ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

CVL Computer Vision Lab

Sparse coding of PSF kernels for fast non-gridded convolution

Semester Thesis

Prateek Purwar
Department of Electrical Engineering and Information Technology

Advisor: Dr. Gregory Paul
Supervisor: Prof. Dr. Orcun Goksel

Abstract

The generation of realistic synthetic microscopy images requires convolution with a point spread function (PSF). Recently, Samuylov et al. described a method to generate realistic microscopy images from objects modeled in object space. The computational bottleneck of this approach is the non-gridded convolution entailed by the description in object space. To keep computational efficiency they restricted the PSF to be gaussian, which allows good performance by using the Improved fast Gaussian transform (IFGT). The goal of this work is to overcome this restriction but to keep the computational efficiency offered by the IFGT. This requires finding a good representation of an arbitrary PSF using sparse combination of Gaussian kernels to achieve both accuracy and speed.

In this work we tackle this task by solving a Bayesian inverse problem aiming at approximating an observed PSF by a sparse mixture of gaussian atoms. This requires solving two intertwinned tasks: a model selection problem (How many? What type? Of gaussian atoms) and a fitting problem (position and intensity of the kernels). We formulate a series of non-convex optimization problems derived from the maximum a posterior problem (MAP). In order to solve this joint task, using robust and efficient algorithms, we reformulate the MAP in a parametric **dictionary** together with a sparsity heuristic based on 1-norm regularization. We derive our own alternating direction method of multiplier (ADMM) iterations and its implementation in Python. This allow exploring the optimal tradeoff between accuracy (in representing the experimental PSF) and computational efficiency (number of gaussian atoms).

We compare experimentally the potential improvement of mixture of gaussians PSF over a single gaussian PSF model. The single gaussian problem is solved using a black-box optimization heuristic, covariance matrix adaptation evolutionary strategy (*CMA-ES*) and is compared to our ADMM. We assess our approach on both synthetic and real PSF image data. For gaussian PSF we can achieve high precision, as expected. For realistic PSF models, such as Gibson-Lanni, we can still approximate the PSF with good accuracy in our **dictionary** .

Acknowledgements

I would like to acknowledge the support of my Semester Thesis advisor Dr. Gregory Paul, Post-Doc. at Computer Vision Laboratory at ETH. He consistently steered me in the right direction whenever he thought I needed it. I would sincerely thank Denis Samuylov, who helped with implementation and coding at each step of my thesis.

Contents

List of Figures

Chapter 1

Introduction

1.1 Focus of this Work

The goal of this project is to develop algorithm to estimate PSF using sparse convex combination of Gaussian kernels. The motivation behind this is to generate realistic synthetic microscopy images. Samuylov et al. [2] developed a general framework (the virtual source framework) for computing the convolution of arbitrary object models formulated in physical space with point spread function (PSF) to generate synthetic images. The objects are represented as conical combinations of computational point sources irregularly positioned in physical space and carrying an effective light intensity. The irregular positioning of point sources requires non-gridded convolution. Unfortunately, fast convolution algorithms (based on the fast Fourier transform, FFT) cannot be used in such a case because they exploit the fact that digital signals and filters are regularly sampled. Nonetheless, Samuylov et al. demonstrated that the convolution can be efficiently carried out when the PSF is Gaussian by using the improved fast Gaussian transform, IFGT [3].

The main scope of this project is to extend the applicability of the virtual microscope framework to experimentally-generated PSF kernels. The classical approach is to represent the PSF using single Gaussian. This approach do not work well for experimentally-generated PSF, as they are rarely close to a Gaussian PSF. Therefore, in this project, we propose to represent an arbitrary PSF by a convex combination of Gaussian kernels, and to carry out the convolution by many independent IFGTs, as described in Samuylov et al. In addition, we propose to learn an optimal sparse convex combination of Gaussian kernels in order to minimize the number of IFGTs. This helps in achieving the optimal trade-off between accuracy (in representing the experimental PSF) and efficiency (number of Gaussians and hence IFGTs).

1.1.1 Formulation of forward problem

The estimation of PSF and object parameters *i.e.* intensity and position of point sources is known as inverse problem. The first necessary modeling step to solve inverse problem is formulation of forward problem. The formulation to synthesize images using virtual point sources, located arbitrarily in 3-dimensional real world, is known as forward problem. Modeling the image formation process spans three spaces: the object space where the sample resides i.e. 3-dimensional real world, the pixel space where light is collected, and the image space where images data is represented. The image space consist of stack of 2-dimensional image planes.

The photon flux in space and time is obtained by convolution of point sources with PSF (Gaussian kernel in our case). The pixel intensity at center of each pixel is computed as accumulation of photons in the pixel area for give acquisition time *i.e.* integration over time and pixel area. Finally, effect of shot noise

is considered to generate real images. The noise in real images comes due to electron multiplier, spurious electron and quantum efficiency [4]. The flow form object space to image space is modeled as following:

1. Point source to photon flux
   The process of collecting photons can be evaluated using PSF, K. For a point source with constant intensity, ϕ, located at $\boldsymbol{x}_s \in \mathbb{R}^3$, the photon flux is convolution of source with PSF:

$$\Phi(\boldsymbol{x}, t) = \phi \left(K * \delta_{\boldsymbol{x}_s(t)} \right)(\boldsymbol{x}, t) \quad .$$

2. Sampling over a pixel area for certain time
   The expected number of photons in a particular area can be measured by discretely counting the number photons for a time period. This leads to integration over time, T and the region of pixel, $\mathcal{P}$.

$$\mu = \int_{\mathcal{P} \times [0,T]} \Phi(\mathrm{d}\boldsymbol{x} \times \mathrm{d}t) \quad .$$

   As source intensity and source location are considered fixed in time, the integral came be replaced by direct multiplication with time interval. The intensity at each pixel is accumulation of photon flux over pixel area. As the size of pixel is very small, we assume a constant photon flux in a pixel area and, this simplifies the integration to:

$$\mu(\boldsymbol{x}_q) = \phi \, A \, T \, K(\boldsymbol{x}_q - \boldsymbol{x}_s) \quad , \tag{1.1}$$

   where $\mu(\boldsymbol{x}_q)$ is expected number of photons in pixel q, $\boldsymbol{x}_q$ is center of pixel and A is area of the pixel.

3. Effect of shot noise
   The final intensity in each pixel is realized after addition of background intensity, ϕ_b, and effect of noise due to various factors. The effect of shot noise is modeled as choosing a random value from Poisson distribution with mean and variance equal to final expected intensity. The image generated for single point source and PSF represented by one Gaussian kernel is shown in figure 1.1.

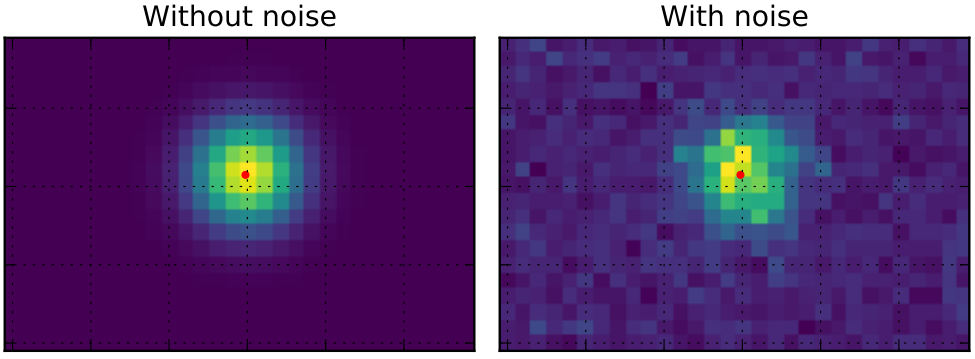


Figure 1.1: Image with no background noise and with SNR = 10

1.1.2 Use of IFGT for non-gridded convolution

The forward problem transforms to evaluating PSF function at each pixel center in object space. As, we are estimating PSF in terms of convex combination of Gaussian kernels, this requires computing Gaussian kernel at each pixel position and thus, is a non-gridded convolution. In gridded convolution, the convolution reduces to multiplication of arrays. This is not the case with non-gridded convolution which computes exponential of squared distance at each pixel center and thus, not efficient to calculate. We use *figtree* library to efficiently compute this non-gridded convolution. The *figtree* library computes the exponential effectively using Improved fast Gaussian transform method described in Yang et al. [3]. The library tends to compute radial basis function, given different centers and query positions. The computation of Gaussian kernel with given mean and covariance matrix is converted to variables required by *figtree* library. As, the expected image created by a point source can be computed as in equation 1.1 with the PSF as multivariate Gaussian kernel. The PSF, K, with mean, $\boldsymbol{m}$ and covariance matrix, $\boldsymbol{\Sigma}$ is equal to:

$$K(\boldsymbol{x} \,|\, \boldsymbol{m}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^3\, det(\boldsymbol{\Sigma})}} \exp\left(\frac{-(\boldsymbol{x}-\boldsymbol{m})^T\, \boldsymbol{\Sigma}^{-1}\, (\boldsymbol{x}-\boldsymbol{m})}{2}\right) \quad .$$

Replacing equation of f above in equation 1.1 gives us the expected image value at each pixel. With the following replacements,

$$\boldsymbol{m}' \,=\, \boldsymbol{x}_s + \boldsymbol{m} \quad \& \quad \phi' \,=\, \phi\, A\, T\ \frac{1}{\sqrt{(2\pi)^3\, det(\boldsymbol{\Sigma})}} \quad ,$$

the above formula can be rewritten as:

$$\mu(\boldsymbol{x}_q) = \phi' \exp\left(\frac{-\left(\boldsymbol{x}_q-\boldsymbol{m}'\right)^T\, \boldsymbol{\Sigma}^{-1}\, \left(\boldsymbol{x}_q-\boldsymbol{m}'\right)}{2}\right) \quad . \tag{1.2}$$

The *figtree* library provides functionality to compute output of a radial basis function at a query point, $\boldsymbol{y}_q$, from center, $\boldsymbol{p}$, as given below:

$$R(\boldsymbol{y}_q) = \, w \, \exp\left(\frac{-\|\boldsymbol{y}_q-\boldsymbol{p}\|^2}{h^2}\right) \quad , \tag{1.3}$$

with w and h as scaling factors. The equation 1.2 can be converted to equation above, if $\boldsymbol{\Sigma}$ is a diagonal covariance matrix *i.e.*

$$\boldsymbol{\Sigma} \,=\, \begin{bmatrix} \sigma_x & 0 & 0 \\ 0 & \sigma_y & 0 \\ 0 & 0 & \sigma_z \end{bmatrix} \quad .$$

For diagonal covariance matrices, the $\boldsymbol{\Sigma}$ can be splitted as $\boldsymbol{\Sigma}^{-1/2} \times \boldsymbol{\Sigma}^{-1/2}$, which allows transforming the exponential term in equation 1.2 as follows,

$$\begin{aligned} \left(\boldsymbol{x}_q-\boldsymbol{m}'\right)^T\, \boldsymbol{\Sigma}^{-1}\, \left(\boldsymbol{x}_q-\boldsymbol{m}'\right) &= \left(\left(\boldsymbol{x}_q-\boldsymbol{m}'\right)^T \boldsymbol{\Sigma}^{-1/2}\right) \times \left(\boldsymbol{\Sigma}^{-1/2}\left(\boldsymbol{x}_q-\boldsymbol{m}'\right)\right) \\ &= \left(\boldsymbol{\Sigma}^{-1/2}\left(\boldsymbol{x}_q-\boldsymbol{m}'\right)\right)^T \times \left(\boldsymbol{\Sigma}^{-1/2}\left(\boldsymbol{x}_q-\boldsymbol{m}'\right)\right) \\ &= \|\boldsymbol{\Sigma}^{-1/2}\left(\boldsymbol{x}_q-\boldsymbol{m}'\right)\|^2 \quad . \end{aligned}$$

Replacing the modified exponential term in equation 1.2, and comparing it to equation 1.3 results in:

$$h = \sqrt{2}; \quad w = \phi'$$
$$\boldsymbol{y}_q = \boldsymbol{\Sigma}^{-1/2}\boldsymbol{x}_q; \quad \boldsymbol{p} = \boldsymbol{\Sigma}^{-1/2}\boldsymbol{m}' \quad .$$

These replacements allows us to compute expected image using *figtree* for any Gaussian kernel with diagonal covariance matrix.

1.2 Thesis Organization

The project is described in detail in the following chapters. The existing research work or projects related to PSF approximation using different methods are described in the *Chapter 2*. Following this chapter, the formulation of optimization problem and algorithm used to solve the problem is illustrated in *Chapter 3*. The testing of algorithm designed for different datasets and the results obtained is presented in the *Chapter 4*. The thesis ends with the discussion and conclusion of this project in *Chapter 5*.

Chapter 2

Related Work

Accurate knowledge of an imaging systems PSF is crucial for generation of synthetic images. For fluorescence microscopy, various methods for PSF estimations based on either theoretical models or experimental measurements are available. These methods range from estimation in digital to analog domain. The estimation in digital domain is relatively easy and efficient due to gridded convolution. The methods in analog domain are also divided into parametric and non-parametric approaches.

The method by Cole et al. [5] is based on a non-parametric approach by recording the 3-D image of sub-resolution point sources. The research outlines a procedure for collecting and analyzing PSFs. It describes how to prepare fluorescent micro-sphere samples, set up a confocal microscope to properly collect 3D confocal image data and perform PSF measurements. A vast number of methods use a certain model for PSF and try to fit the data to estimate parameters of the model. The research by Aguet et al. [6] uses a theoretical model for PSF, based on systems' optical parameters and properties or measures 3D PSF experimentally. Due to shortcomings in both approaches, this paper proposes a combination of both approaches that aims at providing a theoretical representation of actual experimental conditions. This is achieved by fitting a parametric theoretical model to experimental measurements. They use a reduced-parameter version of the scalar model proposed by Gibson and Lanni. They use a Maximum-likelihood formulation for estimation of parameters. A similar approach is used by Kirshner et al. [7]. They also utilize the Gibson and Lanni model but use a least squares PSF fitting framework, instead of Maximum-likelihood.

Similar to our approach, the method described in Zhang et al [8] also use Gaussian approximations to model fluorescence microscope point spread functions.

Chapter 3

Bayesian Inverse Problem

As described earlier, the use of virtual source framework to generate realistic synthetic images requires PSF to be represented in terms of Gaussian kernels. This requires formulation of inverse problem to estimate PSF and source parameters. The stack of images generated using complex PSF generators or real microscopic images for point source forms dataset for our estimation problem. The image value at each pixel along with location of pixel grid in real world co-ordinate system is available to us. Given the dataset, we model our problem as *Bayesian inverse problem*.

The Bayesian perspective make use of Bayes theorem for inference, and focuses on posterior probability i.e. probability distribution of the parameters/model after considering the observed data. The posterior probability is calculated as follows:

$$posterior = \frac{likelihood \times prior}{evidence} \quad .$$

The prior is probability of model even before any data is observed i.e. $P(model)$, while the evidence is probability of data observed. The likelihood is probability of observed data given a specific model i.e. $P(data|model)$. The calculation of posterior probability is quite complex because of difficulty in computing evidence. Due to this reason, we do not try to solve full Bayesian inverse problem. Instead, for a given stack of images, the evidence is fixed and can be dropped to get Maximum a posteriori (MAP) estimate of the model. This modifies posterior being directly proportional to likelihood and prior. The prior is introduced by restricting our choice of models to multivariate Gaussian kernels with isotropic covariance. This is in coherence with use of IFGT for computing convolution Let $\boldsymbol{M}$ to be set of Gaussian kernels, the MAP estimate, $\boldsymbol{m^*}$, for model is calculated as follow:

$$\boldsymbol{m^*} = \arg\max_{\boldsymbol{m} \in \boldsymbol{M}} P(data|model) \quad .$$

The negative log of likelihood of R.H.S., nll converts it into a minimization problem. The inverse problem for estimating PSF and source parameters modifies to a optimization problem for minimizing nll, as given below:

$$\boldsymbol{m}^* = \arg\inf_{\boldsymbol{m} \in \boldsymbol{M}} \mathrm{nll}(data|model) \quad .$$

Later in this section, we formulate the nll function for our dataset, model the optimization problem for our sparse approximation and finally, an algorithm is designed to represent PSF as sparse convex combination of Gaussian kernels.

3.1 Negative log-likelihood

The modeling of nll function makes use of forward problem (equation number) to get expected image value for a given PSF model. Since, the PSF in our inverse problem is estimated using Gaussian kernels, we can formulate nll for a single Gaussian for simplification. Also, we try to make nll function dependent on a single parameter in the Gaussian kernel by assuming source intensity and position to be known, and the kernel with zero mean ([0 0 0]) and isotropic covariance, $\sigma^2 I$. This helps to visualize various effects on our optimization problem eg. effect of quality of observed data(SNR). The expected image value at each pixel can be calculated for any σ. The probability of observing a value at each pixel is represented by Poisson distribution, with mean equivalent to expected image value as function of σ. Assuming the probability for image value at each pixel to be independent , the joint probability for all pixels in the stack of images can be written as:

$$P(\boldsymbol{S}) = \prod_{i_q \in \boldsymbol{S}} P(i_q) \quad ,$$

where $\boldsymbol{S} = \{i_1, i_2 ..... i_K\}$, set of image value at all pixels and $q \in \mathcal{P}$, $\mathcal{P}$ being set of all pixels. The probability of observing image value, i_q, at pixel q is:

$$P(i_q) = \frac{{\mu_q}^{i_q}}{i_q!} \exp(-\mu_q) \quad ,$$

where μ_q is expected image value at each pixel. The μ_q is dependent on σ of Gaussian kernel. Taking negative log of joint probability, we get nll as:

$$\text{nll}(\boldsymbol{S}|\sigma) = -\log\left(P(\boldsymbol{S}|\sigma)\right) = \sum_{q \in \mathcal{P}} \left(\log(i_q!) + \mu_q(\sigma) - i_q \log(\mu_q(\sigma)) \right) \quad .$$

In equation above, the first term in summation do not depend on model parameter, σ, and thus, is constant for a given image. Thus, for obtaining optimal value of σ, first term can be ignored. This reduces the function to normalized nll :

$$\text{nll}(\boldsymbol{S}|\sigma) = \sum_{q \in \mathcal{P}} \left(\mu_q(\sigma) - i_q \log(\mu_q(\sigma)) \right) \quad .$$

The formulation gives us nll of a particular data as a function of σ. In general, the position and intensity of point source are not known and the PSF can be represented by convex summation of Gaussian kernels. Considering a model, $\boldsymbol{M}$, consisting of different Gaussian kernels, the general equation becomes:

$$\text{nll}(\boldsymbol{S}|\boldsymbol{M}) = \sum_{q \in \mathcal{P}} \left(\mu_q(\boldsymbol{M}) - i_q \log(\mu_k(\boldsymbol{M})) \right) \quad . \tag{3.1}$$

3.1.1 Effect of image quality

The quality of image we observe, can be measured in terms of signal-to-noise ratio (SNR). We can visualize this effect by generating realistic synthetic images using single Gaussian kernel with isotropic covariance. The high value of SNR is simulated using high value of source intensity in comparison to background noise. The image generated for different SNR are shown in figure 3.1. Single slide is generated keeping source in image plane to have better visualization of effect of SNR. Next, we plot nll for different values of SNR in figure 3.2(a). It can be observed that the curve gets flattened for low values of SNR. The other behavior we observed was effect of σ of Gaussian kernel with a fixed SNR. It can be observed that even with high SNR, the curve tends to become flat for higher values of σ, as shown in figure 3.2(b).

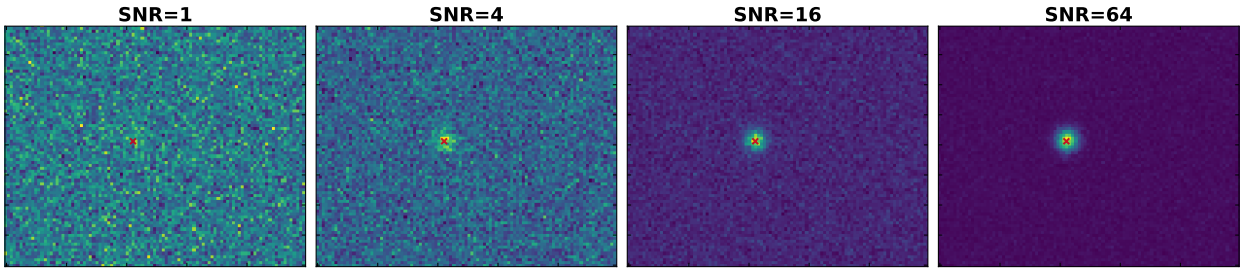
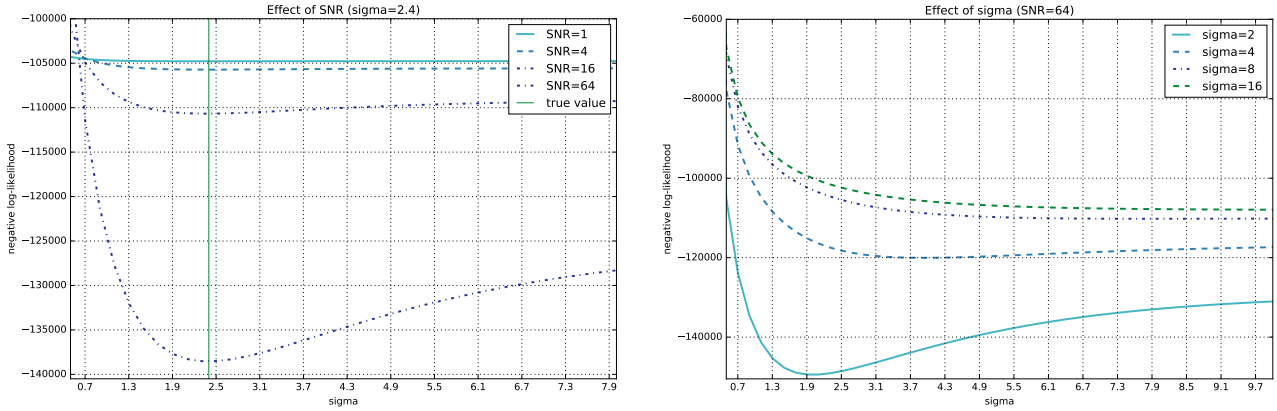


Figure 3.1: Synthetic images for different SNR



a) Effect of SNR

b) Effect of σ

Figure 3.2: Plot of nll for different SNR and $sigma$

3.2 Structure of solution space

In this project, the main target is to obtain a representation of PSF with limited number of Gaussians. The ideal solution for optimization problem in equation 3.1 will be to use Expectation-Maximization (EM) algorithm. With number of Gaussians fixed, we can obtain optimal means and variances for the Gaussians for our optimization problem. The problem with EM is absence of selectiveness i.e. to improve the PSF representation, we have to increase the number of Gaussians and re-run the EM algorithm. This is highly computationally expensive for stack of microscopic images, which are huge in size. In addition, the algorithm tries to estimate exact means and covariance matrices of Gaussian kernels, while we are restricted to diagonal covariance matrices for IFGT, as explained in Section 1.1.2.

Instead of searching for means and covariance in complete solution space, we propose to use quantized solution space in form of a **dictionary** . We construct a **dictionary** of expected means and covariances and try to represent PSF as sparse convex combination of kernels present in the **dictionary** . The optimization problem reduces to finding optimal value of intensity/location of point source and weight vector for convex combination of kernels in **dictionary**. We formulate the optimization problem to find optimal weights.

3.3 Formulation of optimization problem

Let us assume that the **dictionary** contains N Gaussian kernels for PSF approximation with means, $[\boldsymbol{m}_1, \boldsymbol{m}_2.....\boldsymbol{m}_n]$ and diagonal covariance matrices, $[\boldsymbol{\Sigma}_1, \boldsymbol{\Sigma}_2.....\boldsymbol{\Sigma}_n]$. Let $[w_1, w_2.....w_n]$ be the weights for convex combination of kernels, forming weight vector, $\boldsymbol{w}$, the PSF approximated using N Gaussian kernels, K_N can be written as:

$$K_N(\boldsymbol{x}) \;=\; \sum_{k=1}^{N} w_k \, f_k(\boldsymbol{x}) \quad , \tag{3.2}$$

where,

$$f_k(\boldsymbol{x}) = \; c_k \; \exp\left(\frac{-(\boldsymbol{x} - \boldsymbol{m}_k)^T \, \boldsymbol{\Sigma}_k^{-1} \, (\boldsymbol{x} - \boldsymbol{m}_k)}{2} \right) \quad .$$

For convex combination of kernels, the weights should belong to probability simplex, Δ. This introduces constraint on weight vector as:

$$\boldsymbol{w} \in \Delta \implies w_k \geqslant 0 \quad \text{and} \quad \sum_{k=1}^{N} w_k = 1 \quad .$$

For expected image, μ, with M number of pixels, the value at each pixel ($\boldsymbol{x}_q$), given source intensity, ϕ and position, $\boldsymbol{x}_s$, becomes :

$$\begin{aligned} \mu(\boldsymbol{x}_q) \; &= C_0 \, \phi \, K_N(\boldsymbol{x}_q - \boldsymbol{x}_s) \quad \text{for} \quad C_0 \;=\; T \, A \, \text{and} \;\; q \;\in\; 1, 2....M \\ &= C_0 \, \phi \, \sum_{k=1}^{N} w_k \, f_k(\boldsymbol{x}_q - \boldsymbol{x}_s) \quad . \end{aligned}$$

For an observed image, $\boldsymbol{I}$, taking i_q as value at each pixel in $\boldsymbol{I}$, nll function becomes:

$$\begin{aligned} \text{nll}(.) &= \sum_{\boldsymbol{x}_q} \left(\mu(\boldsymbol{x}_q) - i_q \, \log(\mu(\boldsymbol{x}_q)) \right] \\ &= \sum_{\boldsymbol{x}_q} \left(C_0 \, \phi \, \sum_{k=1}^{N} w_k \, f_k(\boldsymbol{x}_q - \boldsymbol{x}_s) \; - \; i_q \, \log\left(C_0 \, \phi \, \sum_{k=1}^{N} w_k \, f_k(\boldsymbol{x}_q - \boldsymbol{x}_s) \right) \right] \quad . \end{aligned} \tag{3.3}$$

The equation for nll can be rewritten in matrix form. Let $\boldsymbol{F_q}$ be a $N \times 1$ column vector, with elements as contribution of N kernels at pixel, $\boldsymbol{x}_q$, the expected image value, μ, can be re-written as:

$$\begin{aligned} (\boldsymbol{F_q})_{\boldsymbol{k}} &= \; C_0 \, \phi \, f_k(\boldsymbol{x}_q - \boldsymbol{x}_s) \\ \mu(\boldsymbol{x}_q) = C_0 \, \phi \, \sum_{k=1}^{N} w_k \, f_k(\boldsymbol{x}_q - \boldsymbol{x}_s) \; &= \phi \, \boldsymbol{F_q}^T \, \boldsymbol{w} \quad . \end{aligned}$$

Arranging μ as a $M \times 1$ column vector, with elements as $\mu(\boldsymbol{x}_q)$ and replacing $\mu(\boldsymbol{x}_q)$ from equation above, the expected image can be computed as:

$$\mu = \phi \begin{bmatrix} \boldsymbol{F_1}^T \\ \boldsymbol{F_2}^T \\ . \\ . \\ \boldsymbol{F_M}^T \end{bmatrix} \boldsymbol{w} = \phi \boldsymbol{D} \boldsymbol{w} \quad .$$

The dictionary matrix, $\boldsymbol{D}$, is a $M \times N$ matrix, which contains contribution of each kernel for all pixels. The nll equation can be rewritten in terms of observed image, $\boldsymbol{I}$ and $\boldsymbol{Dw}$ as follows:

$$\mathrm{nll}(\phi,\, \boldsymbol{w}) \;= \sum_{x_q} \Big(\phi \boldsymbol{Dw} - \boldsymbol{I}.\log(\phi \boldsymbol{Dw}) \Big) \quad . \tag{3.4}$$

3.4 Sparse approximation of PSF

The optimization problem is modified to obtain an optimal sparse convex combination to achieve a balance between accuracy and efficiency. The aim is to obtain a good approximation of PSF with limited number of kernels in affordable computational cost. The representation can be obtained in various ways by introducing a cost for cardinality of weight vector. The optimization can be formulated as minimizing the cardinality of weight vector with a upper bound on nll-

$$\inf_{\substack{\mathrm{nll}(\phi,\, \boldsymbol{w}) \leqslant \epsilon \\ \boldsymbol{w} \in \Delta \\ \phi \in \mathbb{R}^+}} \mathbf{card}(\boldsymbol{w}) \quad ,$$

Or, minimizing the nll with a upper bound on cardinality of weight vector-

$$\inf_{\substack{\mathbf{card}(\boldsymbol{w}) \leqslant \mathcal{K} \\ \boldsymbol{w} \in \Delta \\ \phi \in \mathbb{R}^+}} \mathrm{nll}(\phi,\, \boldsymbol{w}) \quad ,$$

Or, adding a regularization term for cardinality of weight vector while minimizing nll-

$$\inf_{\substack{\boldsymbol{w} \in \Delta \\ \phi \in \mathbb{R}^+}} \mathrm{nll}(\phi,\, \boldsymbol{w}) \;+\; \lambda\, \mathbf{card}(\boldsymbol{w}) \quad .$$

All the objective functions defined above are non-convex optimization problems. These function are relaxed to convert them to convex optimization problems. We use the last formulation with convex relaxation of $\mathbf{card}(\boldsymbol{w})$ to $\mathbf{L}$-1 norm. The objective cost function, S, for sparse approximation becomes:

$$\mathrm{S}(\phi,\, \boldsymbol{w}) = \mathrm{nll}(\phi,\, \boldsymbol{w}) + \lambda \left\| \boldsymbol{w} \right\|_1 \quad . \tag{3.5}$$

The cost function S is optimized under constraints on $\boldsymbol{w}$ and ϕ. The optimization is difficult to solve using *CMA-ES* because it is not simple to accommodate probabilistic simplex constraint on $\boldsymbol{w}$ with in cost function. Also, use of *CMA-ES* does not provide us with flexibility to use other definitions. There, an algorithm is designed which can be easily modified for changes in objective function due to use of other definitions for sparse approximation. The equation 3.5 can be optimized for two representation of PSF: using single Gaussian and, using multiple Gaussians.

3.4.1 Single Gaussian PSF model

The simplest solution for our sparse optimization problem is to use single Gaussian. If we consider single Gaussian to represent our PSF, then the problem reduces to try various pixel centers as means in dictionary and different variances to get best representation. It is better to consider the means and variances as unknown and optimize for them. Thus, this requires optimization for 6 unknowns: intensity of point source(ϕ), mean vector (x_s, y_s, z_s), and diagonal elements i.e. σ_{xy}, σ_z, since we assume same variance in x,y direction. The problem is solved using gradient descent and *CMA-ES* algorithms.

3.4.2 Multiple Gaussian PSF model

For multiple Gaussian case, we try to find optimal values for ϕ and $\boldsymbol{w}$. We can have two variants to solve this problem: one, when ϕ is required and other, when ϕ is not directly needed.

1. ϕ is required: The multiplication of ϕ and $\boldsymbol{w}$ in the cost function, S, makes it to be non-convex in ϕ and $\boldsymbol{w}$ together. Therefore, they are solved iteratively using Alternating Minimizers algorithm:

$$\phi^{k+1} \;=\; \arg\inf_{\phi} \mathrm{S}(\phi,\, \boldsymbol{w}^k) \quad , \tag{3.6}$$

$$\boldsymbol{w}^{k+1} \;=\; \arg\inf_{\boldsymbol{w}} \mathrm{S}(\phi^{k+1},\, \boldsymbol{w}) \quad . \tag{3.7}$$

2. ϕ not required: For estimation of PSF, we do not directly need ϕ. If we combine ϕ and $\boldsymbol{w}$ as $\boldsymbol{w}'$ and replace this to get nll as:

$$\mathrm{nll}(\boldsymbol{w}') \;= \sum_{x_q} \Big(\boldsymbol{D}\boldsymbol{w}' - \boldsymbol{I}.\log(\boldsymbol{D}\boldsymbol{w}') \Big) \quad . \tag{3.8}$$

It can be observed that equation 3.7 and equation 3.8 tries to optimize for $\boldsymbol{w}$ and $\boldsymbol{w}$' only, being independent of ϕ. We use ADMM to solve both of these equations as shown in Section 3.7. The solution from equation 3.7 gives a weight vector, $\boldsymbol{w}$, in simplex space, while, the optimized value of $\boldsymbol{w}$' from equation **??** needs to be normalized for getting a convex combination of kernels.

3.5 Estimation of background intensity

The expected image value at each pixel is not only because of photons emitted by point sources, as seen in Section 1.1.1. In real images, even when there in not point source, the background is not absolutely dark i.e. the image value at each pixel is not zero. Thus, this background intensity, ϕ_b contribute to expected image value in addition to point sources. The expected image value, μ_q in nll function is replaced by $\mu_q + \phi_b$. In addition to ϕ and $\boldsymbol{w}$, the nll depends on ϕ_b. modifies to:

$$nll(\phi,\, \boldsymbol{w},\, \phi_b) = \sum_{q \in \mathcal{P}} [\, \mu_q(\phi,\, \boldsymbol{w}) + \, \phi_b \, - \, i_q \, \log(\mu_q(\phi,\, \boldsymbol{w}) + \, \phi_b)] \quad .$$

The optimization problem for ϕ_b can be solved independently, as it is independent of intensity of point sources. We obtain optimal value of ϕ_b by equating the differential of nll with respect to ϕ_b to zero as given below:

$$\sum_{q \in \mathcal{P}} [\, 1 \, - \, \frac{i_q}{\mu_q + \phi_b}] = 0 \quad .$$

For real images generated by point sources, the sources can be considered to be of compact support i.e. no effect of source at pixel far from point source. Assuming, the image value due to point source in such pixels, to be approximately zero i.e. $\mu_q \;\approx\; 0$, the closed solution for ϕ_b becomes:

$$\phi_b = \frac{1}{|\mathcal{P}'|} \sum_{q \in \mathcal{P}'} i_q \quad , \tag{3.9}$$

where $\mathcal{P}'$ is set of pixels away from source.

3.6 Solving optimization problem for single Gaussian PSF model

The optimization problem tries to estimate 6 unknowns for this model. We use gradient descent and *CMA-ES* algorithm to find optimal solution.

3.6.1 Gradient descent with multiple starts

The *Optimize* library from *SciPy* supports various methods for optimization based on gradient descent. The *Optimize* library provides options to use different methods for minimization for eg. L-BFGS-B algorithm [9] for bound constrained minimization. The library, *lhs* (Latin-hypercube-sampling) is used to provide multiple start points distributed evenly in hyperspace of these parameters. The gradient for the nll function is calculated using *theano* library. The gradient is provided as a argument to *Optimize* to increase speed of convergence.

3.6.2 CMA-ES: A black-box optimization algorithm

The functionality from *Optimize* tries to solve local optimization problem, where as *CMA-ES* tries to solve global optimization problem. The *CMA-ES* (Covariance Matrix Adaptation Evolution Strategy) is an evolutionary algorithm which works well for non-linear and non-convex black-box optimization problems in continuous domain [10]. *CMA-ES* does not require objective function to be differentiable for getting results. In addition, *CMA-ES* is a second order approach, which shall produce results better than *Optimize*. The optimization is tried to get optimal values of same 6 parameters defined above.The functionality from *CMA-ES* is tried with an image produced using single Gaussian with isotropic covariance. The image and sources are taken to be in same plane. The result of the optimization for 4 different starting points is:

Start	σ_{xy}	σ_z	ϕ	x_s	y_s	z_s
1	2.01	1.84	42.23	37.12	34.87	4.46
2	2.01	1.68	38.76	37.12	34.87	5.52
3	2.01	1.53	37.17	37.12	34.87	5.67
4	2.01	3.63	97.48	37.12	34.87	2.71

The *CMA-ES* shows different results for different starting points. The reason for this result is use of single slide of image for optimization. The image data does not provide any information about variation in z-direction and therefore, σ_z and z_s depend on starting points. The variation of σ_z is compensated by corresponding change in ϕ. This also shows that stack of images to be used for better results. The use of *CMA-ES* for single Gaussian tends to perform fine as we optimize for only 6 parameters. The extension to multiple Gaussians increases the number of unknown parameters substantially. The estimation with 100 Gaussians requires optimization for 600 parameters for given stack of images. This is really slow and depends heavily on starting points. Because of this reason also, the approximation of PSF for multiple Gaussian case is done by use of **dictionary**.

3.7 Solving optimization problem for multiple Gaussian PSF model

The optimization for multiple Gaussians is solved using Alternating Direction Method of Multipliers (ADMM), an algorithm that is well suited for convex optimization. The ADMM is implemented to optimize for weights

of kernels. The ADMM is also converges toward optimal solution from starting iterations only. The implementation of algorithm is described in Boyd et al [11]. ADMM works in form of a *decomposition-coordination* procedure, in which the solutions to small local subproblems are coordinated to find a solution to a large global problem. ADMM attempts to blend the benefits of dual decomposition and augmented Lagrangian methods for constrained optimization. First, we solve for optimization problem in equation 3.7, and later modify our algorithm for problem in equation 3.8. According to ADMM, the minimization of cost function, S, is modified in following steps:

1. Introduction of indicator function for simplex constraint

$$\inf_{\boldsymbol{w}\in\Delta} \mathrm{S}(\boldsymbol{w}) \;=\; \inf_{\boldsymbol{w}} \mathrm{nll}(\boldsymbol{w}) + \lambda \left\|\boldsymbol{w}\right\|_1 + \mathrm{i}_\Delta(\boldsymbol{w})$$

$$\text{where,} \quad \mathrm{i}_\Delta(\boldsymbol{w}) = \begin{cases} 0, & \text{if } \boldsymbol{w} \in \Delta \\ \infty, & \text{otherwise} \end{cases}$$

2. Splitting the cost function
   The three parts of the cost function are made independent with introduction of new variables:

$$\begin{aligned} \boldsymbol{v}_1 \;&=\; \boldsymbol{D}\boldsymbol{w} \implies \mathrm{S}_1(\boldsymbol{v}_1) = \mathrm{nll}(\boldsymbol{v}_1) \quad , \\ \boldsymbol{v}_2 \;&=\; \boldsymbol{w} \implies \mathrm{S}_2(\boldsymbol{v}_2) = \lambda \left\|\boldsymbol{v}_2\right\|_1 \quad , \\ \boldsymbol{v}_3 \;&=\; \boldsymbol{w} \implies \mathrm{S}_3(\boldsymbol{v}_3) = \mathrm{i}_\Delta(\boldsymbol{v}_3) \quad . \end{aligned}$$

   For given equality constraints, the optimization problem changes to:

$$\inf_{\substack{\boldsymbol{v}_1,\boldsymbol{v}_2,\boldsymbol{v}_3 \\ \boldsymbol{v}_1=\mathbf{D}\mathbf{w} \\ \boldsymbol{v}_2=\boldsymbol{w},\, \boldsymbol{v}_3=\boldsymbol{w}}} \mathrm{S}_1(\boldsymbol{v}_1) + \mathrm{S}_2(\boldsymbol{v}_2) + \mathrm{S}_3(\boldsymbol{v}_3) \quad .$$

3. Augmented Lagrangian
   Introducing regularization using dual variables $\boldsymbol{b}_1$, $\boldsymbol{b}_2$ and $\boldsymbol{b}_3$ to accomodate equality constraints:

$$\begin{aligned} \mathrm{AS}(\boldsymbol{w},\, \boldsymbol{v},\, \boldsymbol{b}) = \mathrm{S}_1(\boldsymbol{v}_1) + \mathrm{S}_2(\boldsymbol{v}_2) + \mathrm{S}_3(\boldsymbol{v}_3) + \\ \frac{1}{2\gamma} \left\|\boldsymbol{b}_1 + \boldsymbol{D}\boldsymbol{w} - \boldsymbol{v}_1\right\|_2^2 + \frac{1}{2\gamma} \left\|\boldsymbol{b}_2 + \boldsymbol{w} - \boldsymbol{v}_2\right\|_2^2 + \frac{1}{2\gamma} \left\|\boldsymbol{b}_3 + \boldsymbol{w} - \boldsymbol{v}_3\right\|_2^2 \quad . \end{aligned}$$

4. ADMM or Alternating Split Bregmann
   The steps below are iterated to obtain an optimal solution.
   Primal updates:

$$\boldsymbol{w}^{k+1} \;=\; \arg\inf_{\boldsymbol{w}} \mathrm{AS}(\boldsymbol{w}, \boldsymbol{v}_1^k, \boldsymbol{v}_2^k, \boldsymbol{v}_3^k, \boldsymbol{b}^k) \tag{3.10}$$

$$\boldsymbol{v}_1^{k+1} \;=\; \arg\inf_{\boldsymbol{v}_1} \mathrm{AS}(\boldsymbol{w}^{k+1}, \boldsymbol{v}_1, \boldsymbol{v}_2^k, \boldsymbol{v}_3^k, \boldsymbol{b}^k) \tag{3.11}$$

$$\boldsymbol{v}_2^{k+1} \;=\; \arg\inf_{\boldsymbol{v}_2} \mathrm{AS}(\boldsymbol{w}^{k+1}, \boldsymbol{v}_1^{k+1}, \boldsymbol{v}_2, \boldsymbol{v}_3^k, \boldsymbol{b}^k) \tag{3.12}$$

$$\boldsymbol{v}_3^{k+1} \;=\; \arg\inf_{\boldsymbol{v}_3} \mathrm{AS}(\boldsymbol{w}^{k+1}, \boldsymbol{v}_1^{k+1}, \boldsymbol{v}_2^{k+1}, \boldsymbol{v}_3, \boldsymbol{b}^k) \tag{3.13}$$

   Dual updates:

$$\begin{aligned} \boldsymbol{b}_1^{k+1} \;&=\; \boldsymbol{b}_1^k + \boldsymbol{D}\boldsymbol{w}^{k+1} - \boldsymbol{v}_1^{k+1} \\ \boldsymbol{b}_2^{k+1} \;&=\; \boldsymbol{b}_2^k + \boldsymbol{w}^{k+1} - \boldsymbol{v}_2^{k+1} \\ \boldsymbol{b}_3^{k+1} \;&=\; \boldsymbol{b}_3^k + \boldsymbol{w}^{k+1} - \boldsymbol{v}_3^{k+1} \end{aligned}$$

3.7.1 Solution for subproblems

The optimization problems, equation 3.10 to 3.13 in primal update step are treated as local subproblems. The subproblems are solved independently considering other variables to be constant for that subproblem. This other advantage with ADMM is that, even if each subproblem is not solved exactly, the algorithm still converges. Also, the order of primal updates does not matter for convergence of algorithm.
$\boldsymbol{w}$ **subproblem:** The equation 3.10 finds the optimal value of $\boldsymbol{w}$ independently, and thus the equation 3.10 modifies to -

$$\boldsymbol{w}^{k+1} \quad = \quad \arg\inf_{\boldsymbol{w}} \frac{1}{2\gamma} \|\boldsymbol{b}_1{}^k + \boldsymbol{D}\boldsymbol{w} - \boldsymbol{v}_1{}^k\|_2^2 + \frac{1}{2\gamma} \|\boldsymbol{b}_2{}^k + \boldsymbol{w} - \boldsymbol{v}_2{}^k\|_2^2 + \frac{1}{2\gamma} \|\boldsymbol{b}_3{}^k + \boldsymbol{w} - \boldsymbol{v}_3{}^k\|_2^2$$

The equation is similar to regularized least squares. Differentiating the equation w.r.t. $\boldsymbol{w}$ and equating to zero for optimal solution:

$$\begin{aligned} &\boldsymbol{D}^T \left(\boldsymbol{b}_1{}^k + \boldsymbol{D}\boldsymbol{w} - \boldsymbol{v}_1{}^k\right) + \left(\boldsymbol{b}_2{}^k + \boldsymbol{w} - \boldsymbol{v}_2{}^k\right) + \left(\boldsymbol{b}_3{}^k + \boldsymbol{w} - \boldsymbol{v}_3{}^k\right) \;=\; 0 \\ &\implies \; \boldsymbol{w}^{k+1} \;=\; \left[\boldsymbol{D}^T\boldsymbol{D} + 2\boldsymbol{I}\right]^{-1} \left[\boldsymbol{D}^T \left(\boldsymbol{b}_1{}^k - \boldsymbol{v}_1{}^k\right) + \left(\boldsymbol{b}_2{}^k - \boldsymbol{v}_2{}^k\right) + \left(\boldsymbol{b}_3{}^k - \boldsymbol{v}_3{}^k\right)\right] \end{aligned}$$

$\boldsymbol{v}_1$ **subproblem:** The equation 3.11 finds the optimal value of v_1 independently, and thus the equation 3.11 reduces to -

$$\boldsymbol{v}_1^{k+1} \;=\; \arg\inf_{\boldsymbol{v}_1} \mathrm{S}_1(\boldsymbol{v}_1) + \frac{1}{2\gamma} \|\boldsymbol{b}_1{}^k + \boldsymbol{D}\boldsymbol{w}^{k+1} - \boldsymbol{v}_1\|_2^2$$

After replacing $\mathrm{S}_1(\boldsymbol{v}_1)$ by $\mathrm{nll}(v_1)$, it can be observed that both first and second term in R.H.S. are summation over all pixels. Thus solving for each pixel, q, separately, simplifies the above equation to a scalar equation. Let v_q be q^{th} element of $\boldsymbol{v}_1$, the scalar equation is as follows:

$$\arg\inf_{v_q} \mathrm{nll}(v_q) + \frac{1}{2\gamma} \left[(\boldsymbol{b}_1{}^k)_q + (\boldsymbol{D}\boldsymbol{w}^{k+1})_q - v_q\right]^2$$

Using equation 3.1, we replace $\mathrm{nll}(v_q)$ by $v_q - i_q \log(v_q)$ and differentiate w.r.t. v_q to obtain optimal value. The differentiation leads to following quadratic equation:

$$(v_q)^2 + (\gamma - (\boldsymbol{b}_1{}^k)_q - (\boldsymbol{D}\boldsymbol{w}^{k+1})_q).v_q - \gamma.i_q = 0$$

The quadratic equation above gives two possible solutions for each v_q, one positive and other negative. The image value at each pixel can only be positive and thus, only positive solution is used as optimal value.
$\boldsymbol{v}_2$ **subproblem:** The equation 3.12 finds the optimal value of $\boldsymbol{v}_2$ independently, and thus the equation 3.12 reduces to -

$$\boldsymbol{v}_2^{k+1} \;=\; \arg\inf_{\boldsymbol{v}_2} \mathrm{S}_2(\boldsymbol{v}_2) + \frac{1}{2\gamma} \|\boldsymbol{b}_2{}^k + \boldsymbol{w}^{k+1} - \boldsymbol{v}_2\|_2^2$$

After replacing $\mathrm{S}_2(\boldsymbol{v}_2)$ by $\lambda \|\boldsymbol{v}_2\|_1$, it can be observed that both first and second term in R.H.S. are summation over all Gaussian kernels. Thus solving for each kernel, n, separately, simplifies the above equation to a scalar equation. Let v_n be n^th element of $\boldsymbol{v}_2$, the scalar equation is as follows:

$$\arg\inf_{v_n} \lambda \|v_n\| + \frac{1}{2\gamma} \left[(\boldsymbol{b}_2{}^k)_n + (\boldsymbol{w}^{k+1})_n - v_n\right]^2$$

Using sub-gradient of absolute, above equation gives solution as follows:

$$v_n = \begin{cases} \left(1 - \frac{\lambda'}{\|x_n\|}\right), & \forall \ \|x_n\| \leqslant \lambda' \\ 0, & \text{else} \end{cases},$$

where, x_n equals $(\boldsymbol{b}_2{}^k)_n + (\boldsymbol{w}^{k+1})_n$ and λ' is equal to $(\lambda\,\gamma)$. The above solution works in same way as soft-shrinkage operator.

$\boldsymbol{v}_3$ **subproblem:**The equation 3.13 finds the optimal value of $\boldsymbol{v}_3$ independently, and thus the equation 3.13 reduces to -

$$\boldsymbol{v}_3^{k+1} \ = \ \arg\inf_{\boldsymbol{v}_3} \mathrm{S}_3(\boldsymbol{v}_3) + \frac{1}{2\gamma} \|\boldsymbol{b}_3{}^k + \boldsymbol{w}^{k+1} - \boldsymbol{v}_3\|_2^2$$

After replacing $\mathrm{S}_3(\boldsymbol{v}_3)$ by $\mathrm{i}_\Delta(\boldsymbol{v}_3)$, the equation can be re-written as:

$$\arg\inf_{\boldsymbol{v}_3 \in \Delta} \|\boldsymbol{b}_3{}^k + \boldsymbol{w}^{k+1} - \boldsymbol{v}_3\|_2^2$$

The constraint of probability simplex on $\boldsymbol{v}_3$ solves the above equation as projection of vector, $\boldsymbol{b}_3{}^k + \boldsymbol{w}^{k+1}$ on simplex space.

Modifying algorithm for equation 3.8: The modified weight vector in 3.8, $\boldsymbol{w}$' being multiplication of ϕ and $\boldsymbol{w}$, does not belong to simplex space anymore. The complete algorithm for the following change remains same except the simplex constraint on $\boldsymbol{w}$ is relaxed. The weights for combination need to be non-negative only, changing the convex combination to conic combination of kernels. This changes the $\boldsymbol{v}_3$ subproblem to projection on $\mathbb{R}^+$, which is easily obtained by making the non-negative weights to zero.

Chapter 4

Experiments and Results

This section explains about experimental setup to test the optimization algorithms and analyze the results obtained for them. The estimation methods are tested for different datasets. The methods are tested initially for synthetic images generated using IFGT. With the use of synthetic dataset, the true PSF model is known which allows us to compare the effect of different parameters on estimated PSF. The images are generated with pixel slides parallel to x-y plane. We have different slides separated in z-direction with bottom most slide having the pixel center of left bottom pixel aligned to origin in object space. An example of generated image is shown in figure 4.1(a). The figure shows projection of stack of images on x,y,z axis, synthesized using following parameters: 2 Gaussians, $\boldsymbol{m} = [[-2.-2.0.], [2.2.0.]]$ and diagonal of all $\boldsymbol{\Sigma}$ = $[1.51.51.5]$, the weights $\boldsymbol{w}$=$[0.4, 0.6]$ and source parameters,ϕ=1000, SNR=100 and $\boldsymbol{x}_s$=$[10.12.520.]$. The image generated is of size, $40 \times 25 \times 20$ i.e. stacks=40, height=25 and width=20.

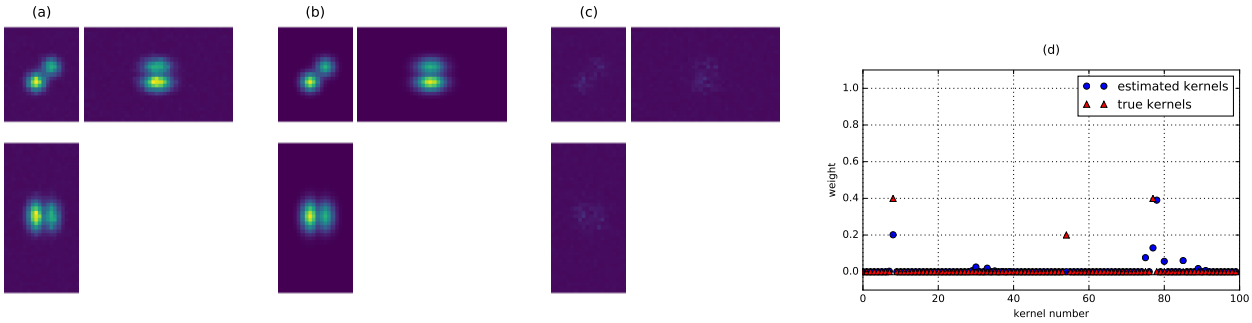


Figure 4.1: Visualization and comparison of results

A **dictionary** containing 64 mean positions including the above means is used to estimate weights. The estimated image and estimated weights are shown in figure 4.1(b) and 4.1(d), respectively. For comparing results, if the PSF model for input images is known, the performance can be measured as difference from the original PSF model, else it can be measured as difference between input image and synthesized image i.e. residual image, shown in figure 4.1(c). The error can be calculated as absolute sum of residual i.e. **L**-1 norm, or sum of square of residuals i.e. **L**-2 norm, or the maximum i.e. **L**-∞ norm.

4.1 CMA-ES vs ADMM

For synthetic images generated as explained above, the optimal weights can be obtained by optimizing equation 3.4. The equation is optimized using both, *CMA-ES* and ADMM, considering source position to be known. A **dictionary** is created with different kernel sizes containing actual centers, and fed to to both algorithms. The regularization parameter in the equation, λ is also modified to compare the speed, accuracy and sparsity of weight vector for both algorithms. The error is calculated as difference in the PSF model and the plots for error, time taken and sparsity of weight vector for λ=1, 10 are shown in figure 4.2. The

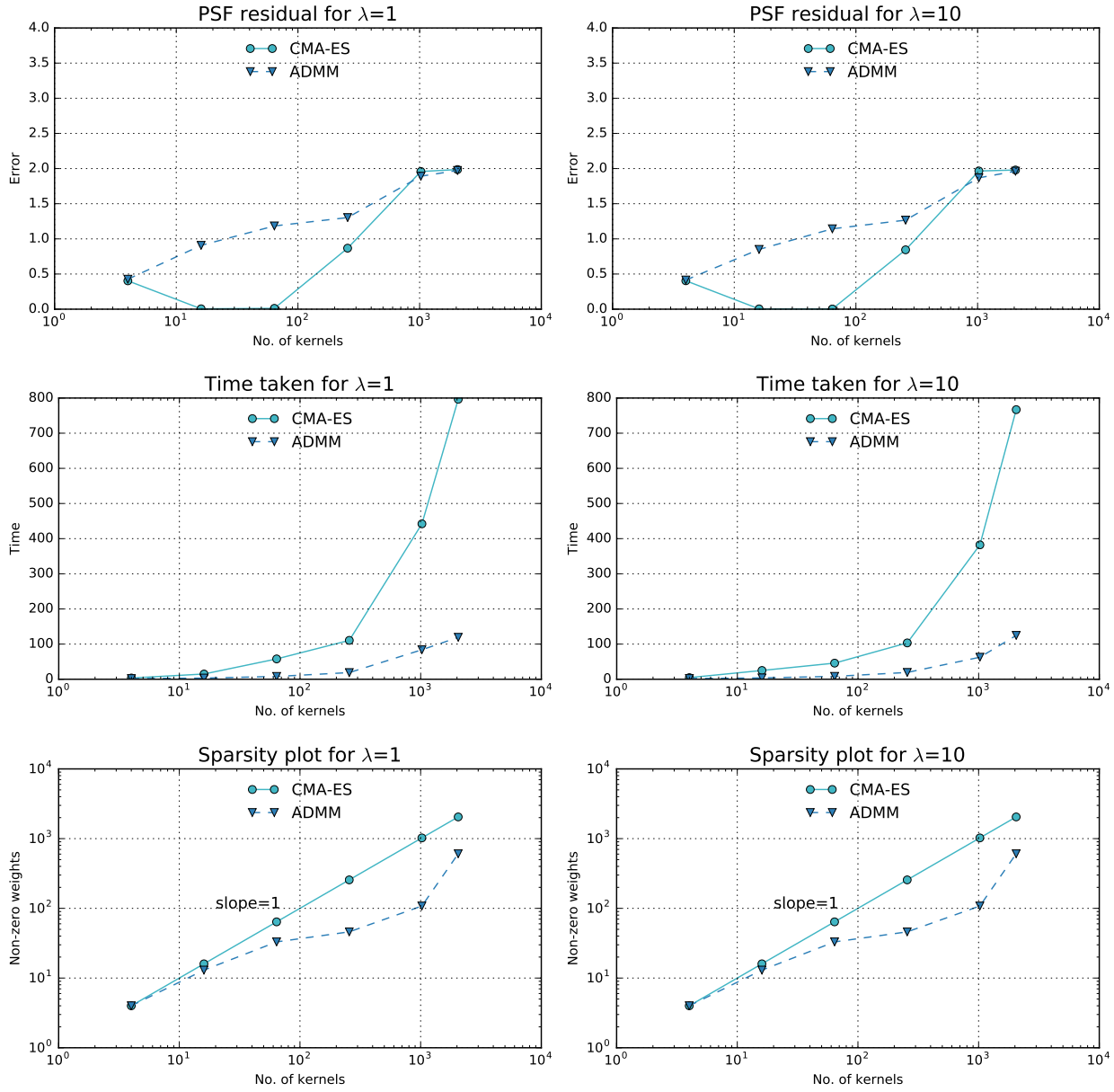


Figure 4.2: Comparison of error,speed and sparsity for CMA-ES and ADMM for λ=1, 10

results for *CMA-ES* are better than ADMM but it can be observed that there is sharp increase in time taken

by *CMA-ES* in comparison to that taken by ADMM with increase in number of kernels. Due to this reason, we can not use *CMA-ES* for huge dictionary sizes. The other difference can be observed in sparsity plots for both algorithms. The *CMA-ES* is not able to enforce sparsity and make use of whole **dictionary** for better results and thus, fails to produce sparse reconstruction.

Other benefit of using ADMM over *CMA-ES* is speed of convergence. The algorithm designed using ADMM, produces satisfactory results with in a limited number of iterations. The algorithms like *CMA-ES* converge slowly in comparison to ADMM. This can be seen in figure 4.3, as the value of nll reduces slowly for each iteration for *CMA-ES* , while it reduces sharply from first iteration itself for ADMM.

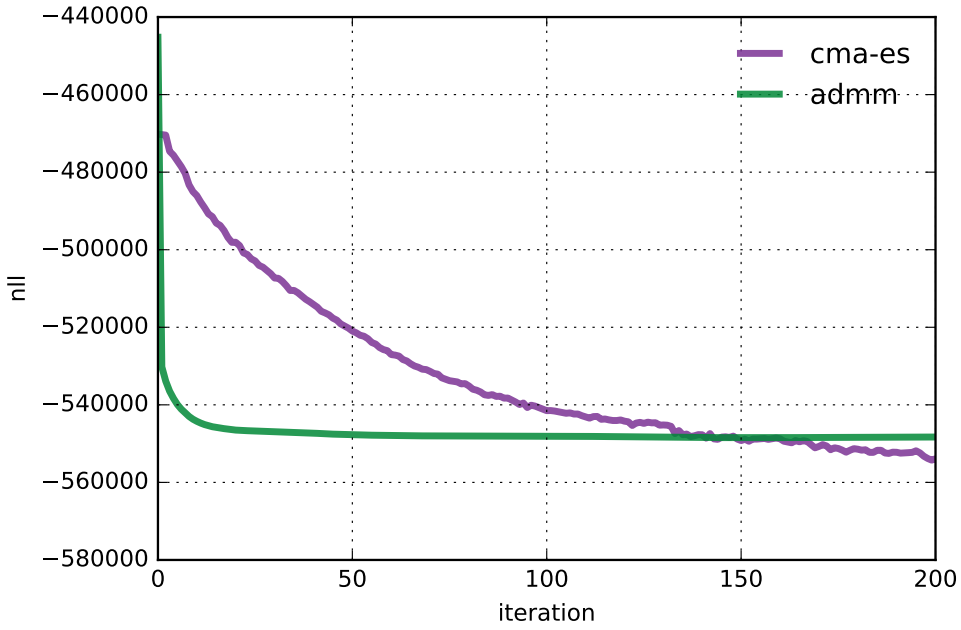


Figure 4.3: Convergence speed for CMA and ADMM

4.2 Estimation results for synthetic data generated using Gaussian PSF

We try to estimate PSF using single Gaussian approach and using ADMM. The algorithms are implemented and tested for synthetic data generated using 4 Gaussian kernels. The parameters used are: set of means, $\boldsymbol{m} = [[-2.\ -2.\ 0.], [-2.\ 2.\ 0.], [2.\ 2.\ 0.], [2.\ -2.\ 0.]]$ and diagonal of all $\boldsymbol{\Sigma}$ = $[1.5\ 1.5\ 2.5]$, the weights $\boldsymbol{w}$ = $[0.25, 0.25, 0.25, 0.25]$ and source parameters, ϕ = 1000 and $\boldsymbol{x}_s$ = $[10.\ 12.5\ 20.]$. The image shown in figure 4.4(a) is projection of generated stack.

4.2.1 Use of single Gaussian

The PSF is estimated using single Gaussian as described in Section 3.6.2. The mean and covariance of kernel, along with source intensity are estimated using *CMA-ES*. We provide certain starting points to *CMA-ES* close to bright pixels, to help it converge to position of point source. The results obtained for *CMA-ES* are given below in table below. The image reconstructed using the estimated parameters and the

residual image is shown in figure 4.4(b) and 4.4(c). The mean of Gaussian kernel can be used as position of point source for estimation using multiple Gaussians.

σ_{xy}	σ_z	ϕ	x_s	y_s	z_s
2.53	2.70	1187.2	10.03	12.43	20.05

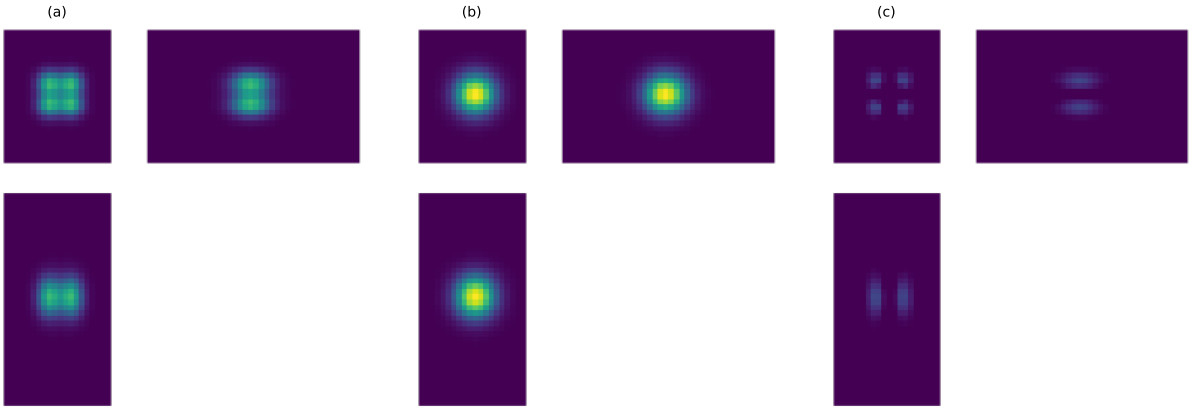


Figure 4.4: Results for single Gaussian approach: a) Input image b) Reconstructed image with estimated PSF c) Residual image

4.2.2 Use of multiple Gaussians

Using position of point source estimated above, the PSF is estimated using multiple Gaussians with the help of ADMM. Since, we have no interest in getting estimate of ϕ, we use the corresponding ADMM variant. A range of values of λ are tried to choose the one with lowest error.

Dictionary design: The algorithm requires a **dictionary** of different means, and diagonal covariances for ADMM. The result of ADMM algorithm depends on choice of dictionary. We choose means and variances, in a manner to get better approximation of our PSF, as given below:

1. The means are chosen as center of all pixels with image value greater than a threshold. The lower the threshold, more is the number of pixels, which in turn increases the size of dictionary. The threshold of $10e-4$ is chosen, which provides us with dictionary of size 376 for this case.

2. The σ_{xy} and σ_z for kernel is chosen with values same as $\mathbf{\Sigma}$. This is done to simplify the choice of kernels for ADMM. This will not be the case with real data or non-gaussian PSF.

The $error$ for different λ is shown in figure 4.5. The reconstructed image for best λ, along with corresponding residual is shown in figure 4.6.

Sparsity of weights: The choice of ADMM algorithm allows us to do fitting and selection of kernels at the same time. If we rank the kernels according to estimated weights, we have a choice of choosing N best kernels, depending on bound on $error$. The plot in figure 4.7 shows the decrease of $error$ with choosing 1 best kernel to choosing all kernels.

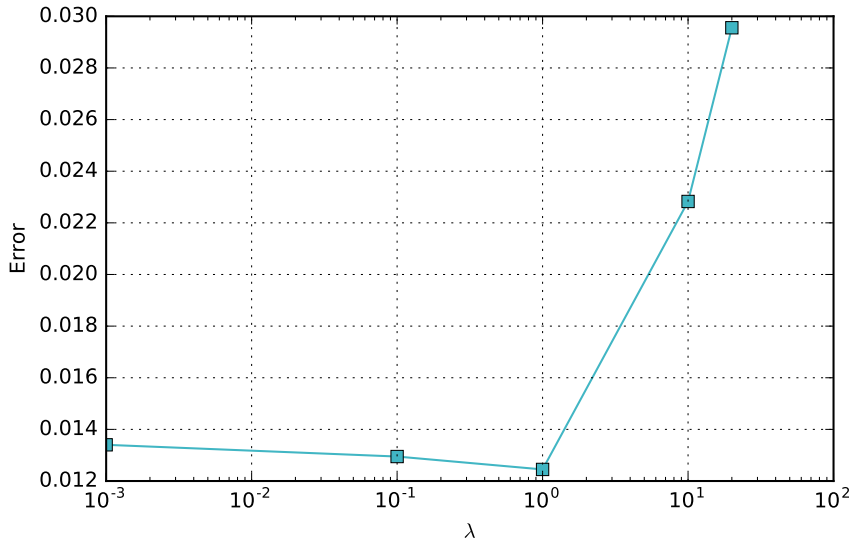


Figure 4.5: Results of ADMM for different λ

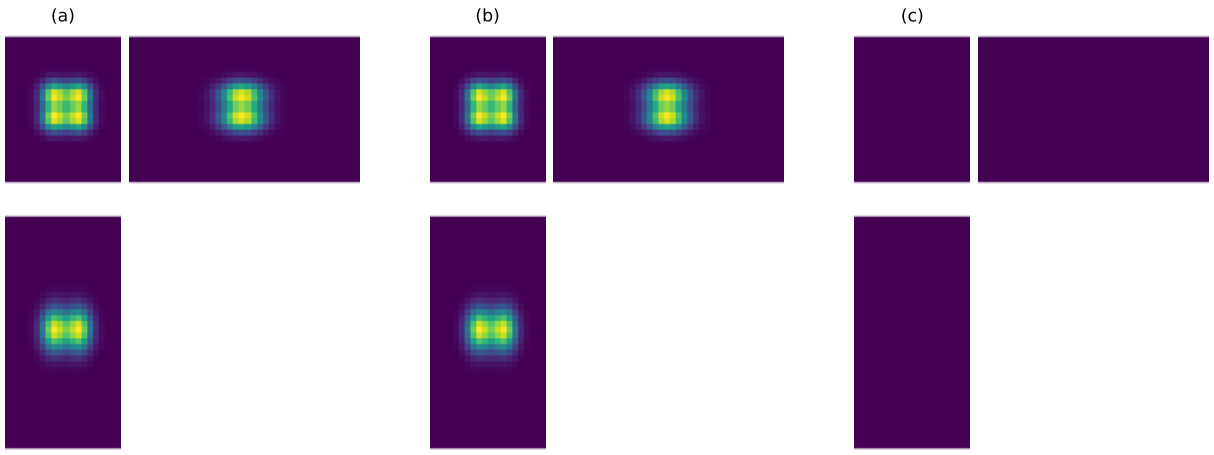


Figure 4.6: Results for ADMM (λ=1): a) Input image b) Reconstructed image with estimated PSF c) Residual image

4.3 Estimation results for synthetic data generated using non-gaussian psf

Next, we try to obtain approximation of PSF for images generated using non-gaussian PSF. The images are generated using PSF generator, developed by Kirshner and Sage [7]. We generate PSF (shown in figure 4.8(a)) using Gibson and Lanni 3D optical model in PSF generator.

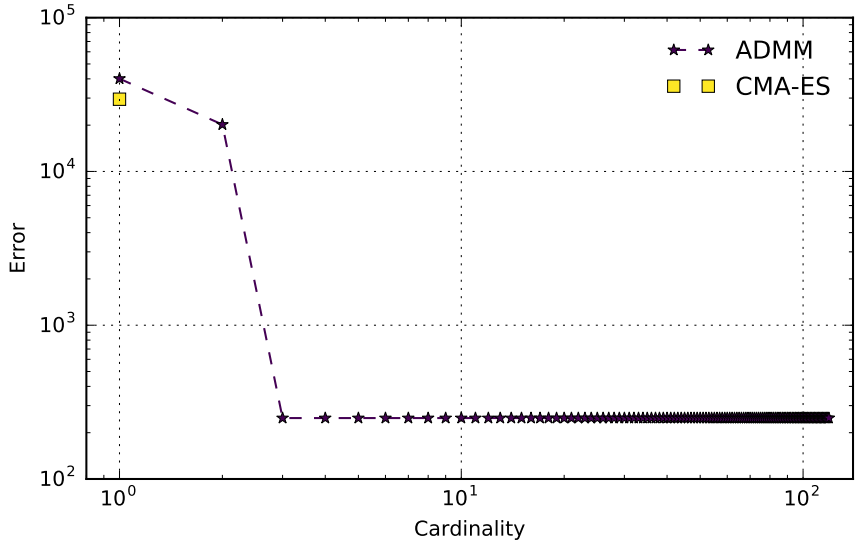


Figure 4.7: Error Vs No. of kernels with non-zero weight

4.3.1 Use of single Gaussian

The mean and variances of single Gaussian are estimated using *CMA-ES*, in the same way as for above data. The results obtained for *CMA-ES* are given below in table below. The image reconstructed using the estimated parameters and the residual image is shown in figure 4.8(b) and 4.8(c).

σ_{xy}	σ_z	ϕ	x_s	y_s	z_s
4.98	4.98	1.57	24.8	24.3	70.02

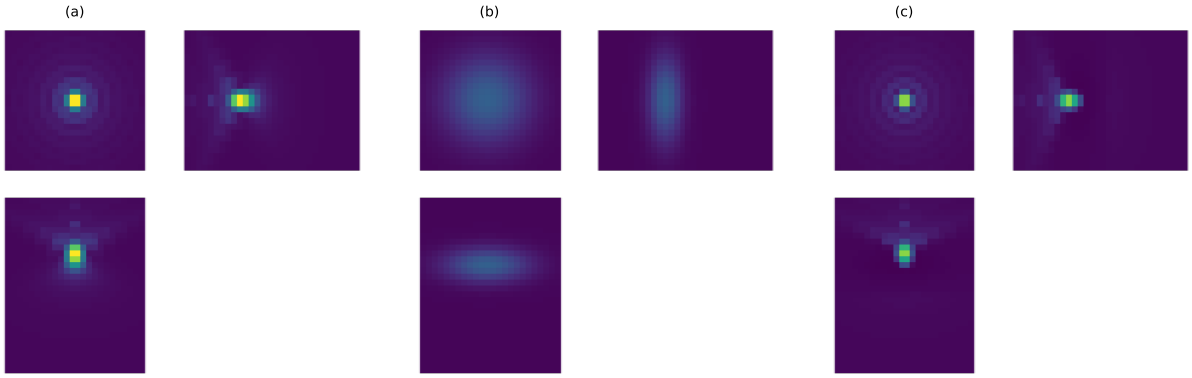


Figure 4.8: Results for single Gaussian model: a) Gibson and Lanni PSF input image b) Reconstructed image with estimated PSF c) Residual image

4.3.2 Use of multiple Gaussians

We use ADMM to approximate using multiple Gaussians. To use ADMM, we try to design our **dictionary** as given below:

1. The means are again chosen as center of all pixels with image value greater than a threshold. The threshold of $2 \times 10e - 5$ is chosen, which provides us with dictionary of size 324 for this case.

2. It is difficult to choose σ_{xy} and σ_z for kernels. We use the values provided by *CMA-ES* as upper bound and try different scales of σ_{xy} and σ_z. The **dictionary** used for this dataset uses a fixed value of covariance, for all mean positions.

The algorithm is run for a range of λ to choose appropriate value. The $error$ for different λ is shown in figure 4.9. The reconstructed image along with corresponding residual are shown in figure 4.10.

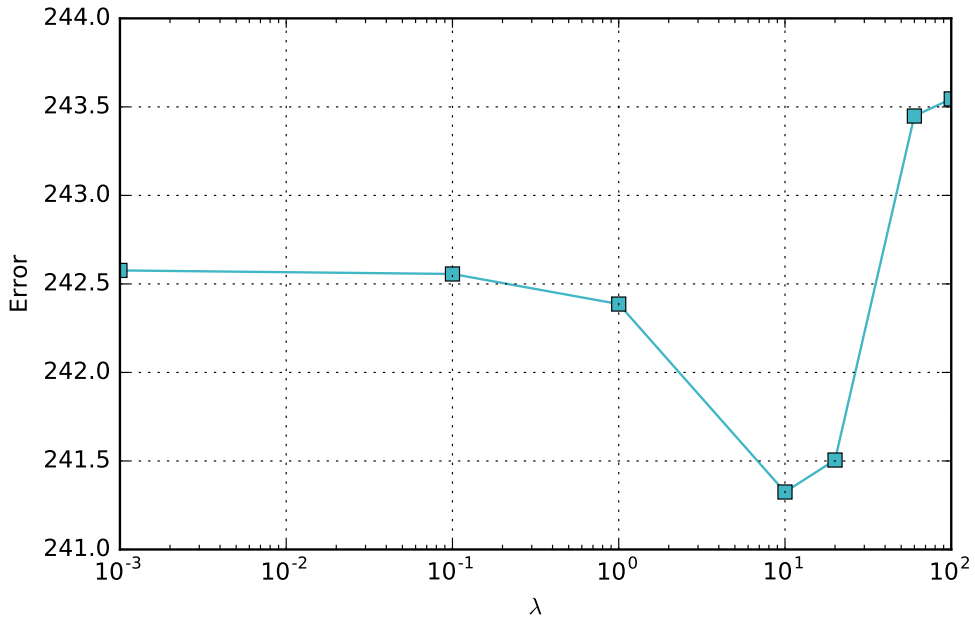


Figure 4.9: Results of ADMM for different λ

Sparsity of weights: As explained for previous dataset, the decrease of $error$ with choosing 1 best kernel to choosing all kernels, one by one, is shown in figure 4.11.

4.4 Estimation results for real bead images

Finally, we try to obtain approximation of PSF for real bead images. The images contain single bead with compact support (shown in figure 4.12(a)). The stack of images are noisy and has a dc-offset for dark pixels i.e. image value when no photons are excited. The dc-offset is taken as minimum image value among all pixels and is removed from image value at all pixels.

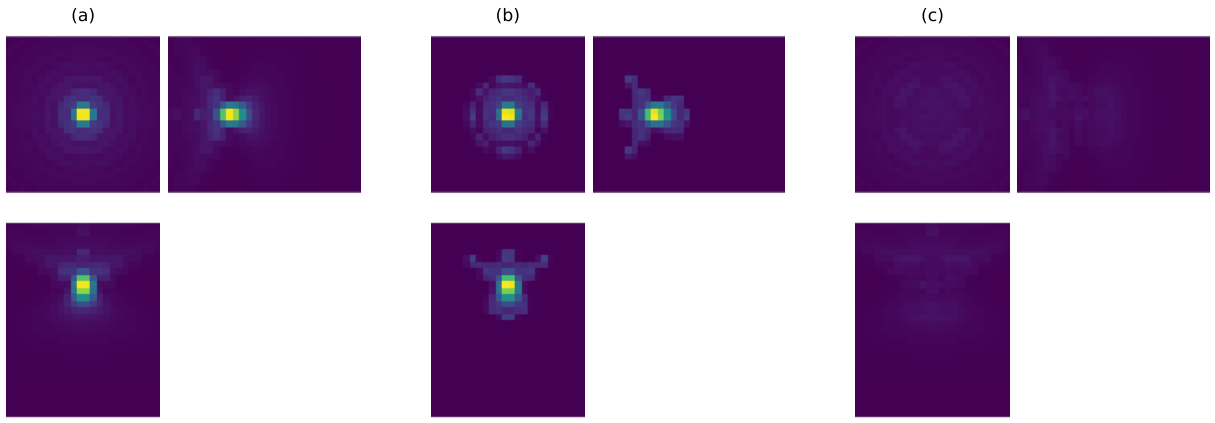


Figure 4.10: Residual Image for λ=1

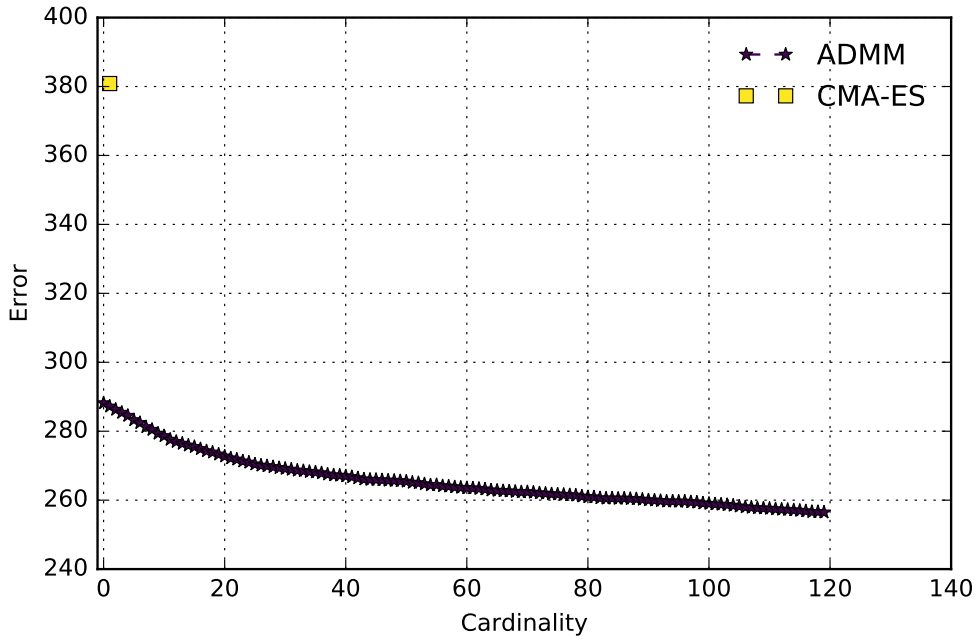


Figure 4.11: Error Vs No. of kernels with non-zero weight

4.4.1 Use of single Gaussian

The mean and variances of single Gaussian are estimated using *CMA-ES*, in the same way as for above data. The *CMA-ES* is executed with different starting points and median of these results is chosen as final result, shown in table below. The image reconstructed using the estimated parameters and the residual image is shown in figure 4.12(b) and 4.12(c).

σ_{xy}	σ_z	ϕ	x_s	y_s	z_s
0.4	0.44	0.0421	1.027	0.799	0.782

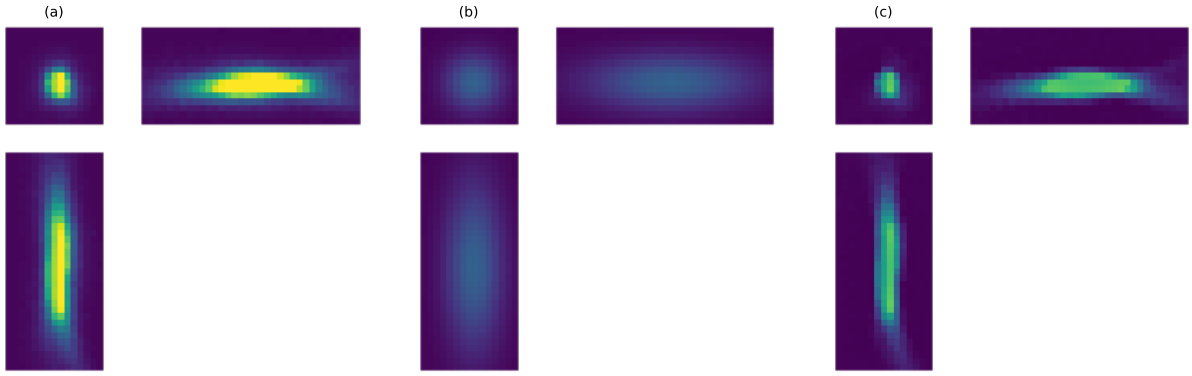


Figure 4.12: Results for single Gaussian model: a) Real bead image b) Reconstructed image with estimated PSF c) Residual image

4.4.2 Use of multiple Gaussians

We use ADMM to approximate using multiple Gaussians. To use ADMM, we try to design our **dictionary** as given below:

1. The means are again chosen as center of all pixels with image value greater than a threshold. The threshold of $2 \times 10e - 5$ is chosen, which provides us with dictionary of size 1054 for this case.

2. It is difficult to choose σ_{xy} and σ_z for kernels. We use the values provided by *CMA-ES* as upper bound and try different scales of $\sigma_x y$ and σ_z. The lower bound on variances is chosen as 2-3 pixels in image space. The **dictionary** used for this dataset uses a fixed value of covariances, for all mean positions.

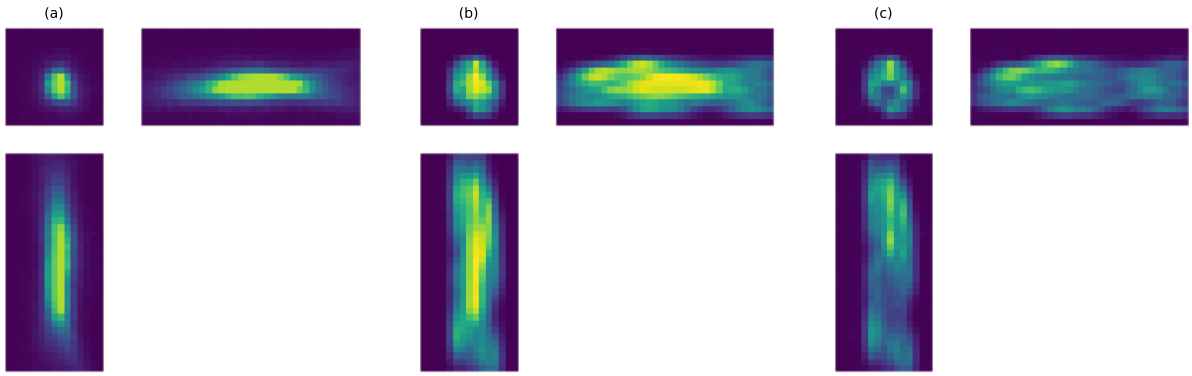


Figure 4.13: Residual Image

The reconstructed image along with corresponding residual are shown in figure 4.13. The results obtained for this case tend to overfit and needs to be corrected.

4.5 Use case: TASEP-LK model

The virtual microscope framework [2] allows us to represent PSF as a single Gaussian. We use the framework developed in thesis to extend it to more complex PSF represented with a mixture of Gaussians. We demonstrate this on the simulation data of TASEP-LK model (Totally Asymmetric Simple Exclusion Process with Langmuir Kinetics), shown in figure 4.14. We use intensities of light sources (fluorescent proteins) attached to filaments (filamentous intracellular structures, called microtubules) as point sources and, use both representation of PSF in virtual microscope framework to compute final images. Figure 4.15 demonstrates the results generated using PSF represented with one Gaussian and with the mixture of Gaussians learned from a real bead image.

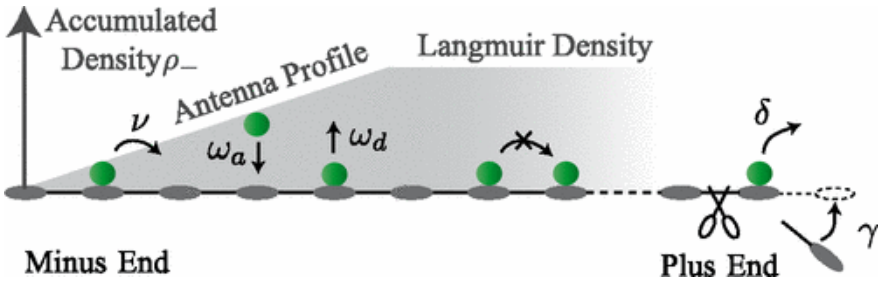


Figure 4.14: Illustration of the TASEP-LK model [1]. Proteins attach to and detach from the filaments

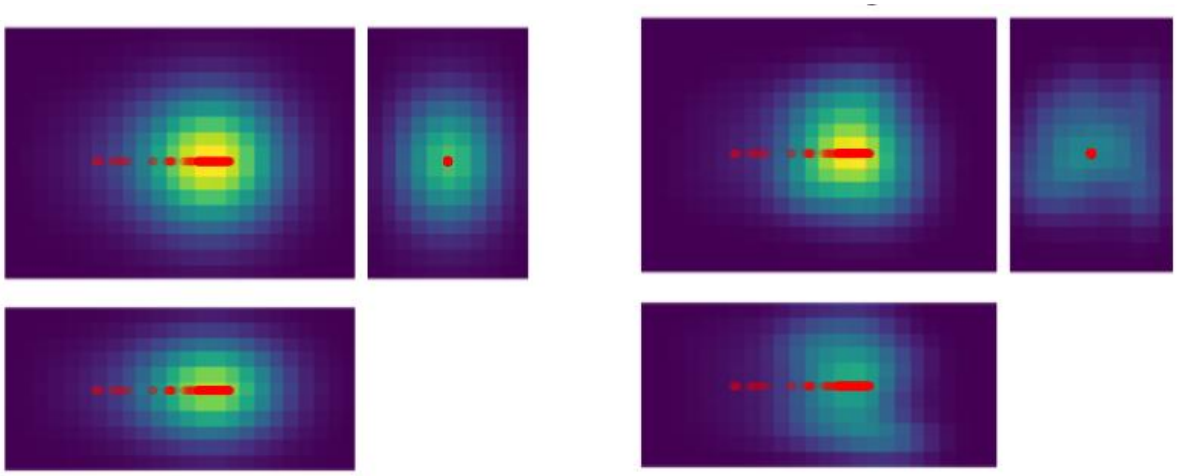


Figure 4.15: Image for TASEP-LK model with single (left) and multiple Gaussian (right)

Chapter 5

Conclusion

The target of the project to estimate PSF using sparse convex combination of Gaussian kernels is obtained by formulation as Bayesian inverse problem. The optimization problem solved using ADMM and **dictionary** of Gaussians gives satisfactory results with a bound on number of iterations. The value of cost function drops rapidly from first iteration itself. Therefore, running a limited number of iteration gives better results for ADMM in comparison to *CMA-ES* . Thus, use of ADMM helps us to obtain a good approximation efficiently. The algorithm gives an accurate estimation in case of synthetic data generated using Gaussians. The results obtained for non-gaussian PSF (Gibson and Lanni 3D optical model) are also satisfactory. The PSF obtained is able to follow the shape of complex true PSF with a limited number of Gaussians. The kernels in the **dictionary** can be ranked according to weights and can be selected as per our requirement.

There are few drawbacks in our algorithm that requires modification. The first one is effect of biasing due to shrinkage operator in $\boldsymbol{v}_2$-subproblem for high values of λ. This requires post-processing of results to remove this biasing effect. The other observation is about use of L-1 norm for obtaining sparse solutions in case of optimizing for ϕ and $\boldsymbol{w}$ separately. It was observed that for our optimization problem, there is no effect of increase of λ after a certain point. The use of L-1 norm cease to increase sparsity with increase of parameter, λ . The relaxation of cost term, $\mathbf{card}(\boldsymbol{w})$, to L-1 norm does not work with simplex constraint on $\boldsymbol{w}$. Therefore, we have to use a different definition for obtaining a sparse solution in simplex space [12].

Apart from problem formulation and algorithm used, the most crucial step for getting good results is choice of **dictionary** i.e means and variances of Gaussian kernels. The algorithm is not able to give good results, if the **dictionary** is badly chosen. The choice of means of **dictionary** by selecting brighter pixels works well, while, the variances of Gaussians are tested and tried to build a **dictionary** for giving satisfactory results. It can be observed that choice of variances is not as direct as choice of means. We propose to use Gaussian scale space for getting an estimate on value of variances to be used. The other way is to learn dictionary with results from optimization [13].

Overall, the project gives you insight in image formation process for microscope using fluoroscent beads. The project enables us to formulate a mathematical optimization problem using Bayesian inverse problem for a physical process of fluoroscent microscopy. It also introduces us to different convex optimization algorithms and their advantages.

Bibliography

[1] Anna Melbinger, Louis Reese, and Erwin Frey. Microtubule length regulation by molecular motors. *Phys. Rev. Lett.*, 108:258104, Jun 2012.

[2] Denis K Samuylov, Lukas A Widmer, Gabor Szekely, and Gregory Paul. Mapping complex spatio-temporal models to image space: The virtual microscope. *In Biomedical Imaging (ISBI), 2015 IEEE 12th International Symposium on. IEEE, pp. 707711.*, 2015.

[3] Changjiang Yang, Ramani Duraiswami, Nail A Gumerov, and Larry Davis. Improved fast gauss transform and efficient kernel density esti- mation. *In Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on. IEEE, pp. 664671.*, 2003.

[4] Michael Hirsch, Richard J. Wareham, Marisa L. Martin-Fernandez, Michael P. Hobson, and Daniel J. Rolfe. A stochastic model for electron multiplication charge coupled devices from theory to practice. *In PLoS ONE 8(1): e53671. doi:10.1371*, 2013.

[5] Richard W Cole, Tushare Jinadasa, and Claire M Brown. 3-D PSF fitting for fluorescence microscopy: Implementation and localization application. *Nat Protoc 6:19291941.*, 2011.

[6] F. Aguet, D. Van De Ville, and M. Unser. An accurate psf model with few parameters for axially shift-variant deconvolution. *In 5th IEEE International Symposium on Biomedical Imaging*, 2008.

[7] H. Kirshner, F. Aguet, D. Sage, and M. Unser. Measuring and interpreting point spread functions to determine confocal microscope resolution and ensure quality control. *Journal of Microscopy*, 2013.

[8] B. Zhang, J. Zerubia, and J.-C. Olivo-Marin. Gaussian approximationsoffluorescence microscope point-spread function models. *AppliedOptics, vol. 46, pp. 18191829*, 2007.

[9] R H Byrd, P Lu, and J. Nocedal. A limited memory algorithm for bound constrained optimization. *In SIAM Journal on Scientific and Statistical Computing 16 (5): 1190-1208*, 1995.

[10] N. Hansen and S. Kern. Evaluating the CMA evolution strategy on multimodal test functions. In X. Yao et al., editors, *Parallel Problem Solving from Nature PPSN VIII*, volume 3242 of *LNCS*, pages 282–291. Springer, 2004.

[11] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *In Foundations and Trends in Machine Learning 3.1, pp. 1122*, 2011.

[12] Julien Mairal, Francis Bach, and Jean Ponce. Sparse modeling for image and vision processing. *In Foundations and Trends in Machine Learning Computer Graphics and Vision Vol. 8, No. 2-3*, 2012.

[13] Trevor Hastie, Robert Tibshirani, and Martin Wainwright. *Statistical Learning with Sparsity The Lasso and Generalizations*. CRC Press, 2016.