as the dirty image.
36
37      Inputs:
38      - data: A dictionary containing the necessary data from OIFITS.
39      - fov: Field of view (float).
40      - scale: The number of pulse functions used in the continuous image
41               representation (int).
42      - pulse_ft: Closed-form Fourier transform of the pulse function used in
43                  the continuous image representation.
44      Outputs:
45      - x: A numpy array of shape (scale,scale) contaning image coefficients.
46      """
47
48      gamma = ftVectors(data['vis_uvcoord'], fov, scale, pulse_ft)
49      gamma_conj = np.conjugate(gamma).T
50      u = data['vis_uvcoord'][:,0]
51      v = data['vis_uvcoord'][:,1]
52      denom = pulse_ft(u,v,fov/scale) * pulse_ft(-u,-v,fov/scale) * scale**2
53      # Eq.(15)
54      x = np.real(gamma_conj @ (data['vis']/denom))
55      x = np.reshape(x, (scale,scale), order='F')
56
57      x = x / np.sum(x)
58      x = np.rot90(x,2)   # (init is upside down)
59
60      return x
61
62  def upscaleImage(x, fov, new_scale, pulse):
63
```

```
64      """
65      Calculate the discretized continuous image representation.
66
67      Inputs:
68      - x: A numpy array of shape (scale,scale) contaning the image coefficients.
69      - fov: Field of view (float).
70      - new_scale: Size of the continuous image representation's discretization
71                   (int).
72      - pulse: Pulse function used in the continuous image representation.
73      Outputs:
74      - im: A numpy array of shape (new_scale,new_scale) contaning the
75            (discretized) continuous image representation.
76      """
77
78      scale = x.shape[0]
79      delta = fov/scale
80      new_delta = fov/new_scale
81
82      # Eq.(6)
83      cir = lambda l, m: np.sum([[x[j,i] *\
84                                  pulse(l - (delta*i + delta/2 - fov/2),
85                                        m - (delta*j + delta/2 - fov/2), delta)
86                                  for j in range(scale)]
87                                  for i in range(scale)], axis=(0,1))
88
89      new_shift = new_delta * np.arange(new_scale) + new_delta/2 - fov/2
90      ll, mm = np.meshgrid(new_shift, new_shift)
91      im = cir(ll,mm)       # measured in Jy
92
93      im *= new_delta**2    # measured in Jy / pixel
94
95      return im
96
97  def mostLikelyPatches(x, beta, data, patch_size, gmm):
98
99      """
100     Optimize the cost function in terms of Z to create an image prior from
101     generated patch priors.
102
103     Inputs:
```

```
104     - x: A numpy array of shape (N,N) contaning the current image coefficients.
105     - gmm: A dictionary containing the GMM's parameters.
106     - patch_size: Size of the patches (int).
107     - beta: half quadratic splitting's weighting parameter (int).
108     Outputs:
109     - Z: A numpy array of shape (N,N) contaning the image prior.
110     """
111
112     # normalize the image coefficients (inspired by patch_prior.py
113     # line 50-53 of https://github.com/achael/eht-imaging/blob/
114     # 960a79557b4de7f2776bcfa1aef2c37cea487ab7/patch_prior.py)
115     minVal = np.min(x)
116     maxVal = np.max(x)
117     x = (x - minVal) / maxVal
118
119     xPad = np.pad(x , (patch_size-1, patch_size-1), 'constant')
120     patches = extractPatches(xPad, patch_size)
121     meanPatches =  np.mean(patches, axis=0)
122     patches -= np.tile(meanPatches, (patch_size**2, 1))
123
124     # determine the mixture components
125     w_scores = np.zeros((gmm['n_components'], patches.shape[1]))
126     for i in range(gmm['n_components']):
127         # add noise to covariance matrices (inspired by patch_prior.py
128         # line 116 of https://github.com/achael/eht-imaging/blob/
129         # 960a79557b4de7f2776bcfa1aef2c37cea487ab7/patch_prior.py)
130         G = np.linalg.cholesky(gmm['covs'][:,:,i] +\
131                                1/beta * np.eye(patch_size**2))
132         # Eq.(16)
133         w_scores[i,:] = np.log(gmm['weights'][i])            -\
134                         patches.shape[0]/2 * np.log(2*np.pi) -\
135                         np.log(np.prod(np.diagonal(G)))      -\
136                         1/2 * np.sum((np.linalg.inv(G) @ patches)**2 , axis=0)
137     j_star = w_scores.argmax(axis = 0)
138
139     # create a set of auxiliary patches
140     Zs = np.zeros(patches.shape)
141     for j in range(gmm['n_components']):
142         inds = np.where(j_star == j)[0]
143         # Eq.(17)
```

```
144         Zs[:,inds] = np.linalg.inv(gmm['covs'][:,:,j] +\
145                                    1/beta * np.eye(patch_size**2)) @\
146                     (gmm['covs'][:,:,j] @ patches[:,inds] +\
147                      1/beta * np.tile(gmm['means'][:,j], (len(inds), 1)).T)
148     Zs += np.tile(meanPatches, (patch_size**2, 1))
149 
150     # combine the auxiliary patches into a prior image
151     domainPad = np.pad(np.ones_like(x),
152                        (patch_size-1, patch_size-1),
153                        'constant')
154     inds_patches = extractPatches(np.reshape(range(np.prod(domainPad.shape)),
155                                              domainPad.shape, order='F'),
156                                   patch_size)
157     stackedZ = np.bincount(inds_patches.flatten(order='F').astype(np.int),
158                            weights=Zs.flatten(order='F'))
159     Z_vec = np.extract(domainPad.flatten(order='F'), stackedZ)
160     Z_vec /= (patch_size**2)
161 
162     Z_vec = (Z_vec * maxVal) + minVal
163     # set negative entries to zero (inspired by patch_prior.py
164     # line 78 of https://github.com/achael/eht-imaging/blob/
165     # 960a79557b4de7f2776bcfa1aef2c37cea487ab7/patch_prior.py)
166     Z_vec[Z_vec < 0] = 0
167     # normalize prior to maximum absolute visbility (inspired by
168     # linearize_energy.py line 17-18 of https://github.com/achael/eht-imaging/
169     # blob/960a79557b4de7f2776bcfa1aef2c37cea487ab7/linearize_energy.py
170     Z_vec = np.max(np.abs(data['vis'])) * Z_vec / np.sum(Z_vec)
171 
172     Z = np.reshape(Z_vec, x.shape, order='F')
173 
174     return Z
175 
176 def extractPatches(im, patch_size):
177 
178     """
179     Extract all the possible patches from an image.
180 
181     Inputs:
182     - im: A numpy array of shape (N,N) contaning an image with scalar data.
183     - patch_size: Size of the patches to be extracted (int).
```

```
184     Outputs:
185     - patches: A numpy array of shape (P,M) containing the extracted
186                vectorized patches in columns.
187     """
188 
189     N = im.shape[0]
190     P = patch_size**2
191     M = (N - patch_size + 1)**2
192     patches = np.zeros((P, M))
193     for i in range(M):
194         row = i % (N - patch_size + 1)
195         col = i // (N - patch_size + 1)
196         patches[:,i] = np.reshape(im[row:row+patch_size, col:col+patch_size],
197                                   P, order='F')
198     return patches
199 
200 def taylorExpansion(x, Z, beta, data, gammas, patch_size, lam):
201 
202     """
203     Optimize the cost function in terms of x performing a second order
204     Taylor expansion.
205 
206     Inputs:
207     - x: A numpy array of shape (N,N) contaning the current image coefficients.
208     - Z: A numpy array of shape (N,N) contaning the current image prior.
209     - beta: half quadratic splitting's weighting parameter (int).
210     - data: A dictionary containing the necessary data from OIFITS.
211     - gammas: A triple of numpy arrays containing the Fourier transform vectors
212               from Eq. 5 corresponding to the uv-coordinates from the
213               bispectrum measurements.
214     - patch_size: Size of the patches (int).
215     - lam: Weighting parameter of the data term (float).
216     Outputs:
217     - x: A numpy array of shape (N,N) contaning the new image coefficients.
218     """
219 
220     x_vec = x.flatten(order='F')
221     Z_vec = Z.flatten(order='F')
222 
223     gr1, gr2, gr3 = np.real(gammas)
```

```
224     gi1, gi2, gi3 = np.imag(gammas)
225     gr1x, gr2x, gr3x = gr1 @ x_vec, gr2 @ x_vec, gr3 @ x_vec
226     gi1x, gi2x, gi3x = gi1 @ x_vec, gi2 @ x_vec, gi3 @ x_vec
227     # Eq.(20)
228     rXi = gr1x*gr2x*gr3x - gi1x*gi2x*gr3x -\
229           gr1x*gi2x*gi3x - gi1x*gr2x*gi3x
230     iXi = gr1x*gi2x*gr3x + gi1x*gr2x*gr3x +\
231           gr1x*gr2x*gi3x - gi1x*gi2x*gi3x
232     rMeas = np.real(data['bi'])
233     iMeas = np.imag(data['bi'])
234
235     # Eq.(21)
236     rA = gr1 * (gr2x * gr3x)[:,np.newaxis]  +\
237          gr2 * (gr1x * gr3x)[:,np.newaxis]  +\
238          gr3 * (gr1x * gr2x)[:,np.newaxis]  -\
239         (gi1 * (gi2x * gr3x)[:,np.newaxis]  +\
240          gi2 * (gi1x * gr3x)[:,np.newaxis]  +\
241          gr3 * (gi1x * gi2x)[:,np.newaxis]) -\
242         (gr1 * (gi2x * gi3x)[:,np.newaxis]  +\
243          gi2 * (gr1x * gi3x)[:,np.newaxis]  +\
244          gi3 * (gr1x * gi2x)[:,np.newaxis]) -\
245         (gi1 * (gr2x * gi3x)[:,np.newaxis]  +\
246          gr2 * (gi1x * gi3x)[:,np.newaxis]  +\
247          gi3 * (gi1x * gr2x)[:,np.newaxis])
248     # Eq.(22)
249     iA = gr1 * (gi2x * gr3x)[:,np.newaxis]  +\
250          gi2 * (gr1x * gr3x)[:,np.newaxis]  +\
251          gr3 * (gr1x * gi2x)[:,np.newaxis]  +\
252          gi1 * (gr2x * gr3x)[:,np.newaxis]  +\
253          gr2 * (gi1x * gr3x)[:,np.newaxis]  +\
254          gr3 * (gi1x * gr2x)[:,np.newaxis]  +\
255          gr1 * (gr2x * gi3x)[:,np.newaxis]  +\
256          gr2 * (gr1x * gi3x)[:,np.newaxis]  +\
257          gi3 * (gr1x * gr2x)[:,np.newaxis]  -\
258         (gi1 * (gi2x * gi3x)[:,np.newaxis]  +\
259          gi2 * (gi1x * gi3x)[:,np.newaxis]  +\
260          gi3 * (gi1x * gi2x)[:,np.newaxis])
261     rB = rXi - rA @ x_vec
262     iB = iXi - iA @ x_vec
263
```

```
264     factor = 3/data['num_tele']/(data['bi_amperr']**2)
265 
266     sum1 = (factor[:,np.newaxis] * rA).T @ rA +\
267            (factor[:,np.newaxis] * iA).T @ iA
268     sum2 = (factor[:,np.newaxis] * rA).T @ (rB - rMeas) +\
269            (factor[:,np.newaxis] * iA).T @ (iB - iMeas)
270 
271     # Eq.(19)
272     new_x_vec = np.linalg.solve(
273                 lam * sum1 + beta * (patch_size**2) * np.eye(len(Z_vec)),
274                 - lam * sum2 + beta * (patch_size**2) * Z_vec)
275 
276     new_x = np.reshape(new_x_vec, x.shape, order='F')
277 
278     return new_x
279 
280 def getData(oidata):
281 
282     """
283     Extract the necessary data from OIFITS.
284 
285     Inputs:
286         - oidata: OIFITS data format extraced by Paul Boley's OIFITS module
287                   containing visibilities and bispectrum measurements.
288     Ouputs:
289         - data: A dictionary containing the necessary data from OIFITS.
290     """
291 
292     data = {}
293 
294     wav = oidata.wavelength['WAVELENGTH_NAME'].eff_wave[0]
295     data['vis_uvcoord'] = np.array([[v.ucoord/wav,
296                                      v.vcoord/wav] for v in oidata.vis])
297     vis_amp = np.array([v.visamp[0] for v in oidata.vis])
298     vis_phi = np.array([v.visphi[0] for v in oidata.vis]) * np.pi/180
299     data['vis'] = vis_amp * np.exp(1j * vis_phi)
300     data['vis_amperr'] = np.array([v.visamperr[0] for v in oidata.vis])
301 
302     data['bi_uvcoord1'] = np.array([[t3.u1coord/wav, t3.v1coord/wav]
303                                     for t3 in oidata.t3])
```

```
304     data['bi_uvcoord2'] = np.array([[t3.u2coord/wav, t3.v2coord/wav]
305                                     for t3 in oidata.t3])
306     data['bi_uvcoord3'] = np.array([[-(t3.u1coord+t3.u2coord)/wav,
307                                      -(t3.v1coord+t3.v2coord)/wav]
308                                     for t3 in oidata.t3])
309
310     bi_amp = np.array([t3.t3amp[0] for t3 in oidata.t3])
311     bi_phi = np.array([t3.t3phi[0] for t3 in oidata.t3]) * np.pi/180
312     data['bi'] = bi_amp * np.exp(1j * bi_phi)
313     data['bi_amperr'] = np.array([t3.t3amperr[0] for t3 in oidata.t3])
314
315     stations = {}
316     for k in oidata.vis:
317         t = k.timeobs.time()
318         time = (t.hour * 60 + t.minute) * 60 + t.second
319         stations[time] = []
320     times = []
321     for k in oidata.vis:
322         t = k.timeobs.time()
323         time = (t.hour * 60 + t.minute) * 60 + t.second
324         stations[time].append(k.station[0].sta_name)
325         stations[time].append(k.station[1].sta_name)
326         times.append((t.hour * 60 + t.minute) * 60 + t.second)
327     num_tele = []
328     for t3 in oidata.t3:
329         t = t3.timeobs.time()
330         time = (t.hour * 60 + t.minute) * 60 + t.second
331         unique_stations = np.unique(stations[time])
332         num_tele.append(len(unique_stations))
333     data['num_tele'] = np.array(num_tele)
334
335     return data
```