Master's Thesis

Development of 3D Spectrometry using Compressive Sensing

Jonas Ravelid

June 2018

LUNDS UNIVERSITET

Supervised by Andreas Ehn & Elias Kristensson

Division of Combustion Physics, Lund University

Abstract

Multi-spectral imaging is a powerful approach in spectral analysis of objects with spatial details. However, such approaches are most often time consuming and experimentally complex since both spectral and spatial information in two dimensions are to be resolved. Presented here is an alternative approach to three-dimensional spectroscopy, utilizing the Compressive Sensing methodology to treat a digital image as an underrepresented system of equations. By utilizing the properties of random bases and enforcing sparsity onto a spectral two-dimensional scene using a binary mask the system of equations can be solved numerically and thus allow spectral separation of two-dimensional spectral scenes in a post-processing script. To achieve this goal, an approach is presented in this report where 1) a spectrometer setup was modified and utilized, 2) an image analysis approach based on Compressive Sensing was developed 3) the method was employed and analyzed for several different data sets and 4) an objective measure of reconstruction quality of spectral components was created and evaluated. The outcome of this work clearly show that Compressive Imaging has potential to become a widely used tool in multi-spectral imaging and that it, despite some drawbacks, is a big step forward from conventional three-dimensional spectrometry.

Contents

1 List of Abbreviations

2D - two-dimensional

3D - three-dimensional

CCD - Charge-Coupled Device

CS - Compressive Sensing

TwIST - Two-step Iterative Shrinkage/Thresholding Algorithm

GoR - Goodness of Reconstruction

2 Introduction

2.1 Fundamental Spectrometry

When mixing two colors, blue and yellow, the human eye interprets that as green because the individual pigments are too small for the eye to distinguish. Consequently, humans cannot distinguish between objects that are green and objects that are both blue and yellow but with very small pigments. One way to overcome this limitation is using spectroscopy.

Spectroscopy is an important tool in many different fields of science. Spectroscopic measurements can give a lot of information about a system, such as the contents of an unknown gas [1], the temperature of a flame [2] or the internal structure of an atom or molecule [3].

All spectroscopic measurements build on the principle of correlating spectral information to something more interpretable. The most basic spectrometers components are a narrow entrance slit, followed by a dispersive element and then a detector screen of some form. A depiction of a very basic spectrometer is displayed in Figure 1. Light from an object entering the spectrometer is thus confined to one spatial dimension by the slit and then divided into separate components that are projected onto different locations of the detector screen by the dispersive element, taking spectral separation from the spectral domain to the spatial. In the case of the example with colors used earlier, this would allow distinguishing between green and yellow-and-blue objects, as the green would be displayed at one spatial location while blue and yellow would be displayed at other locations.

This does, however, limit the object being studied. Since all commercially available photo-detectors are one- or two-dimensional (1D, 2D), and one dimension is allocated to spectral information, spatial resolution is reduced to at most one dimension. Furthermore, if the studied object is at least 2D, spatial information of the object is lost unless the entrance slit is removed. If the entrance slit is removed however, the incoming light is no longer restricted to one spatial dimension and many different spectral components will be projected onto the same location on the detector. It removes the 1:1 correlation between spectral wavelength and spatial position on the detector screen since that property is enforced by the narrow entrance slit. Consequently, simultaneously extracting spectral and 2D spatial information from a system is difficult, albeit not impossible.

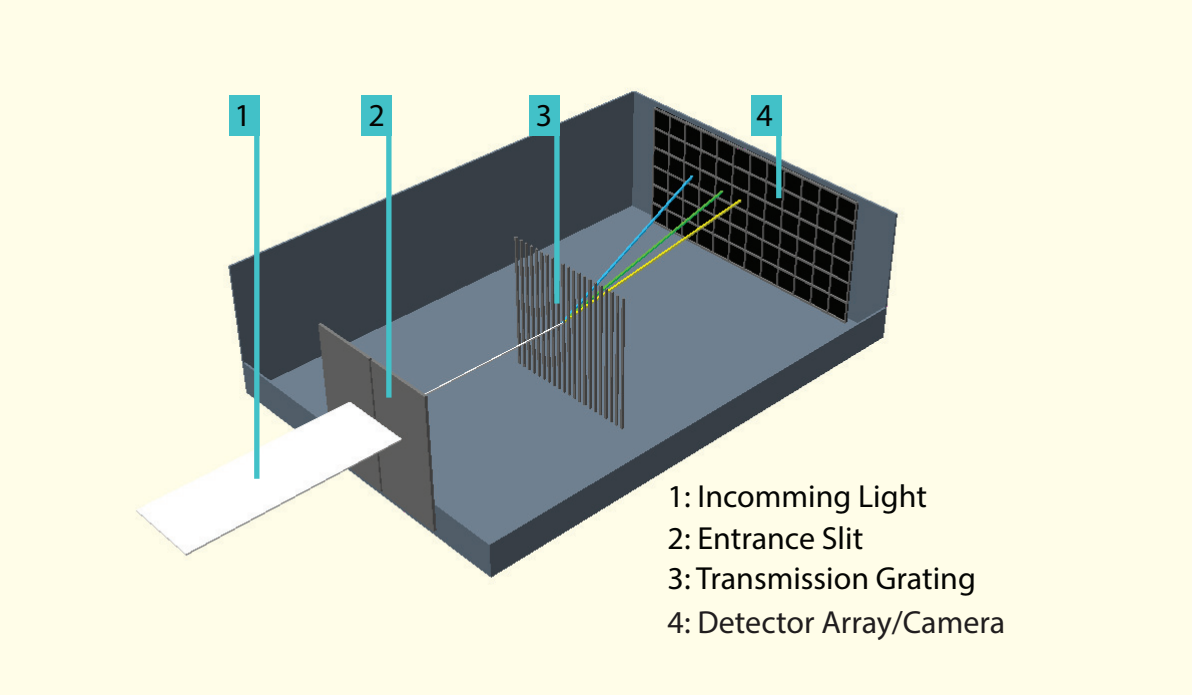

Figure 1: A very basic spectroscopic system with an entrance slit (2), a dispersive element (3) and a detector screen (4). As the incoming light enters the spectrometer the entrance slit restricts the incoming light along the horizontal (dispersion) axis before the light is dispersed. A camera is often used as a detector screen, meaning the spectral components are detected by a pixelated array of photo-detectors.

2.2 Conventional 3D Spectrometry

Three-dimensional (3D) spectroscopy, with two spatially and one spectrally resolved dimension, is technically possible but the method has limitations that reduces its viability [3]. It requires spatially scanning the object along the axis of dispersion to preserve the 1:1 correlation mentioned earlier. This scanning routine makes the process time-consuming, restricting the method to use in stationary scenarios. Furthermore, is does not allow for snap-shot measurements (i.e. measurement times short enough for the change in spatial or spectral information to be negligible), thus excluding very interesting spectral analysis in transient conditions, for example combustions with preserved spatial information and species localization in turbulent environments.

One of the main areas where snap-shot 3D spectroscopy would be of use is in multi-species imaging in combustion processes. One method used to probe such systems is Planar Laser-Induced Fluorescence (PLIF) where the wavelength of the laser is tuned

to a transition of the target molecule. The fluorescence signal is then images with a camera, yielding a spatial distribution of the targeted species in the 2D sheet. Doing this for multiple species simultaneously requires an advanced detection set-up using multiple detectors, as the signal from different species must be separated before detection, that increases in complexity as the number of species increases. A new methodology could circumvent the need for an advanced detection set-up if the spectral components could be separated using only a single detector. Furthermore, such a detector system would not increase in complexity as the number of species probed increases.

Another method that could benefit from a new approach is Raman-spectroscopy, where it could allow spectral resolution of Raman-measurements using pulsed lasers on 2D scenes and thus allow species localization. Furthermore, it could help in the removal of the significantly stronger Rayleigh-signal thus improving the level of quality in Raman-spectroscopy.

Thus, a spectroscopic system capable of snap-shot measurements in two spatial and one spectral dimension simultaneously, could potentially allow for the extraction of spatio-spectral information not previously available. That, however, requires reinstating the 1:1 correlation that makes spectral resolution possible, without re-introducing the entrance slit.

2.3 Compressive Sensing

The following chapter contains a brief description of Compressive Sensing (CS) and it can restore the 1:1 correlation in a spectral measurement of a 2D object without restricting the light spatially. First off, it is important to understand how a modern camera functions in the process of detecting light, and how that can be expressed mathematically.

Most modern cameras utilize a 2D array of pixelated solid-state detectors (CCD-chip), shown in Fig 1. Each individual detector pixel is capable of detecting photons through electron-hole pair generation, resulting in a charge density flux proportional to the number of incoming photons in that pixel. Thus, the measurement intensity signal (image) f detected by a 2D CCD chip is well represented by a 2D matrix, where each matrix element holds the detected intensity in its corresponding pixel:

$$f = \begin{bmatrix} I_{1,1} & I_{2,1} & \dots & I_{x,1} \\ I_{1,2} & I_{2,2} & \dots & I_{x,2} \\ \vdots & \vdots & \ddots & \vdots \\ I_{1,y} & I_{2,y} & \dots & I_{x,y} \end{bmatrix}$$

In the case of 3D spectroscopy, the total image projected onto the detector screen is a set of 2D spectral components shifted by the grating but still superposed on each other. This means that each pixel detects light from several different components and is unable to distinguish between them:

$$I_{x,y} = \sum_{n=1}^{L} I_{x,y,n} \tag{1}$$

where L is the total number of spectral components in the image and $I_{x,y,n}$ is the intensity of the corresponding spectral component n at detector pixel (x, y). All intensity components $I_{x,y,n}$ form a spatio-spectral data-cube with three dimensions, two spatial and one spectral, as shown in Fig 2. When this cube is incident on a detector screen it is projected onto two dimensions, effectively summing the data cube along n, thus removing information on spectral origin, as described in Eq 1. This results in a detected signal S that is the two-dimensional projection of f onto the (x, y)-plane. To re-establish the 1:1 relation between spectral component and detector position, this sum must be separated into its constituent components, despite this loss of information.

It is shown in [4] that Eq 1 can be separated into its components by introducing two constraints: sparsity and incoherence. Both are presented more in-depth in [5], only a short summary is given here.

Sparsity describes a signal f, measured in some basis ϕ, that can be expanded in some space with basis ψ using only a small number of components, a sparse representation. Examples include Fourier Transform Spectroscopy, a spectroscopic method that resolves a spectrum by expanding it in the Fourier domain rather than the spatial domain [3], and data compression in computers [6].

Incoherence is a measure of the lack of correlation between elements in two bases ψ and ϕ. The importance of incoherence comes from the fact that it, in part, determines the level of reconstruction possible for a signal f that is detected in the basis ϕ and sparse

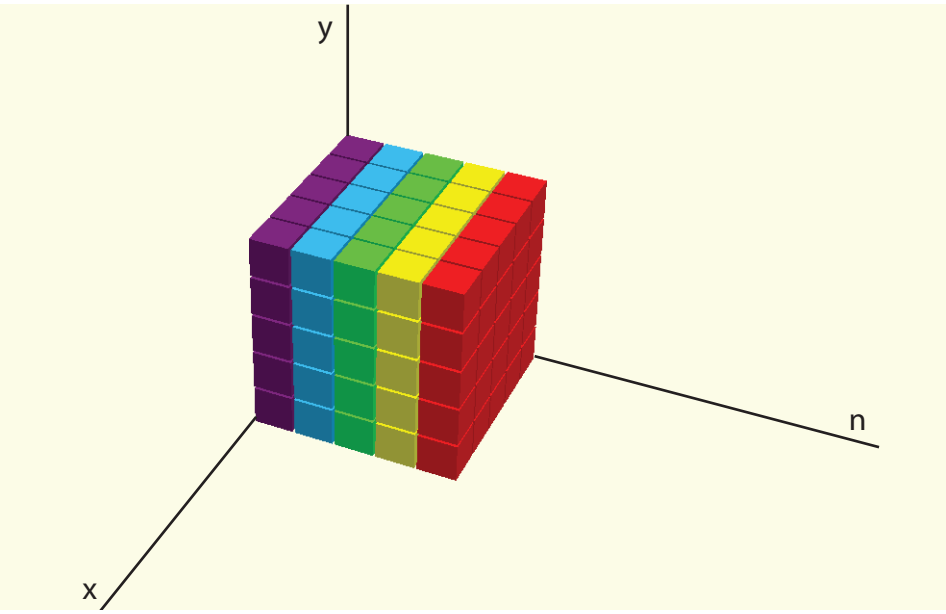

Figure 2: A visual representation of a 3D spatio-spectral data-cube. x and y denote the two spatial dimensions and n denotes the spectral dimension. If this cube was to be projected onto a detector each row along n would collapse, as described in Eq 1. The aim with CS is to be able to separate the different spectral components and their corresponding intensities after such a collapse.

in the basis ψ [5]. The same paper also states that random matrices (such as a binary matrix sampled from a uniform distribution) are sufficiently incoherent with any fixed basis to allow complete reconstruction, making them an interesting point of investigation.

Using these two concepts, it can be shown that a signal f, as well as all its components, can be completely recovered with "overwhelming probability", assuming the detection basis ϕ and the representation basis ψ are sufficiently incoherent and f is sparse in ψ. Moreover, this separation can be achieved by an algorithm in linear time[1], suggesting the method is practically viable [5] [7].

In the case of this project, introducing these constraints suggest that individual spectral components I_n can be separated and recovered in the measured 2D measurement signal S without needing to completely recreate f. This requires the signal to be made sufficiently sparse, using a basis that is incoherent with the detection (spatial) basis. Succeeding in this would mean that the 1:1 correlation between position and wavelength is re-established, allowing spectrally resolved 2D scenes.

[1]Meaning that the expected time the algorithm needs to solve the problem scales linearly with the size of the problem

As previously mentioned, a uniformly random binary matrix is incoherent with any other fixed basis [5]. A uniformly random binary matrix is a 2D matrix where each element is either 0 or 1, determined randomly with equal probability, displayed in Fig 3. Thus, by letting the data-cube pass through such a matrix mask it is made sparse in a basis incoherent with the detection basis, enabling separation of the spectral components.

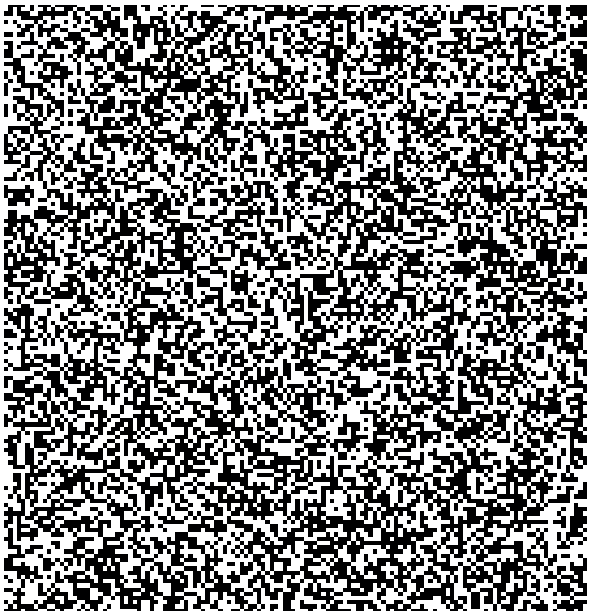

Figure 3: An example of a binary matrix mask. Each pixel (x, y) is valued at either 1 (white/translucent) or 0 (black) with 50% probability. In a 3D spectroscopic measurement, the incoming light would pass through such a mask before entering the dispersive medium, making the incoming light signal sparse by blocking any light that hits a black patch and letting through any light that hits a translucent patch.

Random matrices have been proved to be sufficiently incoherent with any fixed basis and should thus be able to create high quality reconstructions, as shown in [8]. Since incoherence is a non-binary quality, pseudo-random matrices also enable reconstruction of images, as shown in [9]. This implies that a mask can be designed to increase the reconstruction quality or provide additional information of the detection system, as long as it is still sufficiently random to be considered incoherent. These specially designed, pseudo-random masks will hereafter be referred to as "pseudo-random" while binary masks sampled from a uniformly random distribution will be referred to as "uniformly random".

This concept is tested through the design of a mask with no periodic horizontally

repeating pattern, shown in Fig 4, in an attempt to guarantee that no two overlapping spectral components are identical. This pattern is achieved using a segment featuring a short diagonal line. This segment is repeated multiple times but each horizontal repetition has a different random and unique vertical start location from the previous. Because of this, any two spectral components with a spacing equal to the spacing of this pattern can still be differentiated. Furthermore, any other two spectral components can be differentiated since the segment itself is horizontally unique. This pattern should be incoherent with any image not mainly consisting of diagonal stripes and thus allow apt image reconstruction.

This design also limits the spectral resolution inherent in the mask to the width of a pixel represented by one matrix element in I, as no two pixels are directly adjacent horizontally.

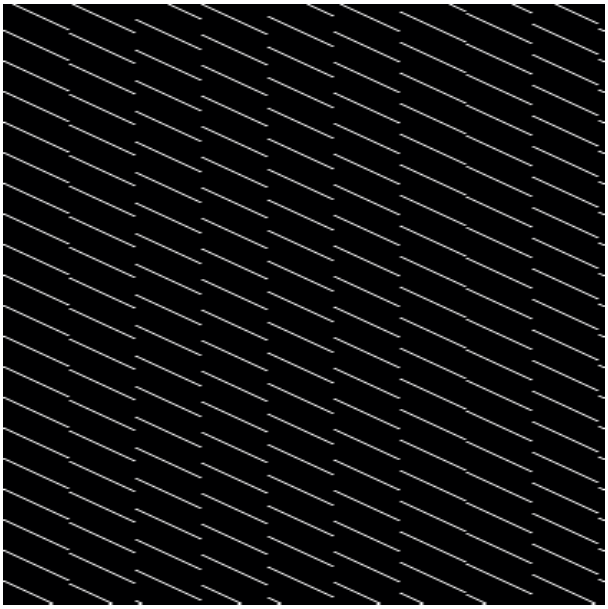

Figure 4: An example of a pseudo-random mask. Each pixel (x, y) is valued at either 1 (white/translucent) or 0 (black). The pattern is designed so that there is no periodic repetition horizontally, guaranteeing complete separability of spectral components.

2.4 Signal Recovery

It has already been stated that this problem can be solved in linear time using a post-processing algorithm [5]. The task of the post-processing algorithm can be understood intuitively by breaking down what happens to the spectral data-cube:

Light that is to be spectrally resolved approaches the system as a 3D spatio-spectral data-cube f (same f as previously). It passes through a random mask H and is thus encoded as:

$$H \cdot f$$

It then passes through a dispersive element $D(\lambda)$ where each spectral component is shifted an amount depending on its wavelength, resulting in a sheered encoded data-cube mathematically described by:

$$D(\lambda) \cdot H \cdot f$$

Finally, the data-cube is projected onto the 2D detector screen but due to the spatial shift introduced by the dispersive element each spectral component detected in an area (x_1, x_2, y_1, y_2) has a slightly different encoding. Fig 5 shows the process of marking the incoming data-cube using a 5x5 uniformly random matrix and then dispersing it before it hits a detector screen, generating a two-dimensional signal S. It also shows how each spectral component in a fixed area has a different encoding.

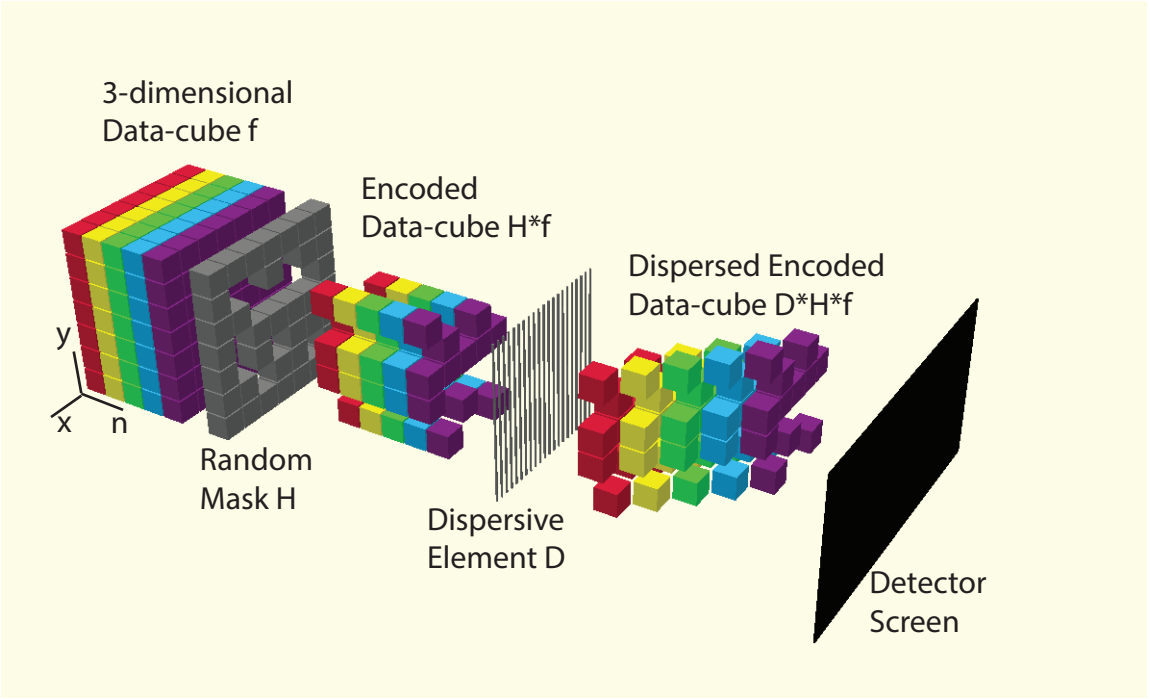

Figure 5: The evolution of the three-dimensional data-cube f, traveling from left to right. When it passes the mask it is spatially encoded. The dispersive element sheers the cube, meaning that different spectral components that hit the detector screen in the same area have different coding, making them separable.

The detected signal S is consequently a 2D projection of the 3D data-cube $D \cdot H \cdot f$. Assuming the pattern of the random mask H is known, [10] shows that this information is sufficient to identify and separate the spectral components making up the original data-cube by numerically solving the underrepresented system of equations given by

$$f = H^{-1} \cdot D^{-1} \cdot S$$

[10] also suggest several algorithms suited for performing this signal recovery. A Two-Step Iterative Shrinkage/Thresholding Algorithm (TwIST) [11] was chosen for this project, as several sources reported good reconstruction ability (see [10] [8]). Furthermore, a black-box MATLAB script was available for the algorithm [12] which negated the need to write a complete script. This still required a surrounding framework to be built in Matlab, handling loading and preparation of images as well as receiving and interpreting the reconstructions.

2.5 Developing the Foundation for 3D Spectrometry

The aim of this project is thus to create a spectroscopic system capable of spectrally resolving 2D scenes, using the CS methodology, from recording images to evaluating reconstructions.

This means constructing a laboratory set-up like the one previously presented, with a spectrometer and a set of random masks on a translation stage. It also includes the writing of a Matlab framework within which input and output for the TwIST algorithm are easily handled, evaluated and displayed. This system is mainly thought to be utilized in Raman-spectroscopy, meaning that spatial localization of spectral components, finding what spectral components originate where in a scene, is prioritized over species identification through the development of a spectrum. Consequently, a definition of "goodness" must be created and briefly evaluated, in order to objectively rate the reconstructions of the spectral scenes.

In both examples presented previously, PLIF and Raman-spectroscopy, the aim is species localization in a 2D scene. Spatial resolution is not imperative in all current methods; the main goal of species localization is more concerned with existence of, and if so approximate spatial spread and location of, species of interest. Thus, the project is more

aimed at achieving good spectral separation of components than level of detail, especially when it comes to snap-shot capability as the quality of reconstruction is expected to suffer due to the low number of measurements.

This also means that mask design, setup automation and a more rigorous and thorough investigation into the definition of goodness lies outside the scope of this project and should be left for future research.

The project is meant to be a first investigation into the possibility of implementing 3D spectroscopy as a tool in the field of spectroscopy and is as such meant to lay a foundation for future deeper research into specific areas of interest within the methodology.

3 Experimental Setup

Two different experimental arrangements were set-up. An overview of the second set-up is also displayed in Fig 6. The first version of the experimental set-up showed the importance of precise alignment, as the quality of the reconstructions created by the first set-up suffered heavily that was caused by a slight mis-alignment. This led to a second, updated version with a more meticulous alignment process.

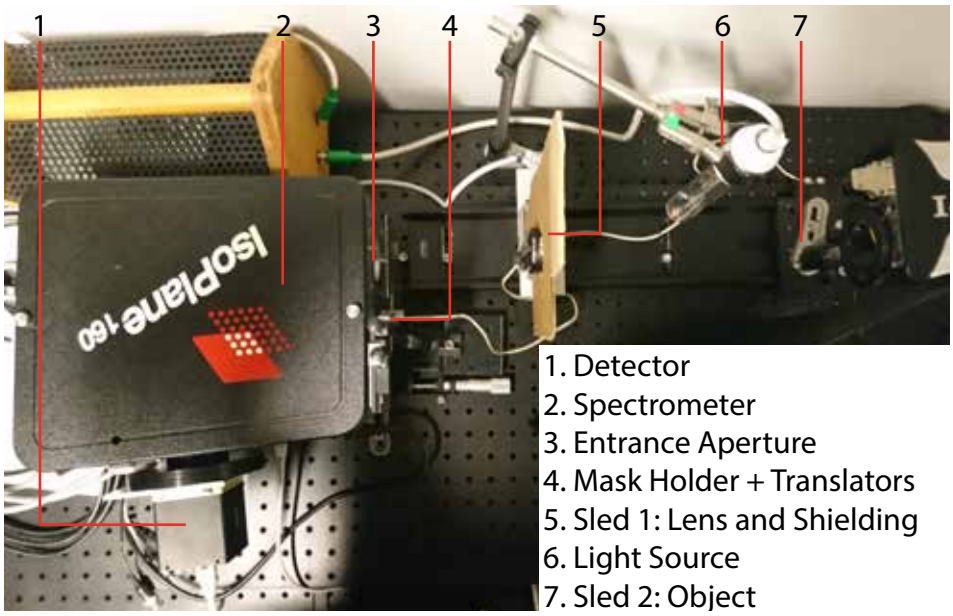

Figure 6: A top-down overview of the experimental set-up, with important parts numbered.

Equipment-wise, the two set-ups were identical. They consisted of an IsoPlane 160 spectrometer from Princeton Instruments with three gratings; 150 groves/mm, 1200 groves/mm and 2400 groves/mm, with an ImperX B4822 camera mounted as the detector. The entrance slit was removed from the spectrometer, leaving a rectangular entrance aperture measuring 10 by 3 mm, as displayed in Fig 7.

A mask holder was placed on a high-precision piezoelectric linear translator (Q-521 Q-Motion Miniature Linear Stage) from PI. The translator stage was used to control and systematically alter the position of the mask during measurements. This translator was in turn fastened on a mount with three degrees of translational freedom, used for switching masks in-between measurements, just in front of the inlet of the spectrometer to allow the mask to be imaged onto the detector screen. This mount, with the mask holder, is displayed in Fig 8.

Figure 7: A close-up of the entrance to the spectrometer, to show the 2D entrance aperture.

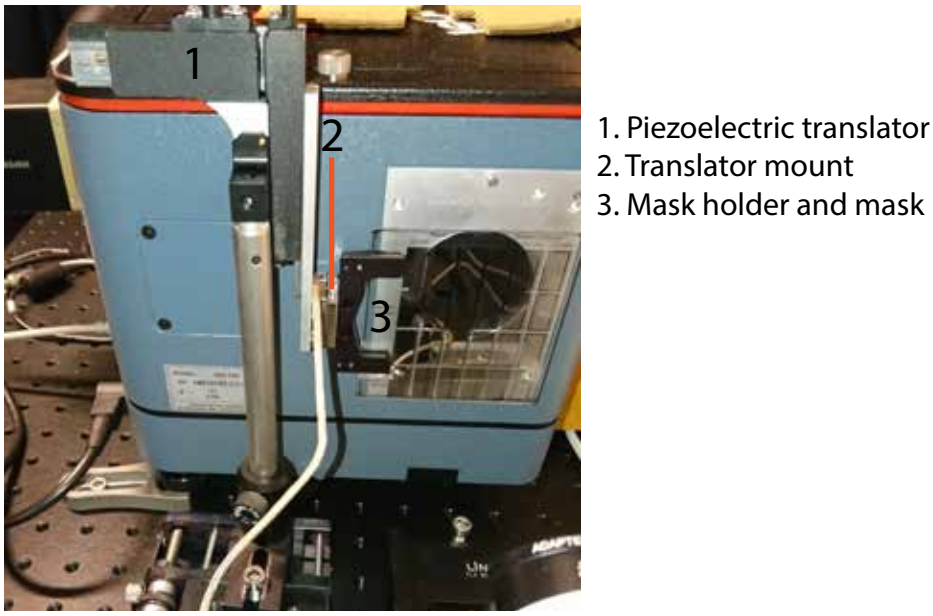

Figure 8: A close-up of the mask holder and its translation mount.

A sliding rail was placed along the optical axis in front of the spectrometer, with two sleds. The first sled was equipped with a lens with a focal length of 150 mm, as well as an iris, while the other sled contained the object to be imaged, highlighted in Fig 6.

The light source was a Philips Zinc spectral lamp [13], placed to illuminate the object while being shielded from the spectrometer to reduce stray light, also highlighted in Fig 6.

The object imaged was a printed Lorem Ipsum text [14], displayed in Fig 9. The part of the text that was projected onto the detector, and thus the target of the reconstructions, are highlighted by a red box.

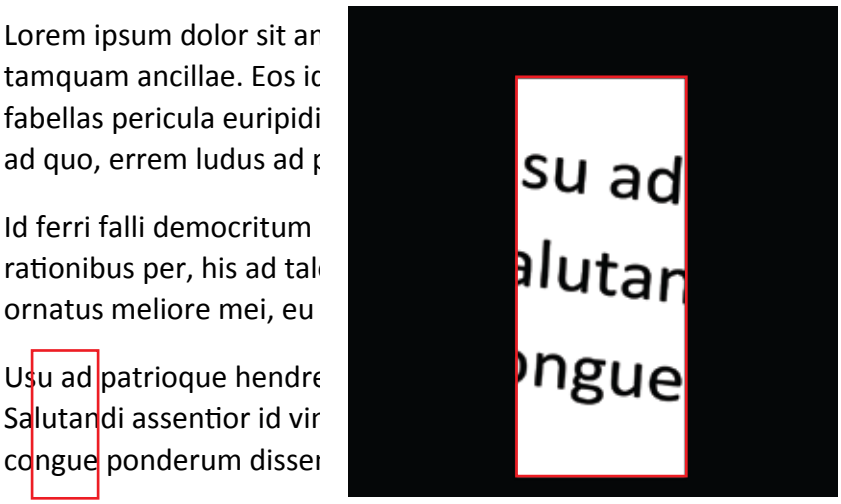

Figure 9: Left: The text used as the object in the measurements. The red box shows the part that was projected onto the detector screen. Right: The text within the red box is displayed as it was detected.

The masks used are displayed in Fig 10. The three leftmost columns are pseudo-random masks and the rightmost three are uniformly random. Each row signifies the pixel size of the masks of that row and are tied to the pixel size, and thus binning, of the detector. The difference between the three columns of pseudo-random masks is the horizontal repetition rate of the masks, which is displayed at the top of each column. The corresponding number for the universally random masks show how much of the masks are transparent.

The spectral band observed in all measurements was 354 nm to 737 nm, using the 150 groves/mm grating.

The mis-alignment of the first set-up mentioned previously was a mis-alignment between detector, grating and mask. The TwIST framework assumes that the 2D array-structures of the mask and the detector screen aligns. Furthermore, it assumed that the direction of dispersion of the spectrometer grating is parallel to one of the sides of the 2D arrays, so

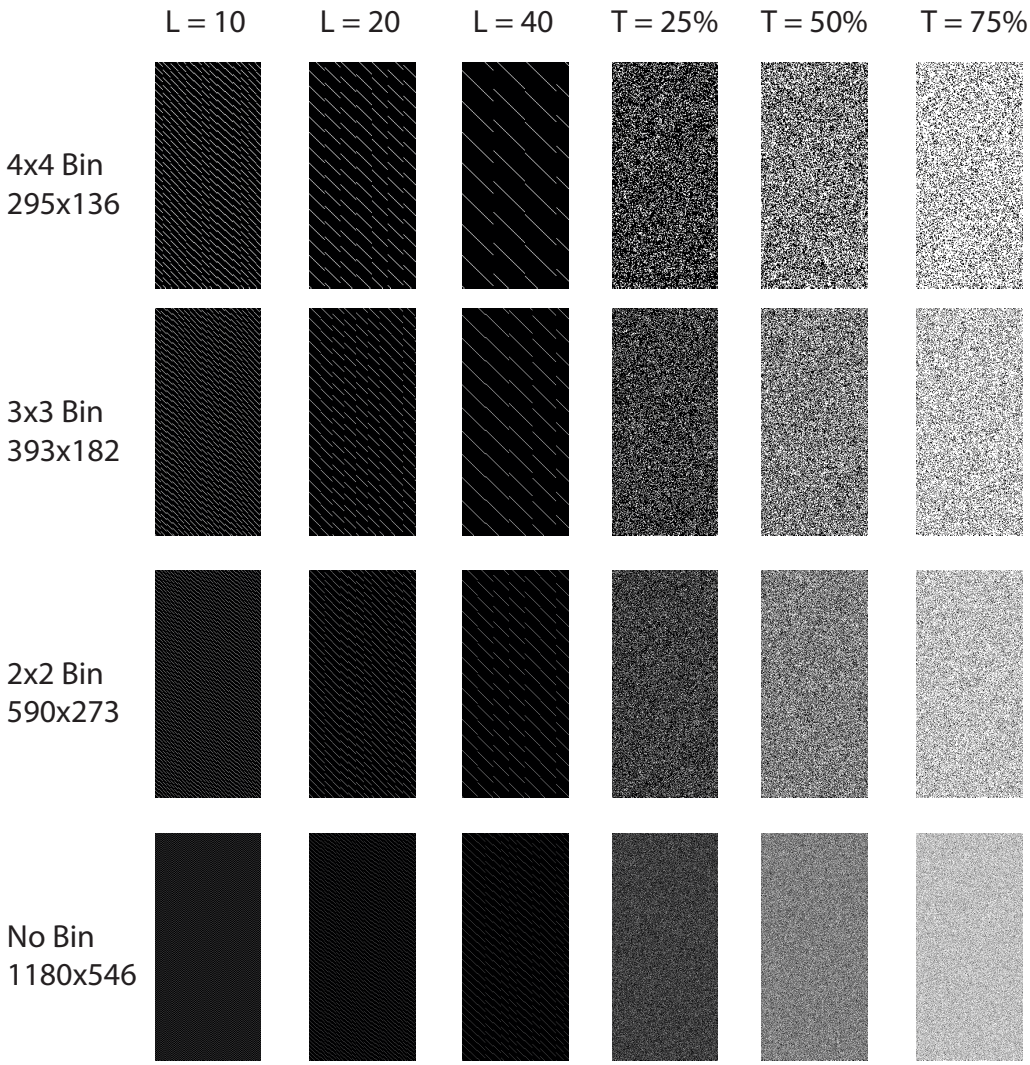

Figure 10: The schematic for the construction of the mask. The three leftmost columns are pseudo-random masks and the rightmost three are uniformly random. Each row signifies the pixel size of the masks of that row. The difference between the three columns of pseudo-random masks is the horizontal repetition rate of the masks, displayed atop each column. The corresponding number for the universally random masks is how much of the masks are transparent.

that spectral dispersion occurs solely along one axis of the detector array. Consequently, proper alignment of these three components was imperative to achieve good quality of reconstruction. This insight led to the creation of a methodical alignment process aiming to prevent mis-alignment. Since the spectrometer was a manufactured high-end spectrometer the grating could not be adjusted freely and thus had to be the basis of the

alignment process.

The first step, however, was to find the optical axis through the spectrometer. This was achieved by covering the hole in the spectrometer meant to hold the detector with a thin sheet of paper. This sheet was then illuminated with a white light source and the grating was turned to reflect the 0:th order diffraction component backwards through the spectrometer, see Fig 11. The direction of the optical axis was found (red crosses) and the sliding rail was placed along this axis.

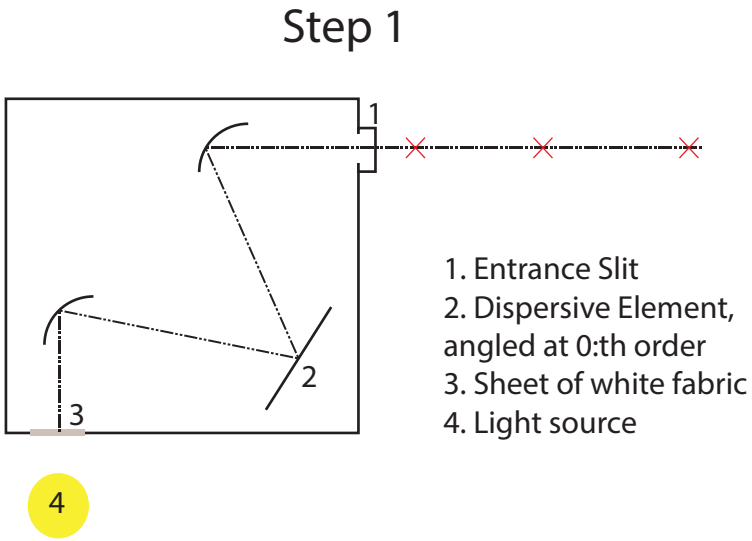

Figure 11: A schematic showing the set-up in the first step of the alignment process. A diffuse light source shines light backwards through the spectrometer, allowing the optical axis to be found.

In the second step the detector was mounted to the spectrometer and the light source was moved in front of the spectrometer, as displayed in Fig 12. The goal of this step was to align the detector array to the dispersion direction of the grating. To achieve this, the grating was turned to project the 1st order of dispersion onto the detector, resulting in the detection of a rainbow-pattern. The camera was adjusted so that the rainbow pattern was in focus and the direction of dispersion was parallel to one of the side of the detector screen, as discussed previously. From this point on no further adjustments was made to the camera.

The fact that the rainbow pattern was in focus assured that the first image plane was located at the entrance slit. In step 3 a lens was placed at a distance of $2f$ from the

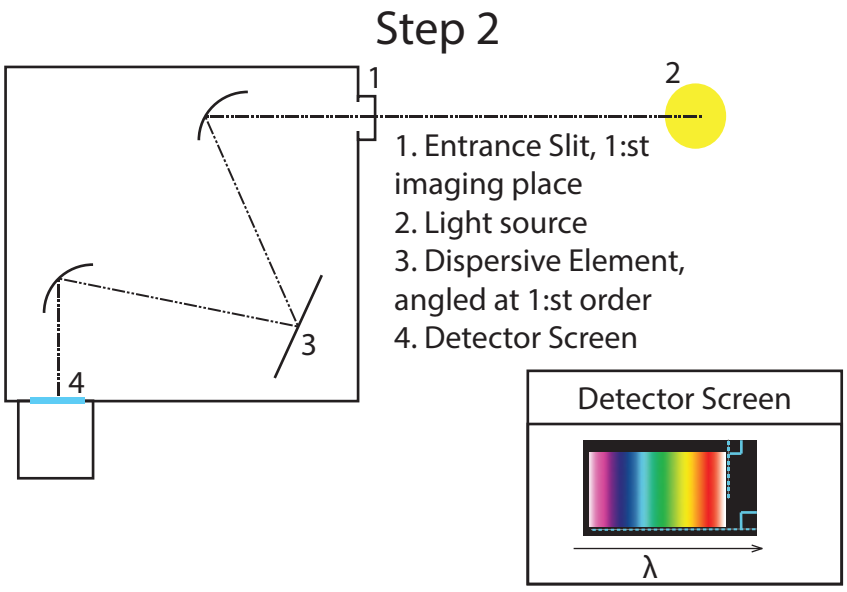

Figure 12: A schematic of the set-up during the second step of the alignment process, as well as a depiction of the detector screen. The detector should be adjusted so that the rainbow pattern is in focus and the direction of dispersion is parallel to one side of the detector screen.

entrance slit to create a second image plane where the studied object was to be placed. The grating was rotated back to 0:th order and a test object was placed at the second image plane. The entrance slit was removed and the location of the lens and the object was fine-tuned until the projection of the object onto the detector screen was in focus. A schematic is displayed in Fig 13.

The final step included placing the translation stage with the mask at the first image plane. The placement of the mask was fine-tuned so that the it was in focus on the detector screen and the 2D pattern of the mask aligned with that of the detector screen, as seen in Fig 14. Finally, the translation increments of the linear translator were fine-tuned so that multi-shot measurements would produce a full image.

This procedure guaranteed that the problems experienced in the first set-up were removed. By starting with the direction of dispersion of the grating and aligning the system step by step based on that the problems with misalignment caused by assumptions in the TwIST algorithm were removed.

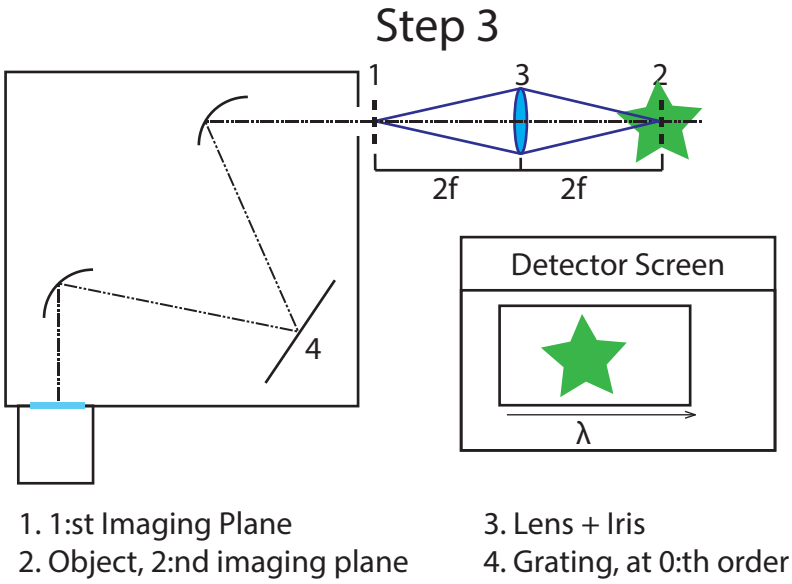

Figure 13: A schematic of the set-up during the third step of the alignment-process. The entrance slit has been removed and a test object is projected onto the detector screen.

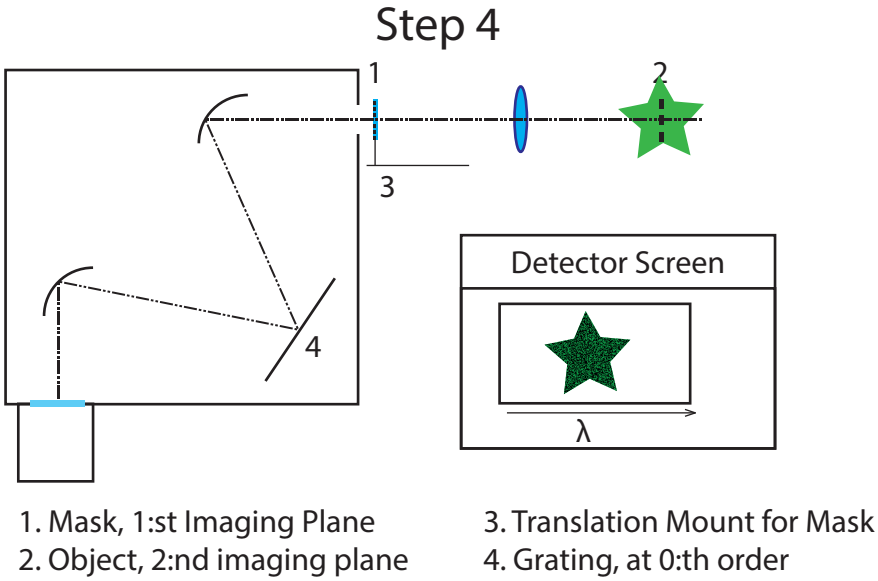

Figure 14: A schematic of the set-up during the fourth step of the alignment process. The mask has been introduced and adjusted so the alignment process is finished.

4 Method

In order to record a full spectrum, the following procedure was developed: A mask was chosen to be used in a measurement. The step-size of the vertical translator holding the mask was determined by the vertical spacing of the repeated pattern of the mask in the

case of a measurement using a pseudo-random mask. In the case of a measurement using a uniformly random mask the step-size was chosen arbitrarily but always in increments of the pixel size of the mask pattern. Between each recorded spectrum the mask was moved one increment until a full image had been covered.

This set of spectra was then loaded into the Matlab framework, together with an image of the mask depicted through the spectrometer at the 0:th order. A library of masks was created from the mask image, one at each increment of the spectral sub-band and each thus representing one spectral position to be investigated. A reconstruction was then achieved as previously presented for each mask in the library, using the set of recorded spectra.

Consequently, the CS methodology attempts to reconstruct spectral components where there are none, leading to false reconstructions. These false reconstructions are non-zero reconstructions located spectrally where no true spectral component exists. Thus, a measure of reconstruction quality must be created.

First, an objective measure of reconstruction quality would allow for the separation of true and false reconstructions. Secondly, it can give information on the level of reconstruction for known spectral components and, thirdly, it could potentially allow for component identification and localization by investigating how the level of reconstruction changes over a span of wavelengths.

A common technique in signal processing is cross-correlation, or sliding dot product (denoted $\star$) [15]. By investigating the cross-correlation between the reconstructed images and the image to be recreated (henceforth referred to as target image), assuming this is known, a measure of quality is achieved by summing the highest level of reconstruction in each row in an image. Furthermore, since the spectral scene is known to be very simple in the spectral band of interest, very low correlation is expected where no spectral component is present in this case, as displayed in Fig 15.

Because of this, an inverse envelope (a smoothed lower envelope function) can be calculated and subsequently subtracted from the cross-correlation spectrum. This is done to remove the low level of "background" cross-correlation, as it holds no information for the measurement. It will also remove additive artifacts (artifacts that additively increase the value of the cross-correlation), if any are inherent in the process. To further fool-proof

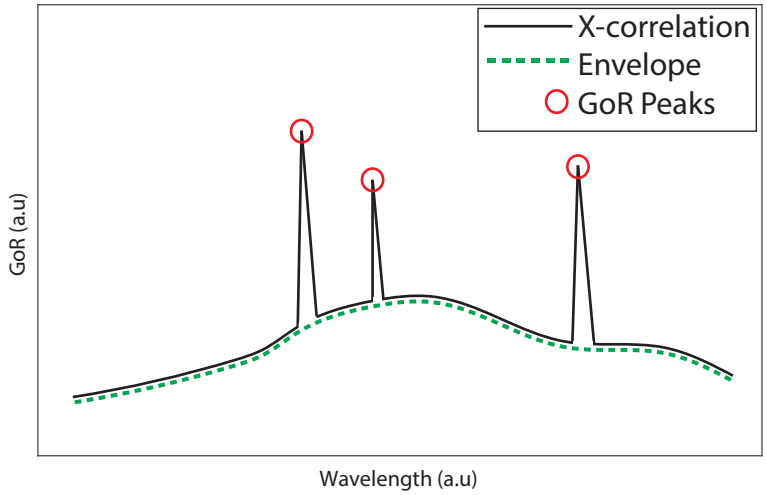

Figure 15: The expected shape of the cross-correlation curve between a reconstruction and the target image (black). Very low correlation is expected everywhere where no spectral component is present, thus an envelope (green) is subtracted from the curve to remove this background. By dividing by the envelope as well, low intensity GoR peaks hidden in the background can be found. This set of operations applied to the cross-correlation results in the Goodness of Reconstruction (GOR) curve of the reconstructions from a measurement.

the quality assessment and identify low-intensity peaks, the value is then divided by the envelope. Consequently, the Goodness of Reconstruction (GoR) curve can be assigned to a set of reconstructions from a measurement using the following formula:

$$GoR(\lambda) = \frac{(Reconstruction(\lambda) \star Original) - Envelope}{Envelope}$$

It is important to note that the amplitude of the GoR varies between measurements and between different masks. Thus, GoR is a relative value within each measurement and cannot be used to compare different measurements to each other.

5 Results and Discussion

5.1 Software Assessment

As a test of viability, data from two simultaneous measurements recorded, using Planar Laser-Induced Fluorescence (PLIF), was used to test the framework. The data consisted of the recorded 2D fluorescence signal from OH and CH_2O in a flame. These two images were used to create two artificial CS spectra, i.e. two spectra where each image is multiplied with a pseudo-random and a uniformly random mask respectively, and then placed at a location where the corresponding element has a spectral component. The spectra contained five components from each species and each component was adjusted to match the relative intensity of the transition. The two original images are displayed in Fig 16. One of the full spectra is displayed in Fig 17. Ideally the script would be able to divide each artificial spectrum in Fig 17 into the five sets of two images that make up the spectrum, identical to the set seen in Fig 16.

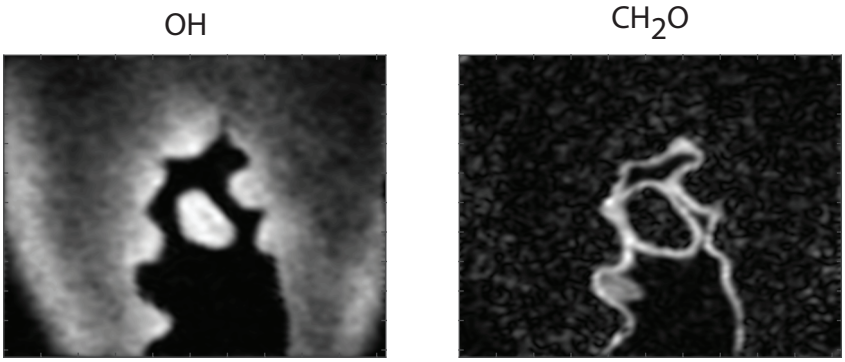

Figure 16: Two images from a PLIF measurement of a flame. Left: The OH fluorescence signal. Right: The CH_2O fluorescence signal. These two images were used to create the artificial spectrum displayed in Fig 17

Some of the resulting reconstructions for the CH_2O and OH signals present in the spectra are displayed in Fig 18 and 19, respectively, marked with what mask was used for the reconstruction. It is concluded that the procedure works, albeit not perfectly.

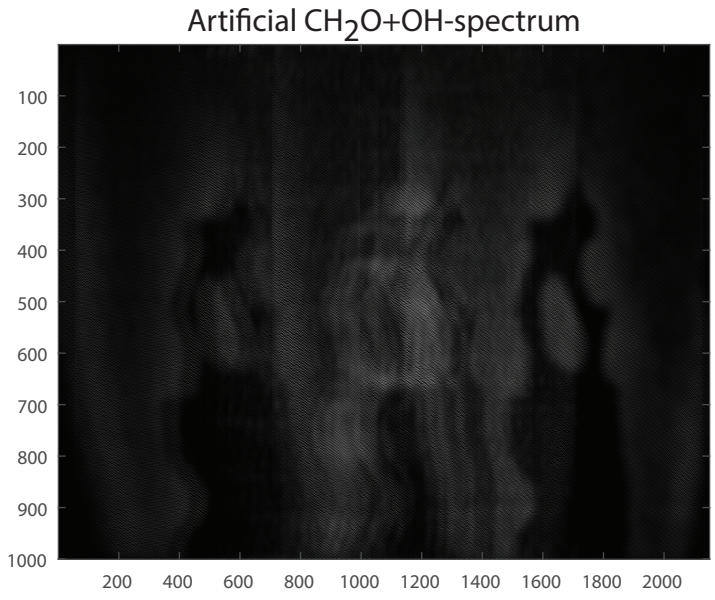

Figure 17: The artificially created CH_2O+OH spectrum, using a pseudo-random mask, to be used as a test-spectrum for the framework and for the TwIST algorithm. Ideally, the script would be able to separate this image into five copies each of the two images displayed in Fig 16

Use of neither mask produced perfect reconstructions, although a pseudo-random mask produced notably better reconstructions. Using the pseudo-random mask, reconstruction for both species show easily recognizable fluorescence signals. Imperfections can be seen in the reconstructions of the CH_2O signal however, as all reconstructions contain traces of several spectral components, albeit weak in most cases.

The reconstructions using a uniformly random mask are of poorer quality, both in terms of intensity of the reconstructed components and because almost all reconstructions show traces of several spectral components, not only in CH_2O but also in OH.

Since this was an early trial, the results were only evaluated briefly, the main goal was to see whether the CS methodology was at all possible. The main conclusions drawn from this early test were that the procedure can produce reconstructions to a fair degree of accuracy and that the pseudo-random mask performed better than the uniformly random mask. It was therefore of interest to continue using a designed mask, together with a uniformly random one, as the procedure was applied practically. It was also concluded that as long as the type of spectrum displayed in Fig 17 is achieved in a laboratory set-up, the Matlab framework could be used to achieve at least acceptable recovery of individual

spectral components. This was better than expected, as the data used for the simulated spectra was not perfect but from a measurement not means for CS reconstruction. It is possible that reconstruction quality can be much better in a measurement optimized for CS-construction.

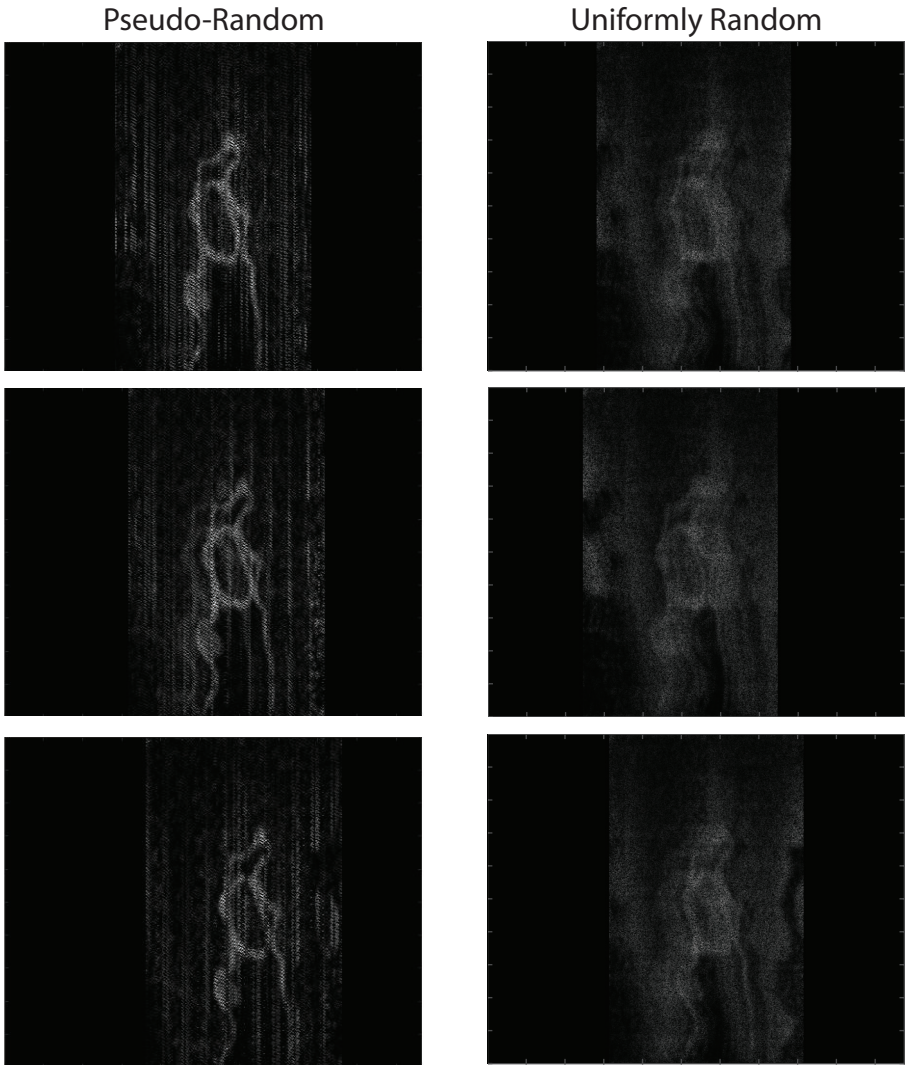

Figure 18: Reconstructions of the CH_2O fluorescence signal using a pseudo-random mask (left) and a uniformly random mask (right).

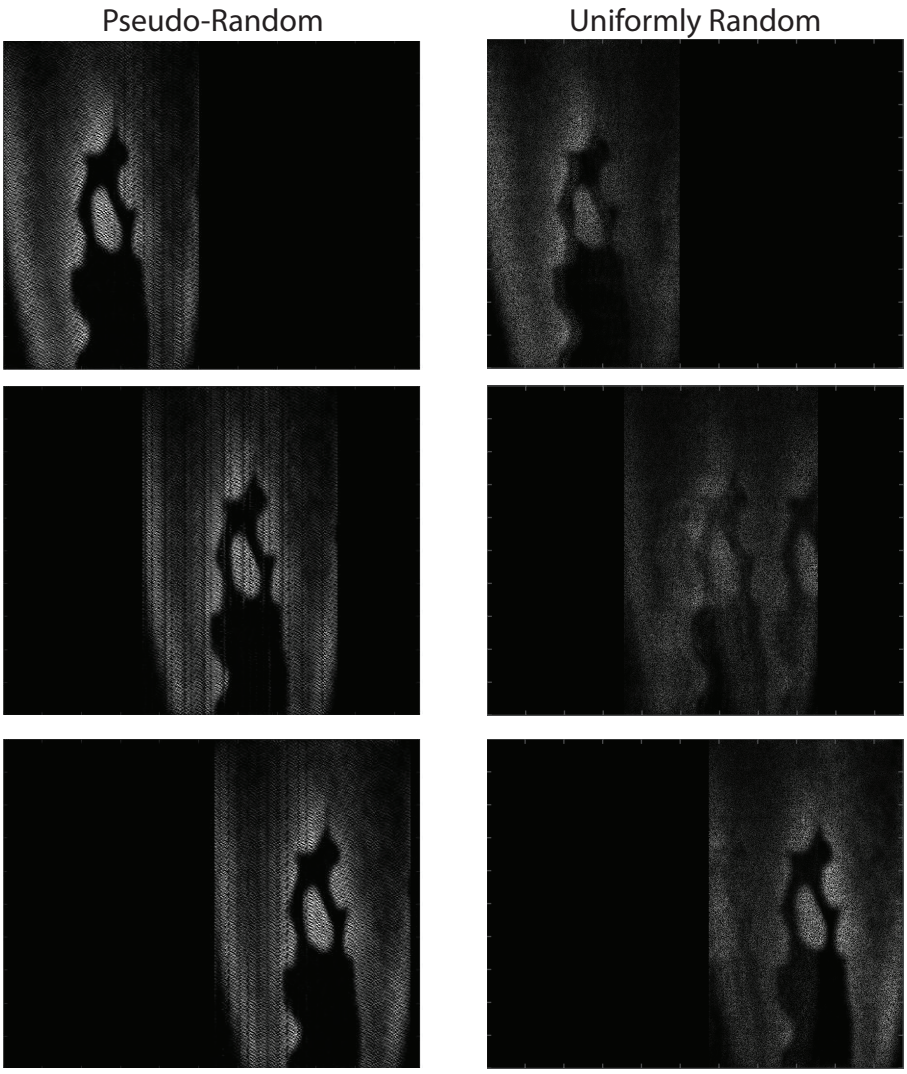

Figure 19: Reconstructions of the OH fluorescence signal using a pseudo-random mask (left) and a uniformly random mask (right).

5.2 First Setup

While the data from the first set-up proved to be of too poor quality to give sufficient information as to whether the method is practically doable, a number of conclusions regarding the set-up and the method as a whole was drawn.

By looking at the images displayed in Fig 20 to 22 it can be seen that the reconstructions in the right parts of the image are superior to those in the middle, which in turn are superior to those on the left side. Since the spectral resolving occurs from right to left in the images, it was concluded that this was most likely caused by an angular miss-alignment between the spectrometer and the detector. The post-processing assumes

that the image is spectrally shifted strictly along the x-axis, while this mismatch causes the shift to occur at a small angle to the x-axis. This resulted in a decrease in correlation between mask and image as shift distance increased. The main reason for this conclusion was the distribution of the light. In a best-case scenario an image consisting of the object at one spectral component would be visible with every other part of the image at 0 intensity. It is therefore not unreasonable to determine GoR in this case as a measure of the area at which the intensity is larger than 0 in the separated image, as the poor quality of the reconstructions lead to nonsense values of the previously presented GoR measure.

As seen in Fig 20 the algorithm cannot even distinguish between far separated spectral components and more than half of the image has an above-zero intensity. Observing Fig 21, the intensity distribution has shrunk into one continuous area but still mixing between the closer spectral components. In Fig 22 only the closest spectral components are mixed, the area with above-zero intensity is much smaller than previously and the image is sharper.

Based on this, it was concluded that a better set-up, with more precise alignment, had to be created. For future reference, the results presented here will henceforth be referred to as the "first results".

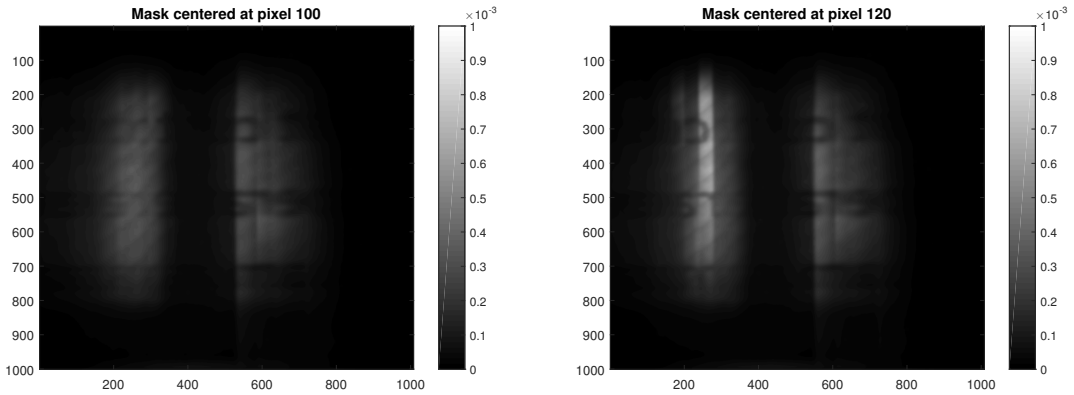

Figure 20: Reconstructions of spectral components between pixel 0 and 200, showing very poor reconstruction quality. The intensity is non-zero everywhere except for a part in the middle of the images.

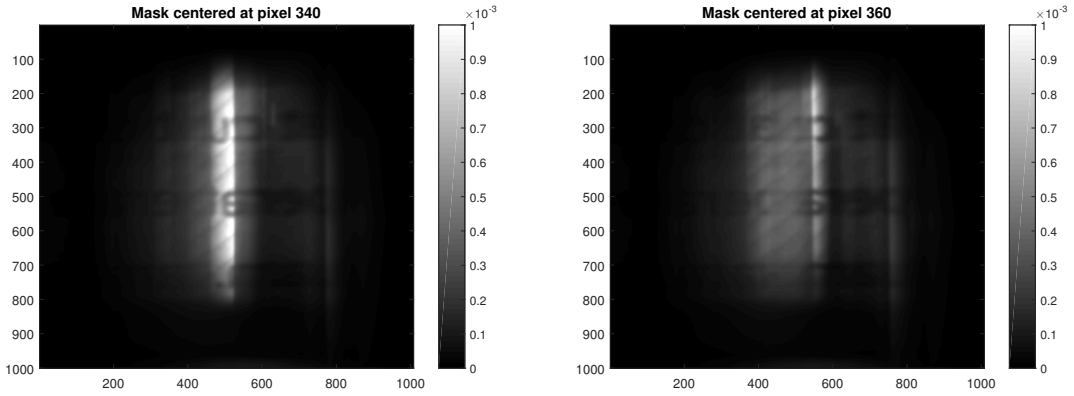

Figure 21: Reconstructions of spectral components between pixel 300 and 400, showing slight improvement in reconstruction quality, compared to Fig 20. While no coherent image structure is observed, the reconstructions are now continuous segments.

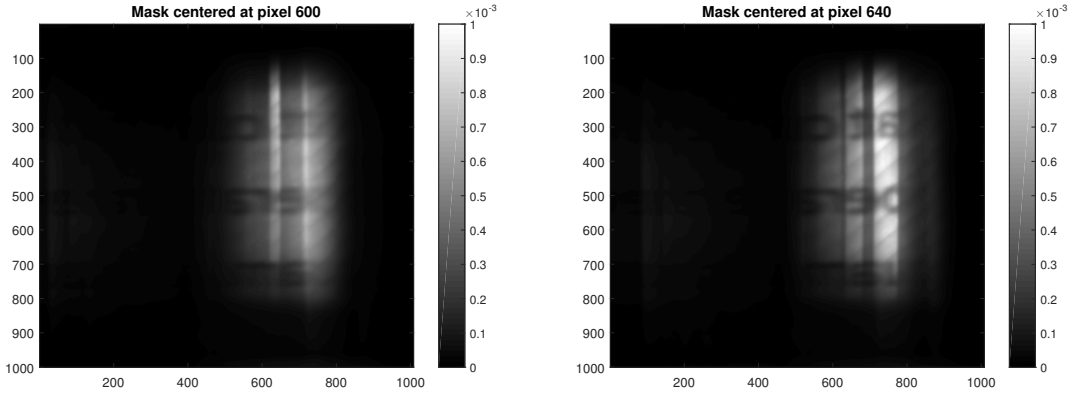

Figure 22: Reconstructions of spectral components between pixel 600 and 700. the reconstruction quality is still poor but the area with non-zero intensity is the right size and letters can be hinted at.

5.3 Second Set-up

5.3.1 Reconstruction using Pseudo-Random Masks

The reconstructions achieved using the second set-up proved to be of much higher quality, highlighting the importance of properly aligning the setup as the post-processing makes certain assumptions about the setup that needs to be fulfilled. The second set-up showed that high quality reconstruction can be achieved with the pseudo-random mask type, as presented in Fig 23. The images show the reconstruction of the spectral components in Zn close to 472 nm, 481 nm and 636 nm [16]. By visual inspection, it is concluded that the reconstructions are of good quality, as the text can be easily read, with all letters

clearly distinguishable.

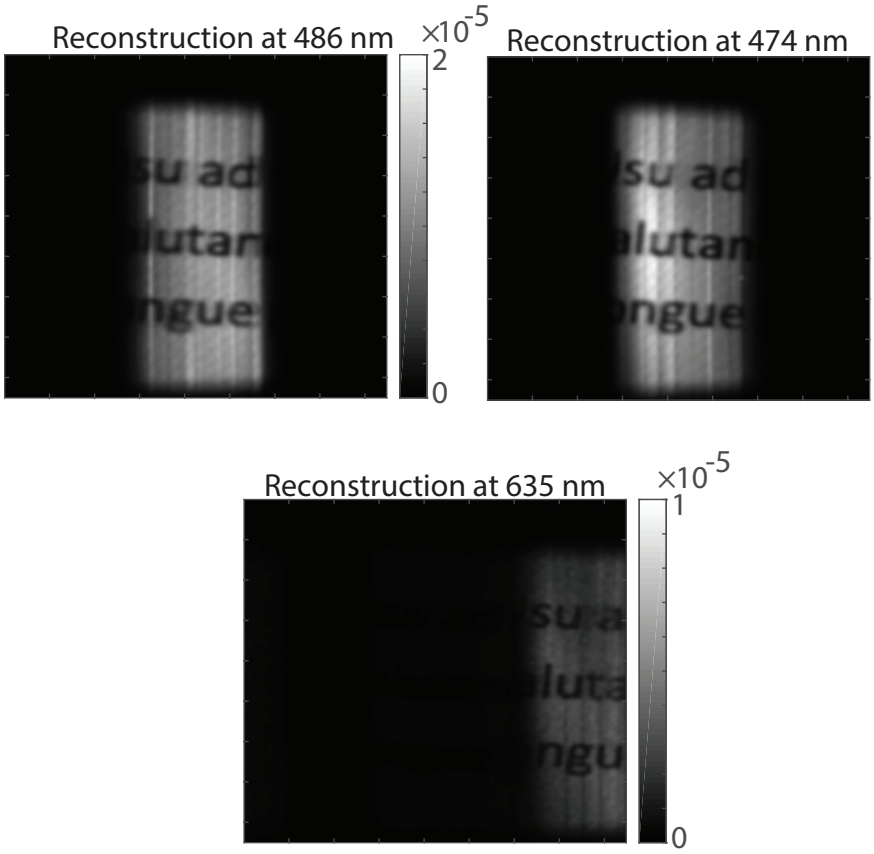

Figure 23: Reconstructions of the three spectral components at 472 nm, 481 nm and 636 nm in Zn, using a pseudo-random mask. The text is easily readable with each letter clearly distinguishable, proving the CS methodology shows promise for 3D spectroscopy.

Furthermore, the GoR-plot for the reconstructions is displayed in Fig 24. The graph plots GoR versus wavelength, i.e. what GoR was achieved at different wavelengths, assuming there is a spectral component present there. Expected is thus three peaks, corresponding to the three transitions in Zn present in the studied spectral band. The three components are marked in green in the figure and are clearly visible, however other peaks marking high GoR is present. The additional peaks mark false reconstructions and are discussed further below. The peak marked with in red is the false reconstruction shown in Fig 26, used to highlight the main differences between real and false reconstructions.

Although not as distinguishable from false reconstructions as hoped, apart from the $481nm$ transition which is also the strongest of the three transitions, all three transitions show relatively high GoR. The relative intensities of the reconstructed true spectral com-

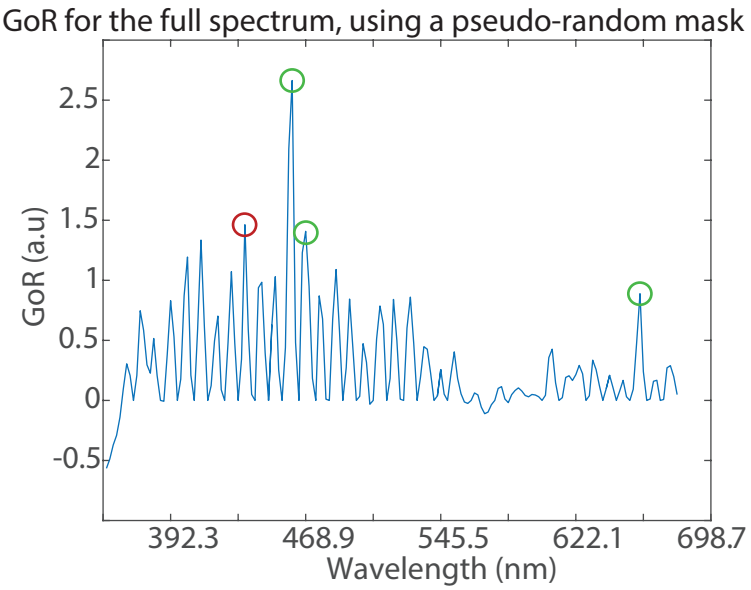

Figure 24: The GoR-spectrum for the reconstructions in Fig 23 and 25. The three true peaks are marked with a green circle (Fig 23) and one of the false reconstructions is marked with a red circle (Fig 25).

ponents are also on the same order of magnitudes as the intensities of the transitions given by [16]. This shows three things:

First, the CS methodology can be used for species identification as discussed earlier if a way to remove the false reconstructions can be achieved, be it through better designed masks or though software post-processing. The prevalence of false reconstructions seems to follow a somewhat periodic pattern so one option could be to apply a band-pass filter in the Fourier domain. This would, however, decrease the amplitude of the true reconstructions as well as introduce a risk for confirmation bias.

Second, weak transitions could be amplified by normalizing the reconstructed images before doing the GoR evaluation in order to remove the dependence on transition strength in the GoR. This was briefly attempted but proved to be detrimental to the results since the false reconstructions are normalized as well, effectively making all GoR peak amplitudes equally big. If the false reconstructions could be removed however, this could be investigated further.

Third, the GoR is decent at locating the true reconstructions. The reason for its sub-stellar performance is most likely the nature of the object being imaged. If it was white letters on a dark background instead of the reverse the overall amplitude of the

images would decrease and the evaluation method would thus be more sensitive to small differences in reconstruction. It is possible that this, if done right, could remove the false reconstruction, or at least reduce their amplitudes to a large extent.

5.3.2 False Reconstructions

A false reconstruction, such as that displayed in Fig 25 is due to the pseudo-random mask pattern, as this phenomenon only occurs with the pseudo-random masks. Despite each horizontal increment being spatially unique, as described earlier, they are still very much alike. This leads to false reconstructions since the algorithm assumes there to be spectral components at every location and thus creates the best reconstruction possible with the information it has.

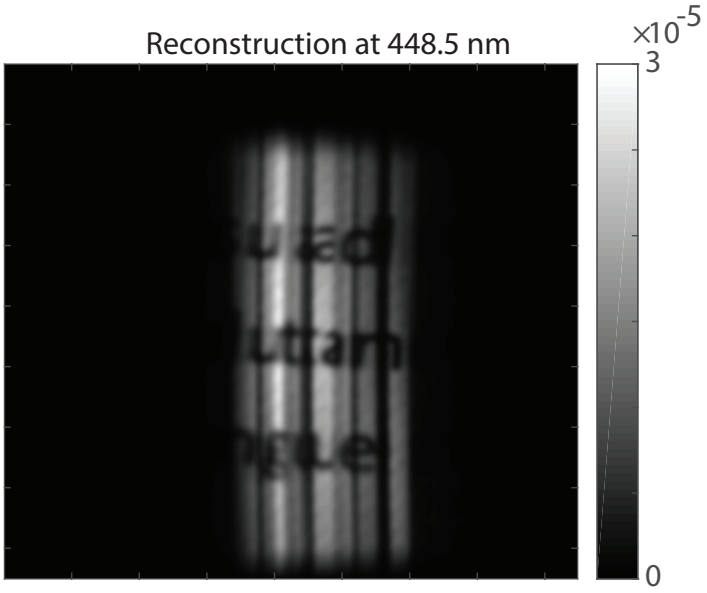

Figure 25: The false reconstruction highlighted in Fig 25. While achieving a high GoR there is no coherence in the reconstructed image.

Since the algorithm has complete information of the mask and its structure, it can interchange or shift horizontal segments of the mask that are similar if it is unable to create a better reconstruction at that location. This leads to false reconstructions with high GoR that essentially is a true reconstruction with the segments in the wrong order. An example of a false reconstruction with high GoR is displayed in Fig 26, together with

a real reconstruction for comparison. Three areas are highlighted with colored rectangles to show the differences. The segments marked with 1 and 2 are shifted, most easily seen by comparing the distances between them in the two reconstructions but also by noting their position in the reconstructions. The rectangle marked 3 highlights a segment present in the real reconstruction but not present in the false reconstruction.

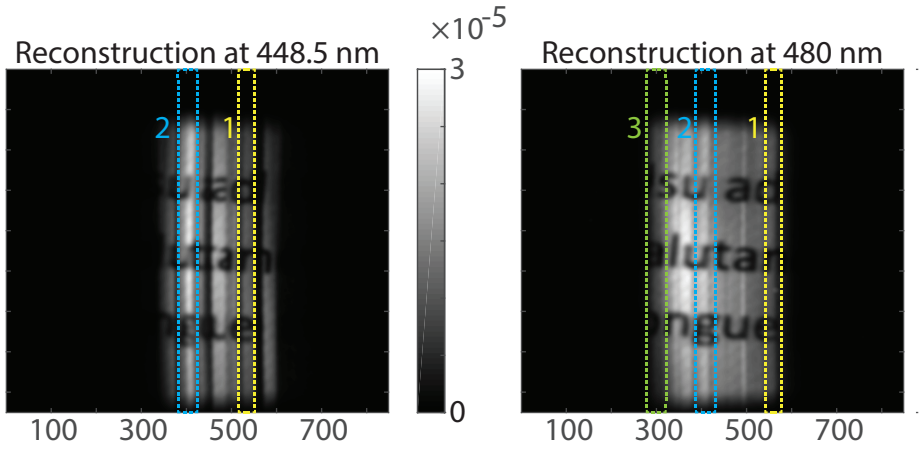

Figure 26: Comparison between a false and a real reconstruction. Three areas are highlighted with colored rectangles to show the differences. The segments marked with 1 and 2 are shifted, most easily seen by comparing the distances between them in the two reconstructions but also by noting their position in the reconstructions. The rectangle marked 3 highlights a segment present in the real reconstruction but not present in the false reconstruction.

It is thus concluded that false reconstructions cannot be identified solely using GoR in its current form, they must be identified through visual inspection of each individual reconstruction. If this is done, however, they are easily identified. All false reconstructions in this project exhibit the same characteristics: Very sharp horizontal steps in reconstruction intensity and no coherence (i.e. horizontal segments being misplaced, making false reconstructions very obvious if the general structure of the imaged object is known) in the image. Since the false reconstructions arise from the mask design, it is likely that they can be removed entirely by using a pseudo-random mask which has less likeness between segments.

The mask design chosen for the pseudo-random masks in this project is fairly simple and was chosen in order to limit the scope of the project. If masks were produced using one of the methods described in Chapter 3 in [17] this might not be a problem but no results are found concerning their effectiveness in this particular problem so that is by no means certain. Since, ideally, high GoR would only occur at wavelengths where there are actual spectral components, GoR plotted against wavelength should correlate well with the wavelength spectrum of the image. It is clear, however, that if 3D spectroscopy is ever going to be used as a species identification method the problem of false reconstructions has to be solved.

5.3.3 Comparison to the First Results

Compared to the first results, the second set-up achieved much better quality. The pseudo-random masks achieved reconstructions with high level of reconstruction, both in terms of GoR but also in terms of visual inspection. In the first results individual letters could be hinted but there was no coherence in the texts. Furthermore, the structure of the reconstructions suggests that the best reconstructions were false reconstructions, as the features identifying a false reconstruction are present in the first results as well.

5.3.4 Reconstruction using Uniformly Random Masks

The same reconstructions, achieved with a uniformly random mask, are displayed in Fig 27, with the GoR-plot in Fig 28. It is clear that a uniformly random mask is unable to achieve an acceptable level of reconstruction. This is motivated by the fact that no coherent text it visible. While the area of light in the reconstruction is accurate and the rows of text can be sensed but no individual letters are discernible, despite a very obviously non-constant horizontal intensity. This is unfortunate, as these reconstructions are made with ten measurements, which suggests that a snapshot method using CS and a uniformly random mask would be incapable of recreating anything but the crudest features in a 2D scene.

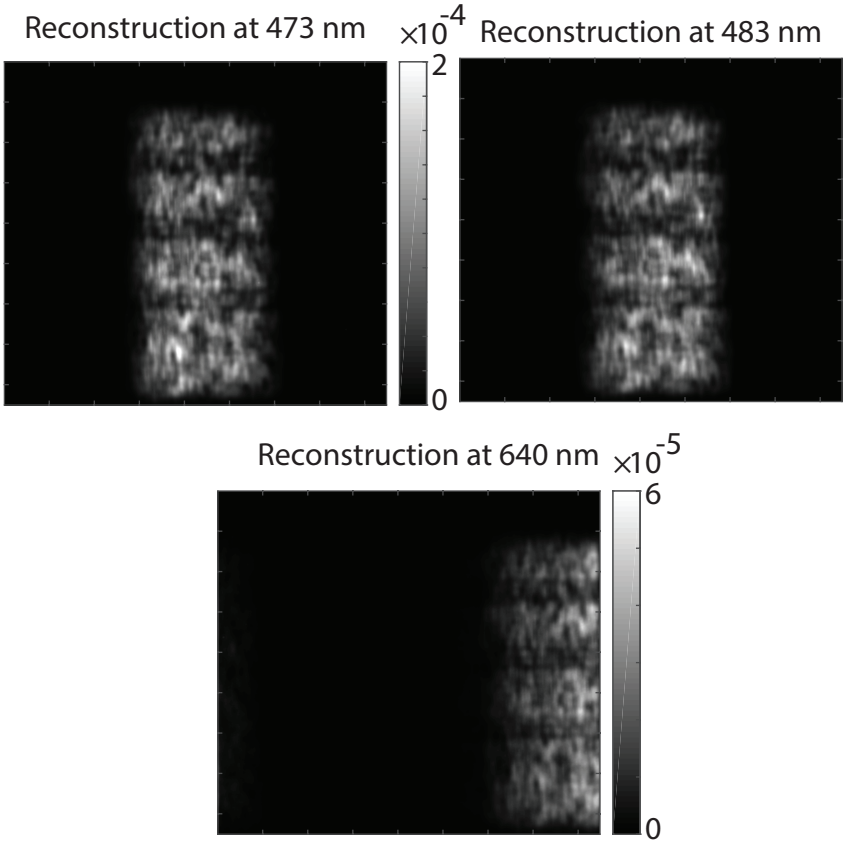

Figure 27: The reconstructions of the three spectral components in Zn, this time using a uniformly random mask. The quality of reconstruction is visibly lower, as no letters or words can be identified. Only darker rows where the text should be is present.

The GoR-plot mainly show that the concept of GoR is flawed in its current state. The reconstruction peaks identified when studying the GoR for pseudo-random masks were identified but only after intense scrutiny due to the very strange shape of the GoR curve. This was confirmed to be because the calculated envelope was very close to zero at two points, Consequently, it is concluded that the current model of GoR has flaws.

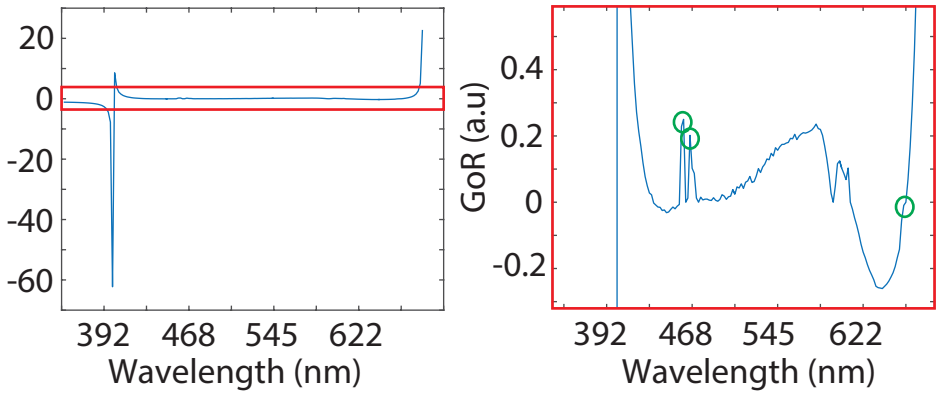

Figure 28: The GoR-curve for the reconstruction using uniformly random masks. The three expected GoR-peaks are located within the red box, expanded to the right, marked with green circles.

6 Conclusions

The following conclusions has been drawn from the project:

- The CS methodology performs to very well, both when software-based and in practice.
- Mask design plays a large role in reconstruction quality, with pseudo-random masks out-performing uniformly random masks consistently.
- Pseudo-random masks introduced a new problem: false reconstructions.
- False reconstructions cannot be identified using GoR in its current state. They are however easy to identify through optical inspection.
- The set-up of a 3D CS spectroscopic system requires high-precision alignment in order to yield good results.
- The CS methodology and the GoR-value can be used both for species identification and localization but it performs much better when used for localization.

- Snapshot imaging using the CS methodology is likely to yield satisfactory reconstructions for species localization assuming a loss of spatial detail is acceptable.
- The GoR needs re-evaluation. Very few conclusions could be drawn solely from the GoR, meaning that interpreting results still comes down to optical inspection, at least to an extent.
- Getting into the CS methodology is very difficult and very time-consuming. Due to the abstract nature of the mathematics behind CS it takes a lot of time to even get a cursory understanding of the method in use, even more so to fully understand the whole process. Despite this the method shows great promise and should be investigated further as a means of 3D spectroscopy, using this work as a basis.

7 Outlook

The next step in this development project would be to test the set-up as a spectral imaging detector in a laser-based experimental arrangement. Such an arrangement could be targeted for 2D Raman spectroscopy imaging or Planar laser-induced fluorescence spectroscopy. Ideal measurement objects would first be stable laminar flames that have well characterized species distributions.

Further development and optimization work should first aim to further investigate optimal mask designs since this is imperative in understanding the capabilities and limitations of the method. Being able to fully utilize the potential of a tailor-made mask could very much improve the quality of the reconstructions. This is in itself large enough for a full master's thesis; understanding the math behind the whole process (incoherence, sparsity, mask design, reconstruction etc.) and then creating a script that generates optimal masks for future use.

As stated, the CS methodology is highly time-consuming and very difficult. There is thus an incentive to learn more about the whole concept so that it can be presented in a less abstract way, so that the method, as well as research about it, is more easily accessible.

One option for the development snap-shot 3D spectroscopy, apart from mask optimization, is a pseudo-solution. Instead of a true snap-shot set-up, a set-up that very quickly records an image, moves the mask one increment and then records another image and so on could be created, possibly even a set-up that takes images as the mask is moves continuously. While not truly snap-shot, this could allow 3D spectroscopy to reach at least partway into the snap-shot domain of multi-spectral imaging.

Finally, one possible subject to study further is the incoherence of bases and in particular the incoherence between Fourier space and real space. It is possible that high quality reconstruction could be achieved based on a mask in the Fourier domain instead, or any other well-known basis for that matter.

8 Acknowledgement

I would like to thank my supervisors, Elias Kristensson and Andreas Ehn, for all the help, feedback and idea-bouncing they have supplied over the years (not limited to this project). Furthermore, their support and patience during hard times has been a huge relief and a helping hand in getting back in business.

Furthermore, I would like to thank my family for their patience with my need to tell them about my work and their curious questions despite the very abstract nature of the work.

I would also like to thank the student union at the Faculty of Science as well as the Physics and Lasershow at the Department of Physics. Without the friends I made and the support and joy I got from my involvement in both of those I would not be here today.

Finally, I would like to thanks everyone, both inside the Department of Physics at Lund University and outside, who have helped shape me to become the physicist this crowning achievement of my studies makes me.

9 Appendix

9.1 Matlab Code

9.2 RUNME.mat

```
%%%%%%%%%% Loading Data %%%%%%%%%%
Dir = 'LabbData\03-09\';

start = tic;

set(0,'DefaultFigureColormap',feval('gray'))

Dir_Mask = strcat(Dir,'R11\Mask\');
Dir_Image = strcat(Dir,'R11\Image\');
srcFiles_Mask = dir(strcat(Dir_Mask,'*.tif'));
srcFiles_Image = dir(strcat(Dir_Image,'*.tif'));
srcFiles_Bg = dir(strcat(Dir,'Background.tif'));
srcFiles_Or = dir(strcat(Dir,'Original.tif'));


 filename = strcat(Dir,srcFiles_Bg.name);
 Background = imread(filename);
 filename = strcat(Dir,srcFiles_Or.name);
 Original = flip(imread(filename),2);


for i = 1:1
    filename = strcat(Dir_Mask,srcFiles_Mask(i).name);
    Mask(:,:,i) = flip(imread(filename)-Background,2);

end
for i = 1:1
```

```
    filename = strcat(Dir_Image,srcFiles_Image(i).name);
    Image(:,:,i) = flip(imread(filename)-Background,2);

end

Mask = im2double(Mask);
Image = im2double(Image);

%%%%%%%%%%% Calculating Wavelength Range %%%%%%%%%%

Center_freq = 500;
Grating_Used = 1;

calib_const = [500 1200 2400];
calib_center = [200 30 15];

lambda_zero = [457 451 449];
lambda_one = [875 949 948];
lambda_diff = lambda_one - lambda_zero;

px2lambda = calib_center./lambda_diff;

lambda_range = round([Center_freq - size(Image,1).*px2lambda(1)./2 Center_freq +...
... size(Image,1).*px2lambda(2)./2 Center_freq + size(Image,1).*px2lambda(3)./2]);

%%%%%%%%%% Input Parameters from the Measurement %%%%%%%%%%

Num_Incr = 170;
y_start = 850;
y_stop = 0;
```

```
Spect_Start = 354;
Spect_End = 737;
lambdarangewhole = linspace(Spect_Start,Spect_End,11);
lambdarange = linspace(Spect_Start+(y_stop.*0.45),Spect_End-...
...(size(Image,1)-y_start).*0.45,11);

y_start_adj = y_start - 570;
y_stop_adj = y_stop - 570;

%%%%%%%%%% Sending data into Twist_Umbrella %%%%%%%%%%

Reconstruction = zeros(size(Image,1),size(Image,2),Num_Incr,size(Image,3));
Recon_Img = zeros(size(Image,1),size(Image,2),Num_Incr);

for i = 1:size(Image,3)

    disp(i);
    Reconstruction(:,:,:,i) = TwIST_Umbrella(Image(:,:,i),Mask(:,:,i),...
     ...Num_Incr,y_start_adj,y_stop_adj);

end

%%%%%%%%%% Creating Reconstructions %%%%%%%%%%

Algorithm_Time = toc(start)


for i = 1:Num_Incr

   Recon_Img(:,:,i) = sum(Reconstruction(:,:,i,:),4);
end
```

```
%%%%%%%%%% Calculating GoR %%%%%%%%%%

close all
for i = 1:Num_Incr


    Recon_Shift = Recon_Img;%circshift(Recon_Img,-round(((y_stop-y_start)...
     ..../(Num_Incr)).*i),2);
    CorrX = zeros([1,size(Image,2)*2-1]);
    for j = 1:size(Recon_Img,1)
        CorrX = CorrX + (xcorr(Original(j,:),Recon_Shift(j,:,i)));

%%%%%%%%%% Displaying Reconstruction, GoR etc. %%%%%%%%%%

    end
    CorrX = CorrX./size(Recon_Shift,1);
    CorrX_Max(i) = max(CorrX);


        %Plotting Correlation
    figure;
    plot(CorrX)

        %Plotting Contours
    figure;
    imagesc(Recon_Img(:,:,i) - circshift(circshift(Recon_Img(:,:,i),1,2),1,1));
    title('Contours of reconstruction')
    colorbar; caxis([(-1e-6) (1e-6)]);
    set(gca,'XTickLabel',[lambdarange]);


        %Plotting Reconstruction
```

```
    ppos = (lambdarange(1)+0.45.*i.*((y_start-y_stop)./Num_Incr));
    figure;
    imagesc(Recon_Img(:,:,i))
    title(['Reconstruction of component at ' num2str(ppos) ' nm']);
    colorbar; caxis([0 1e-5]);

end
UpperEnvelope = 0;
UpperEnvelope = 1-envelope(1-CorrX_Max,1,'peak');
CorrX_Max_Adj = (CorrX_Max - UpperEnvelope)./UpperEnvelope;

set(gca,'XTickLabel',[lambdarange]);

figure;
plot(CorrX_Max_Adj)
set(gca,'XTickLabel',[lambdarange]);

figure;
plot(CorrX_Max)

figure;
plot(UpperEnvelope)

figure;
imagesc(Original)



disp('*** End of Program ***')
```

9.3 TwIST_Umbrella.mat

```
function[Reconstruction] = TwIST_Umbrella(Image,Mask,Num_Incr,y_start,y_stop)
```

```
    MaskWidth = size(Mask,2);
    ImageWidth = size(Image,2);

%%%%%%%%%% Creating Mask Library %%%%%%%%%%

    [m,n] = size(Image);
        Mask(m,n) = 0;
        Mask = circshift(Mask,ImageWidth-MaskWidth,2);



    MaskLib = zeros(size(Mask,1),size(Mask,2),round(size(Image,2)/m));

    Mask = circshift(Mask,y_start,2);

    for i = 1:round(Num_Incr)
        MaskLib(:,:,i) = circshift(Mask,round(((y_stop-y_start)./(Num_Incr)).*i),2);
    end

    %%%%%%%%%% Defining constants for TwIST %%%%%%%%%%

    t = tic;

    tau = 1e-3;                                 %Constant in TwIST
    maxiterations = 100;                        %Limit number of iterations in TwIST
    tolA = 1e-8;                                %Tolerance level of TwIST


    A = @(x) Rfuntwist(x,MaskLib);
    AT = @(x) RTfuntwist(x,MaskLib);
```

```
    piter = 4;

    Psi = @(x,th) mycalltoTVnew(x,th,piter);
    Phi = @(x) TVnormspectralimaging(x);

    tolA = 1e-8;

   %%%%%%%%%% Run TwIST %%%%%%%%%%

    [Reconstruction,dummy,obj_twist,...
        times_twist,dummy,mse_twist]= ...
        TwISTmod(Image,A,tau,...
        'AT', AT, ...
        'Psi', Psi, ...
        'Phi',Phi, ...
        'Initialization',2,...
        'Monotone',1,...
        'StopCriterion',0,...
        'MaxIterA',maxiterations,...
        'ToleranceA',tolA,...
        'Debias',0,...
        'Verbose', 1);

    Run_time = toc(t)

end
```

9.4 TwIST Algorithm

```
function [x,x_debias,objective,times,debias_start,mses,max_svd] = ...
         TwIST(y,A,tau,varargin)
%
```

```
% Usage:
% [x,x_debias,objective,times,debias_start,mses] = TwIST(y,A,tau,varargin)
%
% This function solves the regularization problem
%
%     arg min_x = 0.5*|| y - A x ||_2^2 + tau phi( x ),
%
% where A is a generic matrix and phi(.) is a regularizarion
% function  such that the solution of the denoising problem
%
%     Psi_tau(y) = arg min_x = 0.5*|| y - x ||_2^2 + tau \phi( x ),
%
% is known.
%
% For further details about the TwIST algorithm, see the paper:
%
% J. Bioucas-Dias and M. Figueiredo, "A New TwIST: Two-Step
% Iterative Shrinkage/Thresholding Algorithms for Image
% Restoration",  IEEE Transactions on Image processing, 2007.
%
% and
%
% J. Bioucas-Dias and M. Figueiredo, "A Monotonic Two-Step
% Algorithm for Compressive Sensing and Other Ill-Posed
% Inverse Problems", submitted, 2007.
%
% Authors: Jose Bioucas-Dias and Mario Figueiredo, October, 2007.
%
% Please check for the latest version of the code and papers at
% www.lx.it.pt/~bioucas/TwIST
%
% ------------------------------------------------------------------------
```

```
% Copyright (2007): Jose Bioucas-Dias and Mario Figueiredo
%
% TwIST is distributed under the terms of
% the GNU General Public License 2.0.
%
% Permission to use, copy, modify, and distribute this software for
% any purpose without fee is hereby granted, provided that this entire
% notice is included in all copies of any software which is or includes
% a copy or modification of this software and in all copies of the
% supporting documentation for such software.
% This software is being provided "as is", without any express or
% implied warranty.  In particular, the authors do not make any
% representation or warranty of any kind concerning the merchantability
% of this software or its fitness for any particular purpose."
% ----------------------------------------------------------------------
%
%  ===== Required inputs =============
%
%  y: 1D vector or 2D array (image) of observations
%
%  A: if y and x are both 1D vectors, A can be a
%     k*n (where k is the size of y and n the size of x)
%     matrix or a handle to a function that computes
%     products of the form A*v, for some vector v.
%     In any other case (if y and/or x are 2D arrays),
%     A has to be passed as a handle to a function which computes
%     products of the form A*x; another handle to a function
%     AT which computes products of the form A'*x is also required
%     in this case. The size of x is determined as the size
%     of the result of applying AT.
%
%  tau: regularization parameter, usually a non-negative real
```

```
%       parameter of the objective  function (see above).
%
%
%  ===== Optional inputs =============
%
%  'Psi' = denoising function handle; handle to denoising function
%          Default = soft threshold.
%
%  'Phi' = function handle to regularizer needed to compute the objective
%          function.
%          Default = ||x||_1
%
%  'lambda' = lam1 parameters of the  TwIST algorithm:
%             Optimal choice: lam1 = min eigenvalue of A'*A.
%             If min eigenvalue of A'*A == 0, or unknwon,
%             set lam1 to a value much smaller than 1.
%
%             Rule of Thumb:
%                 lam1=1e-4 for severyly ill-conditioned problems
%                 lam1=1e-2 for mildly  ill-conditioned problems
%                 lam1=1    for A unitary direct operators
%
%             Default: lam1 = 0.04.
%
%             Important Note: If (max eigenvalue of A'*A) > 1,
%             the algorithm may diverge. This is  be avoided
%             by taking one of the follwoing  measures:
%
%                1) Set 'Monontone' = 1 (default)
%
%                2) Solve the equivalenve minimization problem
%
```

```
%             min_x = 0.5*|| (y/c) - (A/c) x ||_2^2 + (tau/c^2) \phi( x ),
%
%             where c > 0 ensures that  max eigenvalue of (A'A/c^2) <= 1.
%
%   'alpha' = parameter alpha of TwIST (see ex. (22) of the paper)
%             Default alpha = alpha(lamN=1, lam1)
%
%   'beta'  =  parameter beta of twist (see ex. (23) of the paper)
%              Default beta = beta(lamN=1, lam1)
%
%  'AT'    = function handle for the function that implements
%            the multiplication by the conjugate of A, when A
%            is a function handle.
%            If A is an array, AT is ignored.
%
%  'StopCriterion' = type of stopping criterion to use
%                    0 = algorithm stops when the relative
%                        change in the number of non-zero
%                        components of the estimate falls
%                        below 'ToleranceA'
%                    1 = stop when the relative
%                        change in the objective function
%                        falls below 'ToleranceA'
%                    2 = stop when the relative norm of the difference between
%                        two consecutive estimates falls below toleranceA
%                    3 = stop when the objective function
%                        becomes equal or less than toleranceA.
%                    Default = 1.
%
%  'ToleranceA' = stopping threshold; Default = 0.01
%
%  'Debias'     = debiasing option: 1 = yes, 0 = no.
```

```
%                 Default = 0.
%
%                 Note: Debiasing is an operation aimed at the
%                 computing the solution of the LS problem
%
%                         arg min_x = 0.5*|| y - A' x' ||_2^2
%
%                 where A' is the  submatrix of A obatained by
%                 deleting the columns of A corresponding of components
%                 of x set to zero by the TwIST algorithm
%
%
%  'ToleranceD' = stopping threshold for the debiasing phase:
%                 Default = 0.0001.
%                 If no debiasing takes place, this parameter,
%                 if present, is ignored.
%
%  'MaxiterA' = maximum number of iterations allowed in the
%               main phase of the algorithm.
%               Default = 1000
%
%  'MiniterA' = minimum number of iterations performed in the
%               main phase of the algorithm.
%               Default = 5
%
%  'MaxiterD' = maximum number of iterations allowed in the
%               debising phase of the algorithm.
%               Default = 200
%
%  'MiniterD' = minimum number of iterations to perform in the
%               debiasing phase of the algorithm.
%               Default = 5
```

```
%
%  'Initialization' must be one of {0,1,2,array}
%               0 -> Initialization at zero.
%               1 -> Random initialization.
%               2 -> initialization with A'*y.
%               array -> initialization provided by the user.
%               Default = 0;
%
%  'Monotone' = enforce monotonic decrease in f.
%               any nonzero -> enforce monotonicity
%               0 -> don't enforce monotonicity.
%               Default = 1;
%
%  'Sparse'   = {0,1} accelarates the convergence rate when the regularizer
%               Phi(x) is sparse inducing, such as ||x||_1.
%               Default = 1
%
%
%  'True_x' = if the true underlying x is passed in
%                this argument, MSE evolution is computed
%
%
%  'Verbose'  = work silently (0) or verbosely (1)
%
% ===================================================
% ============ Outputs ================================
%   x = solution of the main algorithm
%
%   x_debias = solution after the debiasing phase;
%                  if no debiasing phase took place, this
%                  variable is empty, x_debias = [].
%
```

```
%   objective = sequence of values of the objective function
%
%   times = CPU time after each iteration
%
%   debias_start = iteration number at which the debiasing
%                  phase started. If no debiasing took place,
%                  this variable is returned as zero.
%
%   mses = sequence of MSE values, with respect to True_x,
%          if it was given; if it was not given, mses is empty,
%          mses = [].
%
%   max_svd = inverse of the scaling factor, determined by TwIST,
%             applied to the direct operator (A/max_svd) such that
%             every IST step is increasing.
% ========================================================


%-----------------------------------------------------------------
% test for number of required parametres
%-----------------------------------------------------------------
if (nargin-length(varargin)) ~= 3
     error('Wrong number of required parameters');
end
%-----------------------------------------------------------------
% Set the defaults for the optional parameters
%-----------------------------------------------------------------
stopCriterion = 1;
tolA = 0.01;
debias = 0;
maxiter = 1000;
maxiter_debias = 200;
miniter = 5;
```

```
miniter_debias = 5;
init = 0;
enforceMonotone = 1;
compute_mse = 0;
plot_ISNR = 0;
AT = 0;
verbose = 1;
alpha = 0;
beta  = 0;
sparse = 1;
tolD = 0.001;
phi_l1 = 0;
psi_ok = 0;
% default eigenvalues
lam1=1e-4;   lamN=1;

% constants ans internal variables
for_ever = 1;
% maj_max_sv: majorizer for the maximum singular value of operator A
max_svd = 1;

% Set the defaults for outputs that may not be computed
debias_start = 0;
x_debias = [];
mses = [];

%------------------------------------------------------------
% Read the optional parameters
%------------------------------------------------------------
if (rem(length(varargin),2)==1)
  error('Optional parameters should always go by pairs');
else
```

```
 for i=1:2:(length(varargin)-1)
   switch upper(varargin{i})
    case 'LAMBDA'
      lam1 = varargin{i+1};
    case 'ALPHA'
      alpha = varargin{i+1};
    case 'BETA'
      beta = varargin{i+1};
    case 'PSI'
      psi_function = varargin{i+1};
    case 'PHI'
      phi_function = varargin{i+1};
    case 'STOPCRITERION'
      stopCriterion = varargin{i+1};
    case 'TOLERANCEA'
      tolA = varargin{i+1};
    case 'TOLERANCED'
      tolD = varargin{i+1};
    case 'DEBIAS'
      debias = varargin{i+1};
    case 'MAXITERA'
      maxiter = varargin{i+1};
    case 'MAXIRERD'
      maxiter_debias = varargin{i+1};
    case 'MINITERA'
      miniter = varargin{i+1};
    case 'MINITERD'
      miniter_debias = varargin{i+1};
    case 'INITIALIZATION'
      if prod(size(varargin{i+1})) > 1   % we have an initial x
init = 33333;    % some flag to be used below
x = varargin{i+1};
```

```
       else
 init = varargin{i+1};
       end
     case 'MONOTONE'
       enforceMonotone = varargin{i+1};
     case 'SPARSE'
       sparse = varargin{i+1};
     case 'TRUE_X'
       compute_mse = 1;
       true = varargin{i+1};
        size(true)
        size(y)
       if prod(double((size(true) == size(y))))
           plot_ISNR = 1;
       end
     case 'AT'
       AT = varargin{i+1};
     case 'VERBOSE'
       verbose = varargin{i+1};
     otherwise
      % Hmmm, something wrong with the parameter string
      error(['Unrecognized option: ''' varargin{i} ''''']);
    end;
  end;
end
%%%%%%%%%%%%

% twist parameters
rho0 = (1-lam1/lamN)/(1+lam1/lamN);
if alpha == 0
    alpha = 2/(1+sqrt(1-rho0^2));
```

```
end
if  beta == 0
    beta  = alpha*2/(lam1+lamN);
end


if (sum(stopCriterion == [0 1 2 3])==0)
   error(['Unknwon stopping criterion']);
end

% if A is a function handle, we have to check presence of AT,
if isa(A, 'function_handle') & ~isa(AT,'function_handle')
   error(['The function handle for transpose of A is missing']);
end


% if A is a matrix, we find out dimensions of y and x,
% and create function handles for multiplication by A and A',
% so that the code below doesn't have to distinguish between
% the handle/not-handle cases
if ~isa(A, 'function_handle')
   AT = @(x) reshape(AT*x(:),[size(y,1) size(y,2)]);
   A = @(x) reshape(A*x(:),[size(y,1) size(y,2)]);
end
% from this point down, A and AT are always function handles.

% Precompute A'*y since it'll be used a lot
Aty = AT(y);
% psi_function(Aty,tau)

% if phi was given, check to see if it is a handle and that it
% accepts two arguments
```

```
if exist('psi_function','var')
   if isa(psi_function,'function_handle')
      try  % check if phi can be used, using Aty, which we know has
           % same size as x
            dummy = psi_function(Aty,tau);
            psi_ok = 1;
      catch
         error(['Something is wrong with function handle for psi'])
      end
   else
      error(['Psi does not seem to be a valid function handle']);
   end
else %if nothing was given, use soft thresholding
   psi_function = @(x,tau) soft(x,tau);
end

% if psi exists, phi must also exist
if (psi_ok == 1)
   if exist('phi_function','var')
      if isa(phi_function,'function_handle')
         try  % check if phi can be used, using Aty, which we know has
              % same size as x
              dummy = phi_function(Aty);
         catch
           error(['Something is wrong with function handle for phi'])
         end
      else
        error(['Phi does not seem to be a valid function handle']);
      end
   else
      error(['If you give Psi you must also give Phi']);
   end
```

```
else  % if no psi and phi were given, simply use the l1 norm.
   phi_function = @(x) sum(abs(x(:)));
   phi_l1 = 1;
end


%---------------------------------------------------------------
% Initialization
%---------------------------------------------------------------
switch init
    case 0   % initialize at zero, using AT to find the size of x
       x = AT(zeros(size(y)));
    case 1   % initialize randomly, using AT to find the size of x
       x = randn(size(AT(zeros(size(y)))));
    case 2   % initialize x0 = A'*y
       x = Aty;
    case 33333
       % initial x was given as a function argument; just check size
       if size(A(x)) ~= size(y)
          error(['Size of initial x is not compatible with A']);
       end
    otherwise
       error(['Unknown ''Initialization'' option']);
end

% now check if tau is an array; if it is, it has to
% have the same size as x
if prod(size(tau)) > 1
   try,
      dummy = x.*tau;
   catch,
      error(['Parameter tau has wrong dimensions; it should be scalar or size(x)']),
```

```
   end
end

% if the true x was given, check its size
if compute_mse & (size(true) ~= size(x))
   error(['Initial x has incompatible size']);
end


% if tau is large enough, in the case of phi = l1, thus psi = soft,
% the optimal solution is the zero vector
if phi_l1
   max_tau = max(abs(Aty(:)));
   if (tau >= max_tau)&(psi_ok==0)
      x = zeros(size(Aty));
      objective(1) = 0.5*(y(:)'*y(:));
      times(1) = 0;
      if compute_mse
        mses(1) = sum(true(:).^2);
      end
      return
   end
end


% define the indicator vector or matrix of nonzeros in x
nz_x = (x ~= 0.0);
num_nz_x = sum(nz_x(:));

% Compute and store initial value of the objective function
resid =  y-A(x);
prev_f = 0.5*(resid(:)'*resid(:)) + tau*phi_function(x);
```

```
% start the clock
t0 = cputime;

times(1) = cputime - t0;
objective(1) = prev_f;

if compute_mse
   mses(1) = sum(sum((x-true).^2));
end

cont_outer = 1;
iter = 1;

if verbose
    fprintf(1,'\nInitial objective = %10.6e,  nonzeros=%7d\n',...
        prev_f,num_nz_x);
end

% variables controling first and second order iterations
IST_iters = 0;
TwIST_iters = 0;

% initialize
xm2=x;
xm1=x;

%-----------------------------------------------------------------
% TwIST iterations
%-----------------------------------------------------------------
```

```
while cont_outer
    % gradient
    grad = AT(resid);
    while for_ever
        % IST estimate
        x = psi_function(xm1 + grad/max_svd,tau/max_svd);
        if (IST_iters >= 2) | ( TwIST_iters ~= 0)
            % set to zero the past when the present is zero
            % suitable for sparse inducing priors
            if sparse
                mask = (x ~= 0);
                xm1 = xm1.* mask;
                xm2 = xm2.* mask;
            end
            % two-step iteration
            xm2 = (alpha-beta)*xm1 + (1-alpha)*xm2 + beta*x;
            % compute residual
            resid = y-A(xm2);
            f = 0.5*(resid(:)'*resid(:)) + tau*phi_function(xm2);
            if (f > prev_f) & (enforceMonotone)
                TwIST_iters = 0;  % do a IST iteration if monotonocity fails
            else
                TwIST_iters = TwIST_iters+1; % TwIST iterations
                IST_iters = 0;
                x = xm2;
                if mod(TwIST_iters,10000) == 0
                    max_svd = 0.9*max_svd;
                end
                break;  % break loop while
            end
        else
            resid = y-A(x);
```

```
        f = 0.5*(resid(:)'*resid(:)) + tau*phi_function(x);
        if f > prev_f
            % if monotonicity  fails here  is  because
            % max eig (A'A) > 1. Thus, we increase our guess
            % of max_svs
            max_svd = 2*max_svd;
            if verbose
                fprintf('Incrementing S=%2.2e\n',max_svd)
            end
            IST_iters = 0;
            TwIST_iters = 0;
        else
            TwIST_iters = TwIST_iters + 1;
            break;  % break loop while
        end
    end
end

xm2 = xm1;
xm1 = x;



%update the number of nonzero components and its variation
nz_x_prev = nz_x;
nz_x = (x~=0.0);
num_nz_x = sum(nz_x(:));
num_changes_active = (sum(nz_x(:)~=nz_x_prev(:)));

% take no less than miniter and no more than maxiter iterations
switch stopCriterion
    case 0,
```

```
        % compute the stopping criterion based on the change
        % of the number of non-zero components of the estimate
        criterion =  num_changes_active;
    case 1,
        % compute the stopping criterion based on the relative
        % variation of the objective function.
        criterion = abs(f-prev_f)/prev_f;
    case 2,
        % compute the stopping criterion based on the relative
        % variation of the estimate.
        criterion = (norm(x(:)-xm1(:))/norm(x(:)));
    case 3,
        % continue if not yet reached target value tolA
        criterion = f;
    otherwise,
        error(['Unknwon stopping criterion']);
end
cont_outer = ((iter <= maxiter) & (criterion > tolA));
if iter <= miniter
    cont_outer = 1;
end



iter = iter + 1;
prev_f = f;
objective(iter) = f;
times(iter) = cputime-t0;

if compute_mse
    err = true - x;
    mses(iter) = (err(:)'*err(:));
```

```
    end

    % print out the various stopping criteria
    if verbose
        if plot_ISNR
            fprintf(1,'Iteration=%4d, ISNR=%4.5e  objective=%9.5e, nz=%7d, criterion=%
                iter, 10*log10(sum((y(:)-true(:)).^2)/sum((x(:)-true(:)).^2) ), ...
                f, num_nz_x, criterion/tolA);
        else
            fprintf(1,'Iteration=%4d, objective=%9.5e, nz=%7d,  criterion=%7.3e\n',..
                iter, f, num_nz_x, criterion/tolA);
        end
    end

end
%-----------------------------------------------------------------
% end of the main loop
%-----------------------------------------------------------------

% Printout results
if verbose
    fprintf(1,'\nFinished the main algorithm!\nResults:\n')
    fprintf(1,'||A x - y ||_2 = %10.3e\n',resid(:)'*resid(:))
    fprintf(1,'||x||_1 = %10.3e\n',sum(abs(x(:))))
    fprintf(1,'Objective function = %10.3e\n',f);
    fprintf(1,'Number of non-zero components = %d\n',num_nz_x);
    fprintf(1,'CPU time so far = %10.3e\n', times(iter));
    fprintf(1,'\n');
end


%-----------------------------------------------------------------
```

```
% If the 'Debias' option is set to 1, we try to
% remove the bias from the l1 penalty, by applying CG to the
% least-squares problem obtained by omitting the l1 term
% and fixing the zero coefficients at zero.
%-----------------------------------------------------------------
if debias
    if verbose
        fprintf(1,'\n')
        fprintf(1,'Starting the debiasing phase...\n\n')
    end

    x_debias = x;
    zeroind = (x_debias~=0);
    cont_debias_cg = 1;
    debias_start = iter;

    % calculate initial residual
    resid = A(x_debias);
    resid = resid-y;
    resid_prev = eps*ones(size(resid));

    rvec = AT(resid);

    % mask out the zeros
    rvec = rvec .* zeroind;
    rTr_cg = rvec(:)'*rvec(:);

    % set convergence threshold for the residual || RW x_debias - y ||_2
    tol_debias = tolD * (rvec(:)'*rvec(:));

    % initialize pvec
    pvec = -rvec;
```

```
% main loop
while cont_debias_cg

    % calculate A*p = Wt * Rt * R * W * pvec
    RWpvec = A(pvec);
    Apvec = AT(RWpvec);

    % mask out the zero terms
    Apvec = Apvec .* zeroind;

    % calculate alpha for CG
    alpha_cg = rTr_cg / (pvec(:)'* Apvec(:));

    % take the step
    x_debias = x_debias + alpha_cg * pvec;
    resid = resid + alpha_cg * RWpvec;
    rvec  = rvec  + alpha_cg * Apvec;

    rTr_cg_plus = rvec(:)'*rvec(:);
    beta_cg = rTr_cg_plus / rTr_cg;
    pvec = -rvec + beta_cg * pvec;

    rTr_cg = rTr_cg_plus;

    iter = iter+1;

    objective(iter) = 0.5*(resid(:)'*resid(:)) + ...
        tau*phi_function(x_debias);
    times(iter) = cputime - t0;

    if compute_mse
```

```
            err = true - x_debias;
            mses(iter) = (err(:)'*err(:));
        end

        % in the debiasing CG phase, always use convergence criterion
        % based on the residual (this is standard for CG)
        if verbose
            fprintf(1,' Iter = %5d, debias resid = %13.8e, convergence = %8.3e\n', ..
                iter, resid(:)'*resid(:), rTr_cg / tol_debias);
        end
        cont_debias_cg = ...
            (iter-debias_start <= miniter_debias )| ...
            ((rTr_cg > tol_debias) & ...
            (iter-debias_start <= maxiter_debias));

    end
    if verbose
        fprintf(1,'\nFinished the debiasing phase!\nResults:\n')
        fprintf(1,'||A x - y ||_2 = %10.3e\n',resid(:)'*resid(:))
        fprintf(1,'||x||_1 = %10.3e\n',sum(abs(x(:))))
        fprintf(1,'Objective function = %10.3e\n',f);
        nz = (x_debias~=0.0);
        fprintf(1,'Number of non-zero components = %d\n',sum(nz(:)));
        fprintf(1,'CPU time so far = %10.3e\n', times(iter));
        fprintf(1,'\n');
    end
end

if compute_mse
    mses = mses/length(true(:));
end
```

```
%-------------------------------------------------------------
% soft for both real and  complex numbers
%-------------------------------------------------------------
function y = soft(x,T)
%y = sign(x).*max(abs(x)-tau,0);
y = max(abs(x) - T, 0);
y = y./(y+T) .* x;
```

References

[1] Hearnshaw, J.B. *The analysis of starlight* Cambridge: Cambridge University Press. ISBN 0-521-39916-5.

[2] A. Hosseinnia, E. Nordström, H. Fatehi, J. Bood, P. Bengtsson*Rotational CARS thermometry and concentration measurements in ethane-nitrogen mixtures using Fourier analysis*

[3] S. Svanberg, *Atomic and Molecular Spectroscopy* Fourth Edition, ISBN 3-540-20382-6.

[4] M. Davenport, M. Duarte, Y. Eldar and G. Kutyniok, *Compressed Sensing: Theory and Applications*(pp. 1-64). Cambridge: Cambridge University Press. doi:10.1017/CBO9780511794308.002

[5] E. Candès and M. Wakin, *An Introduction to Compressive Sampling* IEEE Signal Processing Magazine, vol. 25, no. 2, pp. 21-30, March 2008. doi: 10.1109/MSP.2007.914731

[6] J. Hou, L. P. Chau, N. Magnenat-Thalmann and Y. He, *Sparse Low-Rank Matrix Approximation for Data Compression* IEEE Transactions on Circuits and Systems for Video Technology, vol. 27, no. 5, pp. 1043-1054, May 2017. doi: 10.1109/TCSVT.2015.2513698

[7] E. Candès and J. Romberg, *Sparsity and incoherence in compressive sampling* Inverse Prob., vol. 23, no. 3, pp. 969-985, 2007

[8] P. Llull, X. Liao, X. Yuan, J. Yang, D. Kittle, L. Carin, G. Sapiro, and D. Brady, *Coded aperture compressive temporal imaging* Opt. Express 21, 10526-10545 (2013)

[9] P. Ye, H. Arguello, and G. Arce, *Spectral Aperture Code Design for Multi-Shot Compressive Spectral Imaging*, Biomedical Optics and 3-D Imaging, OSA Technical Digest (CD) (Optical Society of America, 2010), paper DWA6

[10] A. Rajwade, D. Kittle, T. Tsai, D. Brady and L. Carin, *Coded Hyperspectral Imaging and Blind Compressive Sensing* SIAM Journal on Imaging Sciences 2013 6:2, 782-812

[11] J. M. Bioucas-Dias and M. A. T. Figueiredo, *New TwIST: Two-Step Iterative Shrinkage/Thresholding Algorithms for Image Restoration* IEEE Transactions on Image Processing, vol. 16, no. 12, pp. 2992-3004, Dec. 2007. doi: 10.1109/TIP.2007.909319

[12] J. M. Bioucas-Dias and M. A. T. Figueiredo *TwIST - Two-step Iterative Shrinkage/Thresholding Algorithm for Linear Inverse Problems* Retrieved from http://www.lx.it.pt/ bioucas/TwIST/TwIST.htm 2018-03-18

[13] Specification for Philips Zn Spectral lamp *http://www.lamptech.co.uk/Spec%20Sheets/D%20SP%20* Retrieved 2018-04-27

[14] Lorem Ipsum text generator *http://generator.lorem-ipsum.info/* Retrieved 2018-05-02

[15] K. Kapinchev, A. Bradu, F. Barnes and A. Podoleanu, *GPU implementation of cross-correlation for image generation in real time* 2015 9th International Conference on Signal Processing and Communication Systems (ICSPCS), Cairns, QLD, 2015, pp. 1-6. doi: 10.1109/ICSPCS.2015.7391783

[16] Table of transitions in Zn *https://physics.nist.gov/PhysRefData/Handbook/Tables/zinctable2.htm* Retrieved 2018-04-23

[17] H. Arguello Fuentes *Coded Aperture Optimization in Compressive Spectral Imaging*, PhD, University of Delaware (2013)