Stray Light Compensation in Optical Systems

Aleksis Pirinen
Angelos Toytziaridis

Master's thesis
2015:E21

LUND UNIVERSITY

Faculty of Engineering
Centre for Mathematical Sciences
Mathematics

Lund University

Master Thesis

Stray Light Compensation in Optical Systems

Authors:
Aleksis Pirinen
Angelos Toytziaridis

Supervisor:
Prof. Cristian Sminchisescu

Co-supervisors:
Jonas Hjelmström
Per Wilhelmsson

Company:
Axis Communications
Mats Thulin

Examiner:
Prof. Kalle Åström

A thesis submitted in fulfilment of the requirements for the degree of Master of Engineering Mathematics

at the

Faculty of Engineering
Centre for Mathematical Sciences

June 2015

"Strength does not come from winning. Your struggles develop your strengths. When you go through hardships and decide not to surrender, that is strength."

Arnold Schwarzenegger

LUND UNIVERSITY

Abstract

Faculty of Engineering
Centre for Mathematical Sciences

Master of Engineering Mathematics

Stray Light Compensation in Optical Systems

by Aleksis Pirinen
Angelos Toytziaridis

All optical equipment suffers from a phenomenon called stray light, which is defined as unwanted light in an optical system. Images contaminated by stray light tend to have lower contrast and reduced detail, which motivates the need for reducing it in many applications. This master thesis considers computational stray light compensation in digital cameras. In particular, the purpose is to reduce stray light in surveillance cameras developed by Axis Communications. We follow in the spirit of other digital stray light compensation approaches, in which measurements are fit to a parametric shift-variant point spread function (PSF) describing the stray light characteristics of the optical system. The observed contaminated image is modelled as an underlying ideal image convolved with the PSF. Once the PSF has been determined, a deconvolution is performed to obtain a restored image. We provide comparisons of a few deconvolution strategies and their performances regarding the restoration of images. Also, we discuss different techniques for decreasing the computational cost of the compensation. An experiment in which the images are compared to a ground-truth is proposed to objectively measure performance. The results indicate that the restored images are closer to the ground-truth compared to the observed image, which implies that the stray light compensation is successful.

Keywords: stray light, point spread function, deconvolution, image processing

Acknowledgements

Realizing this master thesis would have been impossible without the generous support of several dedicated persons, including but not limited to the Core Tech Imaging group at Axis. We would now like to thank a few key persons. First, thank you Cristian for being our thesis supervisor. Your input on key questions has been really helpful. Second, we want to thank our co-supervisors Jonas and Per. You have both been eager to discuss any ideas regarding the thesis, and have provided many important insights. Thank you Björn for being patient with our questions; you have virtually been a third co-supervisor for this thesis! Many technical issues were resolved by Andreas - without you, we would probably still be stuck in the basement. Anders, thank you for taking your time to discuss what expectations are reasonable with respect to stray light compensation. Thank you, Mats, for believing in us and lettings us do our thesis at Axis; it has been really rewarding! Regards to professor Jansson for taking your time to answer questions regarding your work. Finally, we want to thank our close ones for all their love and support.

Contents

Symbols

$*$	Convolution (both shift-variant and shift-invariant)
$\mathcal{F}$	Fourier transform
E	Expectation
T	Matrix transpose
$\mathcal{O}$	Big ordo
$\lceil \cdot \rceil$	Ceiling operator

Chapter 1

Introduction

All optical equipment suffers from a phenomenon called *stray light*, which is defined as unwanted light in an optical system. Stray light is generally categorized as either *specular* or *scatter* [1]. Specular mechanisms, such as *ghost reflections*, are deterministic and follow physical models, e.g. Snell's law of reflections. Scatter on the other hand behaves more unpredictably and stems from surface roughness and contamination. Examples of both specular and scattering stray light is shown in Figure 1.1.

Being able to reduce stray light in images is crucial in many applications. For example, a surveillance camera may have difficulties detecting faces of people walking by if there is a bright street lamp nearby. This is because the light spreads over the image plane, resulting in a contrast reduction and reduced detail. In the worst case, parts of the image become completely corrupted. This is illustrated in Figure 1.2.

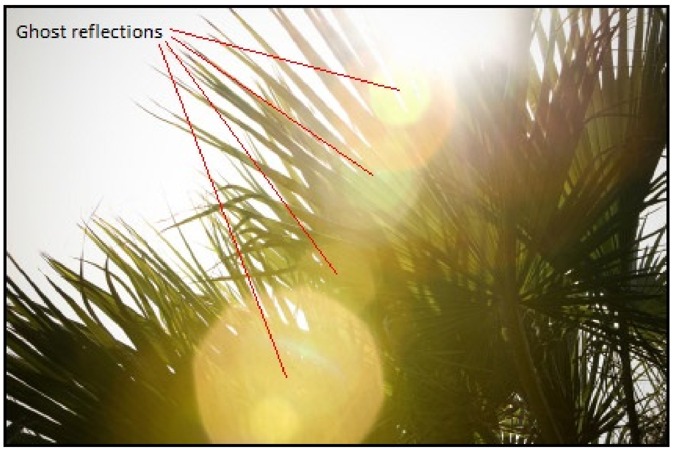
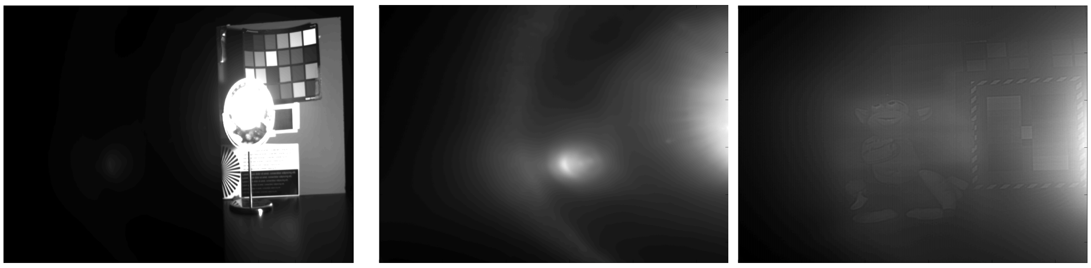


Figure 1.1: Stray light is unwanted light which contaminates the image. Some of the ghost reflections have been marked. Scattering can be seen as the blur around the ghost reflections. Image source: www.photography.ca/blog/tag/flare.



(a) Original image. Notice the mirror reflecting light. In Figure 1.2b the dark left area has been zoomed in.

(b) Nothing is visible in the dark area due to scatter and the ghost reflection.

(c) Same as in Figure 1.2b, but the reflecting mirror in Figure 1.2a was removed. Now a monkey and other details can be seen.

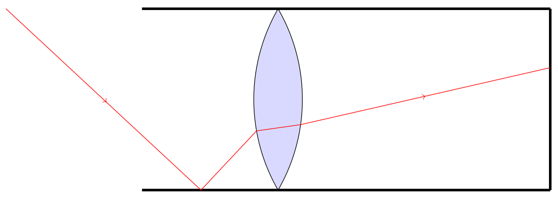
Figure 1.2: Illustration of how stray light may corrupt images.

Several approaches for reducing stray light have been proposed, either focusing on physical modifications of the system [1], or using computational techniques [2–7]. Physical modifications are generally more effective and refers to designing the optical system to reduce stray light, e.g., applying anti-reflexive coating on lenses; making surfaces of the system less reflective; or adding baffle vanes to stop light from reaching the image plane (see Figure 1.3). In [1, p. 9] it is argued that physical modifications can be summarized in the following sentence:

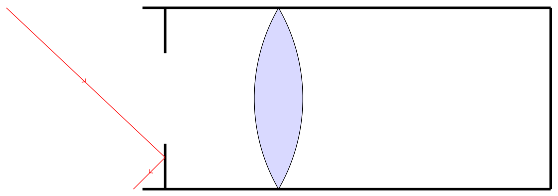
Move it or block it or paint/coat it or clean it.

It may be very difficult and/or expensive to design a system which removes stray light completely. Therefore, digital stray light compensation techniques are often used as a complement to physical modifications. Most digital approaches model the observed contaminated image as some underlying ideal image convolved with a filter characterizing the stray light. Assuming this model, the image restoration process can be posed as a filter inversion problem.

Another reason for utilizing digital techniques is that they allow cost-efficient simulation of what might be achieved by further physical improvements of the optical system, such as investing in better optics.



(a) Without baffle vanes, light from outside the field of view is more likely to hit the image plane.



(b) Using baffle vanes will stop some light from outside the field of view to hit the image plane.

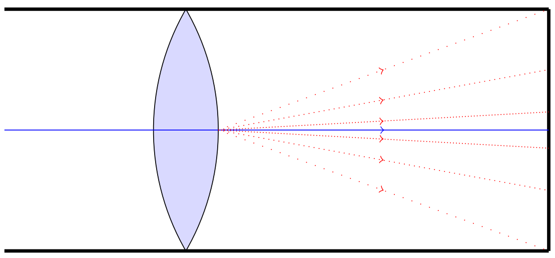
Figure 1.3: Example of a physical modification to reduce stray light.

1.1 Aim of this master thesis

This master thesis concerns computational techniques for stray light compensation. More specifically, the aim is to measure the stray light characteristics of a type of surveillance camera at Axis Communications (henceforth abbreviated Axis) and to perform digital stray light compensation in images of some scenes for said camera. To obtain the stray light characteristics of the camera, we measure its *point spread function* (PSF). The PSF describes the system's response to a point source and is also known as the impulse response. It is usually assumed to be *rotationally invariant* with respect to the optical axis [2–4]. This means that the shape of the PSF is only dependent on the distance from the center of the image, and not on the rotation.

For an ideal system, the point spread function is such that a point source is projected onto a single point in the image plane. Such an ideal system does not exist in practice which results in the point becoming smeared out over the image plane, i.e., the point gets surrounded by stray light. Note that both specular and scatter stray light is captured in the PSF. However, existing models (see for example [2] and [5]) often neglect the specular part of stray light, which is why the compensation might work poorly when the specular portion is relatively large.

Optical systems are mostly assumed to be linear and hence we can assume that the stray light in a natural image can be modelled as follows. Suppose there exists some underlying ideal image, even though we cannot directly observe it. Then the observed image containing stray light is given by the ideal image convolved with the PSF. Therefore, digital stray light compensation is in principle a matter of inverting the PSF, i.e., performing a deconvolution. Due to lens convexity and other space-varying characteristics,


(a) Light source located at the center of the image.

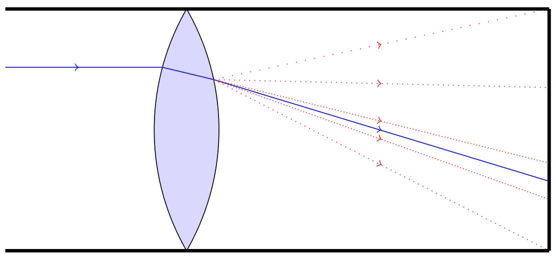

(b) Light source located closer to a corner of the image. Note how the expected scatter characteristics differ from Figure 1.4a.

Figure 1.4: Illustration of the shift-variant nature expected in a point spread function.

the point spread function is likely to be *shift-variant*, which means that the stray light properties vary over the image plane (see Figure 1.4). This makes it significantly more expensive to perform the deconvolution.

The above discussion gives rise to the following central questions for this thesis.

- Does existing equipment at Axis suffice to measure the point spread function?
- Do the measurements depend on the rotation of the camera around the optical axis, or is the camera rotationally invariant?
- Can the camera be modelled as a shift-invariant system, or must one account for shift-variance?
- Given a stray light compensation strategy, is it feasible in practice, i.e., in the hardware of the surveillance camera?

1.2 Overview

A schematic picture illustrating the chosen approach for compensating stray light is shown in Figure 1.5. This approach is suggested in [2, 3]. The purpose of Figure 1.5 is to give the reader an intuition for the work process and to offer guidance regarding where in this thesis the respective blocks are discussed in more detail.

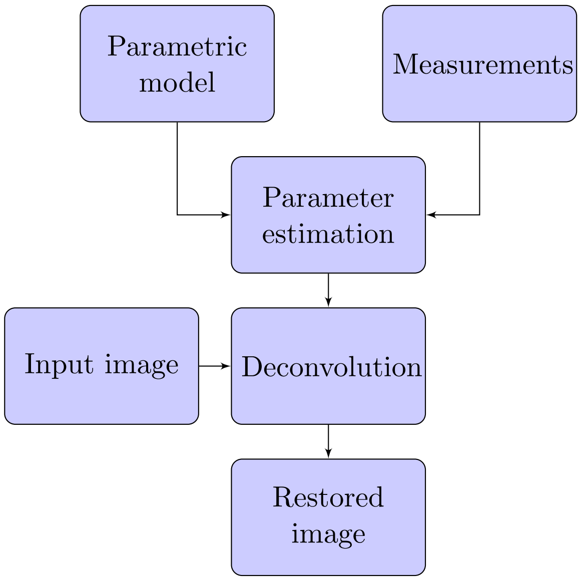


Figure 1.5: Schematic overview of the stray light compensation process. The *Parametric model* block is explained in Chapter 2 and refers to the stray light modelling process, in which a parametric model is chosen. The *Measurements* block is explained in Chapter 3 and concerns measurements of the stray light characteristics of the camera. In the *Parameter estimation* block (Chapter 4), the parametric model and the measurements are combined to estimate the model parameters. Given the full parametric model with estimated parameters, we use it to deconvolve an input image to obtain a restored image. The *Deconvolution* block is discussed in Chapter 5. Results of the stray light compensation are presented in Chapter 6. A discussion of the results and conclusions are given in Chapter 7.

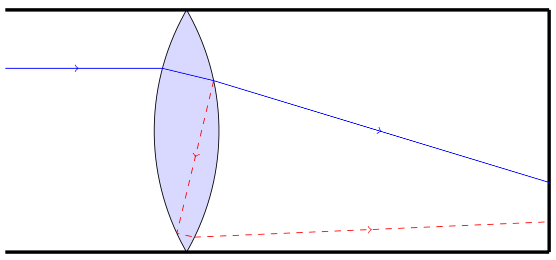
Chapter 2

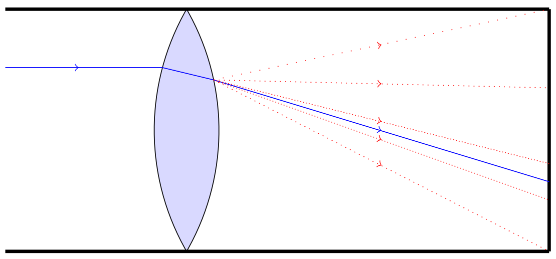
Stray light modelling

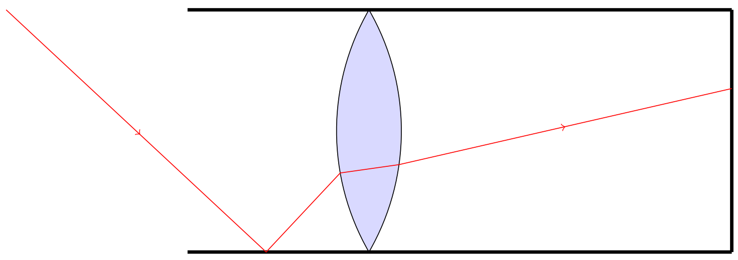
An image containing stray light is commonly explained with a *convolutive model*, in which the image is modelled as an underlying ideal image convolved with some filter [2–5]. Here, ideal refers to complete absence of stray light. Working with this thesis we have not found any alternative to a convolutive model, which is why we have chosen this approach to modelling stray light. In this chapter we try to intuitively illustrate why convolution is a useful tool for explaining stray light and introduce the model we use.

Several physical laws exist describing the behavior of light under ideal circumstances. For example, *Snell's law* explains the relationship between the incidence and refraction angles when light passes through a boundary between two different isotropic media. Unfortunately, any real optical system is far from ideal, as explained in Chapter 1. Recall that system imperfections such as surface roughness cause light to scatter in several directions in addition to following the path predicted by the physical laws. Moreover, some energy is lost to absorption in the reflecting surface.

In Figure 2.1 we conceptualize a few different stray light mechanisms. Figures 2.1a and 2.1b illustrate ghost reflections and scatter, respectively. Since stray light is defined as all unwanted light in an optical system, we also have to consider light entering the system from outside the *field of view* (FOV). This is illustrated in Figure 2.1c. Finally, in Figure 2.1d we illustrate all of the effects at the same time. Note that these figures are highly simplified for the sake of clarification. In reality, light is scattered in all directions and it does not only occur in the lenses but in all surfaces. Moreover, a camera contains several lenses and other optical elements, which further complicates the stray light characteristics. Hence, a single light source will have its energy spread over the entire image plane, causing a smearing effect.


(a) The incident ray is reflected inside the lens, resulting in a ghost reflection.


(b) The incident ray is scattered in all directions due to lens roughness. Not all scatter paths are shown, but only (a portion of) the paths hitting the image plane.


(c) Unwanted light entering the image plane through a baffle reflection.

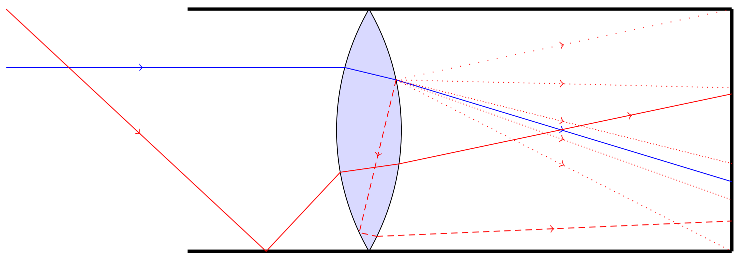

(d) Ghost reflection, scatter and baffle reflections at the same time.

Figure 2.1: Illustrations of some different stray light mechanisms. Note that for visualization purposes, only a few representatives of the light rays are shown in each figure.

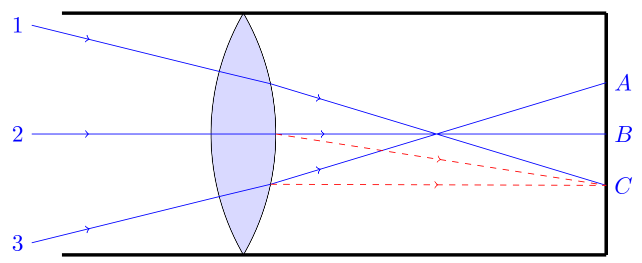


Figure 2.2: In an ideal optical system, the incident rays, would be projected according to the solid blue lines, i.e., we would have the one-to-one mappings $1 \leftrightarrow C$, $2 \leftrightarrow B$, and $3 \leftrightarrow A$. In reality, a point in the image plane will be contaminated by light which ideally would belong to other points. In this simplified illustration, the point C is contaminated by light which under ideal circumstances would belong to A and B.

In a natural scene, light enters the camera from all directions and each incoming ray will cause smear on the image plane. In the following however, we assume that light entering the camera stems only from within the field of view. Under this assumption, a given point in the image is contaminated by light which in an ideal image would belong to other points; see Figure 2.2. Hence, the intensity in a point is given by a linear combination of the intensities in all points in an underlying ideal image. That is, the intensities in the observed image are given by an ideal image convolved with some kernel.

We are now ready to formalize the above discussion. Let the function $o(x, y)$ describe a planar object emitting light, and let $i(x, y)$ be a function describing the magnitude of the light hitting the image plane. We assume that all light hitting the image plane comes from the object $o(x, y)$. Then the relationship between o and i can be modelled as

$$i(x, y) = \int\int s(x, y; x', y')o(x', y')\mathrm{d}x'\mathrm{d}y', \tag{2.1}$$

where the function s is the *point spread function* (PSF) of the system. The PSF describes the system's response to an infinitesimal point source. Note that in previous terms, o is the underlying ideal image and i is the observed image. In the case of a shift-invariant PSF, (2.1) reduces to an ordinary convolution.

Since images captured with a digital camera are discrete, we will mainly consider the discrete analog to (2.1),

$$i(x, y) = \sum_{x'}\sum_{y'} s(x, y; x', y')o(x', y'). \tag{2.2}$$

The only thing immediately available to us is the observed image $i(x, y)$. To find the underlying image $o(x, y)$, we also need the PSF $s(x, y, x', y')$. The PSF is also known as the system impulse response. In imaging systems the impulse response in a certain pixel is defined as an image of a point source located in that pixel. A point source is defined as a light source which is contained within a single pixel. For an ideal system, the PSF is such that a point source is projected onto a single pixel in the image plane. Such an ideal system does not exist in practice, which results in the point becoming smeared out over the image plane, as discussed above.

To completely determine the PSF we have to measure it in each pixel. For a camera with mega-pixel resolution this requires performing millions of measurements, which is infeasible in practice. Even worse, measuring the PSF in one single pixel is a non-trivial task, as we shall see in Chapter 3. Instead of measuring the PSF in each pixel, we can measure it in a few well-chosen pixels and perform a suitable interpolation to acquire an approximation of the full PSF. This is suggested in [8]. For example, one can use piecewise-constant or linear interpolation. The former would imply approximating the PSF with shift-invariant regions, which in turn would speed up computations, as will be discussed in Chapter 5.4.1. It should be added that both types of interpolation still require a lot of measurements to yield good PSF-approximations, since the PSF can hardly be approximated with such simple interpolations other than very locally.

If one cannot perform direct measurements, it may be possible to indirectly measure the PSF. In [2, 3] it is suggested to capture images of an object with known geometrical shape and size, so that one can calculate how many pixels the object would cover in an ideal image. With this knowledge, one can fit a suitable parametric model to the measurements, derived from the physical properties of the system. One drawback with such an approach is the difficulty of deriving an analytical parametric model, since the behavior of the PSF is highly nonlinear.

In this thesis we perform direct measurements of the point spread function and perform interpolation using the same parametric model as suggested in [2]. The reason for choosing this model is because it has been derived from physical properties of optical systems. It thus contains prior information, such as rotational invariance. Using this knowledge, we do not have to perform as many measurements as with the simpler interpolation techniques. For example, measurements along the diagonals should be sufficient to fit the model due to the assumed rotational invariance. The parametric model is discussed next.

2.1 Parametric model

In [2], the point spread function s is assumed to comprise of two components, s_I and s_D, according to

$$s(x_0, y_0, x_1, y_1; \beta, \sigma, a, b, d, f, g) = s_I(x_0, y_0, x_1, y_1; \beta, \sigma) + s_D(x_0, y_0, x_1, y_1; a, b, d, f, g), \tag{2.3}$$

where

$$s_I(x_0, y_0, x_1, y_1; \beta, \sigma) = \frac{\beta}{\sigma} \exp\left(-\frac{(x_1 - x_0)^2 + (y_1 - y_0)^2}{\sigma^2}\right) \tag{2.4}$$

and

$$s_D(x_0, y_0, x_1, y_1; a, b, d, f, g) = \frac{d}{\left(\frac{\frac{\left(x_1x_0+y_1y_0-x_0^2-y_0^2\right)^2}{\left[a\left(x_0^2+y_0^2\right)+g\right]^2}+\frac{(-x_1y_0+y_1x_0)^2}{\left[b\left(x_0^2+y_0^2\right)+g\right]^2}}{x_0^2+y_0^2}+1\right)^f}. \tag{2.5}$$

Here, x_0 and y_0 are the image plane coordinates of an infinitesimal area element emitting flux, and x_1 and y_1 are the image plane coordinates of an infinitesimal area element affected by that flux.

As seen in (2.4), s_I is a shift-invariant Gaussian kernel; it models imperfections due to phenomena other than lens scatter, such as *aperture diffraction*. Aperture diffraction (see Figures 2.3 - 2.4) is another contributor of stray light, and occurs at the entrance pupil of the camera. The parameters β and σ control the proportion and the spread of s_I, respectively.

The function s_D in (2.5) is the scatter component and is constructed such that it resembles the *bidirectional radiance distribution function* (BRDF). The BRDF is used in optics design to quantify surface scattering [1, p. 25]. It is rotationally invariant with respect to the optical axis and in [2] it is said to be well approximated by a 2D Gauss function, a 2D Cauchy function, or a combination of the two, such as a Voigt function. In addition to having these properties, s_D does not become numerically unstable at the origin. When evaluated in the center of the image plane, the contours of s_D will be circular and otherwise approximately elliptical (depending on a and b). Finally, it is smooth over the image plane. Its parameters can be interpreted as follows (citing [2]):

- a and b govern the rate of change of the respective elliptical cross section widths with respect to the distance from the optical axis center;
- d controls the proportion of s_D;
- f is a decay exponent;
- g governs the circular diameter of the PSF when it is centered at the optical axis.

In [3] it is suggested to normalize the PSF. After normalization, we obtain the final formula for the PSF,

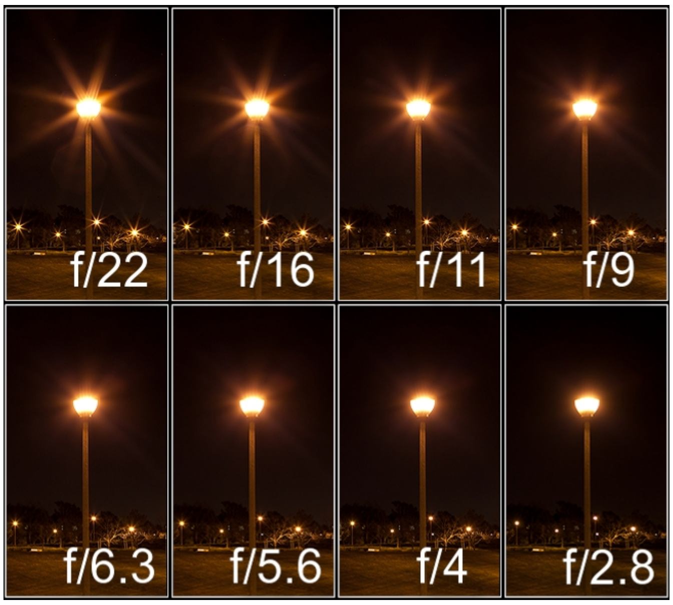


Figure 2.3: Aperture diffraction occurs at the entrance pupil of the camera, and is seen as the star-shaped light around the sources. The shape varies depending on what aperture is used. In (2.4), it is assumed that the aperture is circular, which results in a circular aperture diffraction. The extent of the aperture diffraction is connected to the focal length of the camera. Image source: `www.slrlounge.com/school/diffraction-aperture-and-starburst-effects`.

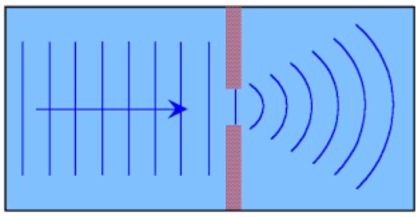


Figure 2.4: Simplified illustration of diffraction. A circular aperture results in spherical light-spread, which is projected as circles on the image plane. Image source: `http://www.gcsescience.com/pwav44.htm`.

$$s(x_0, y_0, x_1, y_1; \beta, \sigma, a, b, d, f, g)$$

$$= \frac{s_I(x_0, y_0, x_1, y_1; \beta, \sigma) + s_D(x_0, y_0, x_1, y_1; a, b, d, f, g)}{\sum\limits_{x_0}\sum\limits_{y_0}[s_I(x_0, y_0, x_1, y_1; \beta, \sigma) + s_D(x_0, y_0, x_1, y_1; a, b, d, f, g)]}. \quad (2.6)$$

The normalization makes sense, since the purpose of the PSF is to tell what *fraction* of each pixel in the underlying ideal image should be included to obtain a pixel in the observed image.

2.2 Model limitations

Since there exists no such thing as a perfect model, we will in this section highlight phenomena which are modelled insufficiently. These phenomena are ghost reflections and light entering the optics from outside the field of view.

As for ghost reflections, the problem does not lie in the convolutive model itself, but in the way it is parametrized. A measured PSF is unlikely to contain only a global maxima. Instead, we can expect it to contain several smaller local maxima as well, some of which correspond to ghost reflections. However, the parametric model in (2.6) does not exhibit such behavior and thus ghost reflections cannot be accounted for. It might be possible to derive an extended parametric model which supports ghost reflections, but it has not been considered in this thesis.

The second problem regards light entering from outside the FOV and is more severe, since it is not even supported by the convolutive model. To understand this, we note that light from outside the FOV should ideally not be in the image at all, and hence there exists no well-defined PSF for light sources outside the field of view.

Chapter 3

Measuring the point spread function

In this chapter we discuss the non-trivial matter of measuring the PSF. Before explaining our measurements, we provide a brief background on image acquisition and noise sources.

3.1 Image acquisition

Figure 3.1 shows a schematic overview of how a scene is captured and transformed into a digital image. For an image sensor to be able to interpret colors, incoming light is separated based on wavelength. Most commonly, the *Bayer filter* is used for color separation into different color channels. The Bayer filter uses a color filter array (CFA) that passes red (R), green (G) and blue (B) light to the sensor. As can be seen in Figure 3.1, the CFA consists of 50% green entries and only 25% red and blue entries, respectively. According to US patent [9], this filter design is intended to mimic the physiology of the human eye. When intensities of the respective colors are registered, an RGB-image can be obtained using e.g. linear interpolation on each color channel. This process is known as *demosaicing*. Note that light refraction is dependent on wavelength, as illustrated in Figure 3.2. Therefore, we can expect that the PSF will differ depending on if we only use red, green or blue light. Hence, stray light compensation has to be done for each channel separately.

To get a feel for how an image sensor works, a very simplified description is as follows. When capturing a photo, light is collected by the pixels located on the sensor. After a certain amount of time has passed, the so called exposure time, the sensor registers how much light has been gathered in each pixel. Analogously, we can think of a pixel as a

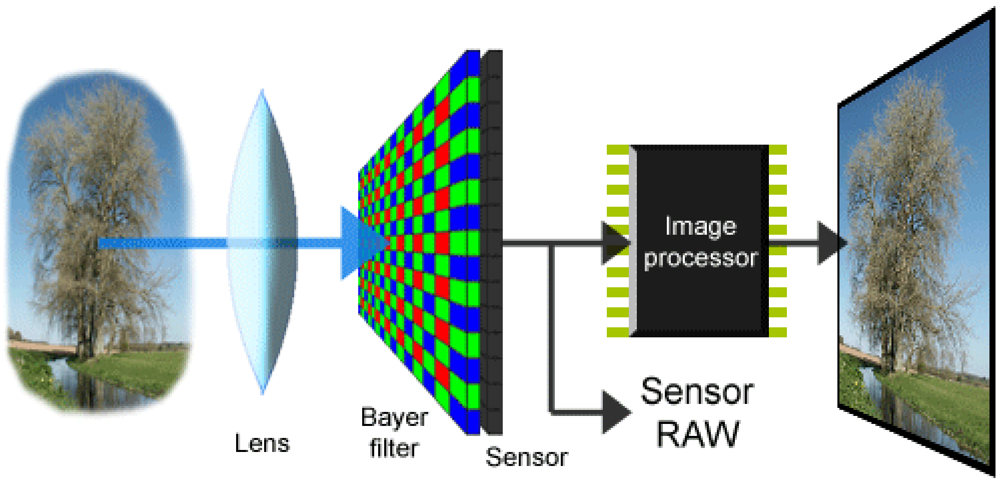


Figure 3.1: Schematic overview of the image acquisition process. Light describing the natural scene first enters through the lens. This light is separated into specific colors, most commonly using the Bayer filter, which separates light into red, green and blue. The image sensor registers the intensities of the different colors, which yields raw image data. This data can be forwarded to an image processor, which constructs an RGB-image, for example by interpolating the values within the color channels. Note that the image processor can be designed to do a lot more, such as image sharpening or stray light compensation. Image source: `www.trustedreviews.com/opinions/digital-photography-tutorial-raw-usage`.

glass being filled with water from a tap; the exposure time is the time we let the water run. A pixel is said to be saturated when it has collected the maximum amount of light its physical properties allow. In the water tap analogy, this means that the glass runs over with water. Saturated pixels provide only a limited amount of information, since they give no knowledge about how much they have exceeded their physical properties.

Most of today's cameras capture digital images. This means that light intensity must be measured in discrete quantities known as intensity levels, or bits. Since sensors are yet to handle saturation, the range of describable intensities is limited. The number of available intensity levels depends on the quality of the sensor in the camera. The sensor in the camera used in this thesis utilizes 12 bits to describe the intensity in each pixel, which means that it has 4096 intensity levels in total.

3.1.1 Image noise

All data acquired via measurements will contain some amount of noise and images are no exception. In this section we briefly consider a few different types of image noise. We refer to [10] for more details on image noise.

As explained above, an image sensor captures a scene by counting the number of photons hitting the sensor during a specific time interval. Due to the inherent randomness of light

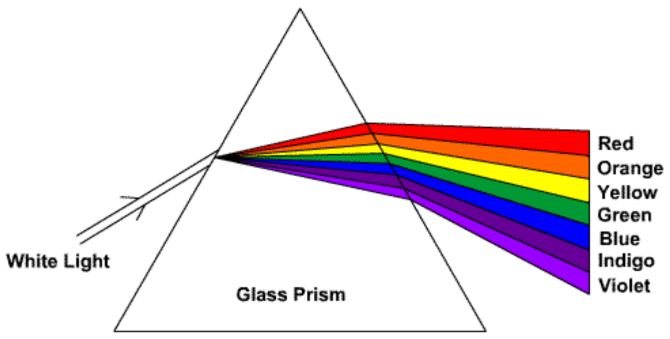
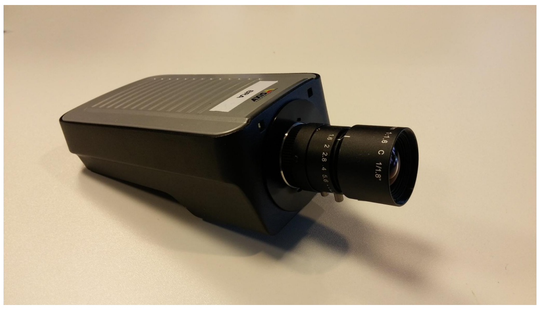


Figure 3.2: Illustration of how light refraction depends on the wavelength. Image source: physics.stackexchange.com/questions/2169/why-does-light-refract-if-photons-are-not-bound-by-an-axle.

quantization, there will be random fluctuations in the amount of intensity registered by the sensor. This variation in measured intensity is known as *photon shot noise*. The Poisson distribution is commonly used to model shot noise, which approximates a Gaussian distribution, except at very low intensities. Therefore, an additive Gaussian model is also often used for photon shot noise. Analog-to-digital conversion and bit-errors in transmission cause *read noise*. It is usually independent from the underlying image. An additive Gaussian model is typically consulted for describing read noise. Another form of noise can appear due to electron leakage from the image sensor. This effect, which is occasionally known as *dark current*, increases with increased exposure time. The effect is in the most simple case described as a constant offset on all pixel values.

To effectively deal with noise, one needs to be able to estimate the noise properties of the optical system. This requires some caution when designing the system. Recall that a camera captures digital images, which in our case means that we have at most 4096 intensity levels available. Even though light intensity is strictly positive, the noise can cause the measured signal to become negative. If the camera is not able to capture such fluctuations, information about the noise will be lost, since its negative part is cut off. Therefore, camera manufacturers often put the zero intensity reference level somewhat higher than 0 intensity levels. The camera we use has its reference level set to approximately 240 intensity levels, meaning that we have $4096 - 240 = 3856$ levels available to describe a scene. A common way to classify noisy pixels is to define an intensity threshold, commonly called the noise floor, under which the *signal-to-noise ratio* (SNR) is deemed to be to low. For our camera, this threshold has been estimated to be 20 intensity levels.


(a) The Q1615 camera provided by Axis.

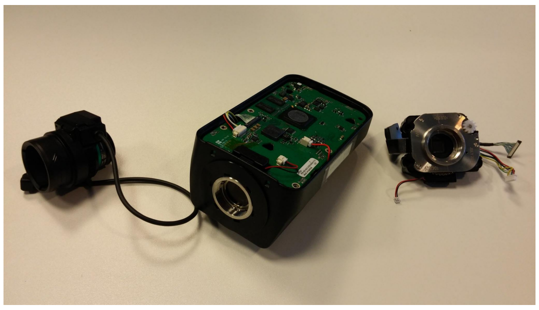

(b) Different parts of the camera. To the left is the optics; in the middle is the chip containing the image processor; and to the right is the sensor.

Figure 3.3: The camera used in this thesis.

3.2 Measurements

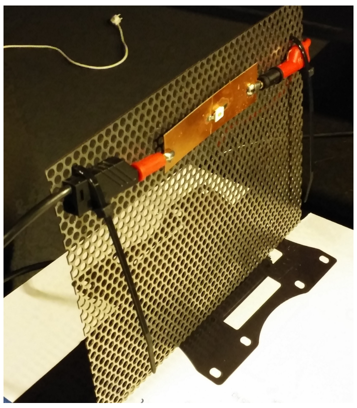
The camera we use in this thesis is shown in Figure 3.3 and is provided by Axis, as is all other equipment. Measuring the PSF is a nontrivial problem and gives rise to the following questions.

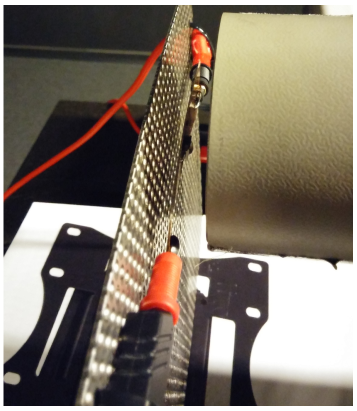
1. When measuring a point source, we cannot allow light to be reflected on the walls and into the system, since such reflections will be interpreted as additional light sources, which corrupts the measurement. How can such reflections be avoided?
2. Where in the image plane should we measure the PSF in order to capture as much of the shift-variance as possible?
3. How can we get a precise measurement of a point source? That is, how do we deal with corrupted (i.e., noisy and saturated) pixels?

In the next section we provide potential answers to the first and second questions; the third question is answered in the succeeding section.

3.2.1 Experimental setup

To create a point source, we use a small LED-diode (OSRAM, `LCW W5AM`). At Axis, we have access to a black-painted 30 meter long room, which allows us to place the LED far enough from the camera to make it smaller than the size of a pixel in the image. This yields our point source. The LED is mounted to a copper plate which helps dissipate the heat produced. This plate is in turn mounted to a simple construction; see Figure 3.4a.


(a) A small LED-diode is mounted to a cooling copper plate. The LED is used at 0.654 mA and 3.22 V, and the copper plate is used to prevent it from overheating.


(b) The diode is placed at the end of a 1.5 meter tube with a diameter of about 7 cm. The inside of the tube is coated to reduce light reflections.

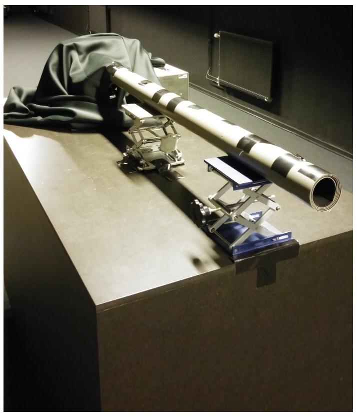

(c) The entire light cannon setup. The camera in Figure 3.3 captures images of the light source from approximately a 25 meter distance.

Figure 3.4: The measurement setup.

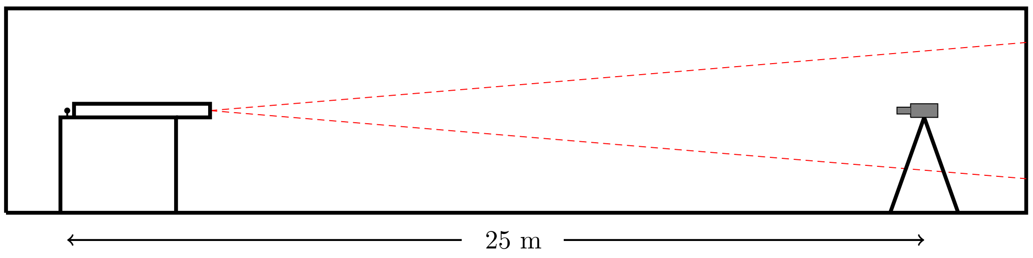


Figure 3.5: Two-dimensional illustration of the measurement setup, not drawn to scale. To the left is the light cannon and to the right is the camera. The dashed lines from the light cannon illustrate that light from the cannon does not reflect on any external surfaces before reaching the image plane of the camera.

In order to stop reflections bouncing off the walls, we direct the light from the LED by constructing a light cannon. We place the LED at the end of a long tube which has a black-coated interior, in hope of reducing reflections inside the tube. To stop light from propagating backwards, we cover the LED with a black piece of cloth. The resulting light cannon is shown in Figures 3.4b and 3.4c. The overall measurement setup is shown in Figure 3.5.

As can be seen in Figure 3.5, the resulting light cone hits the wall directly behind the camera. This can be used to determine if the light cone is small enough to avoid hitting the walls before the image plane. If the light behind the camera forms a coherent circle, no reflections have occurred, or at least only negligible reflections. Note however, that upon hitting the wall the, light will bounce backwards and eventually enter the camera one way or another. We choose to ignore this, since the effect is likely negligibly small.

To capture as much of the possible shift-variance of the PSF as possible, we measure it at

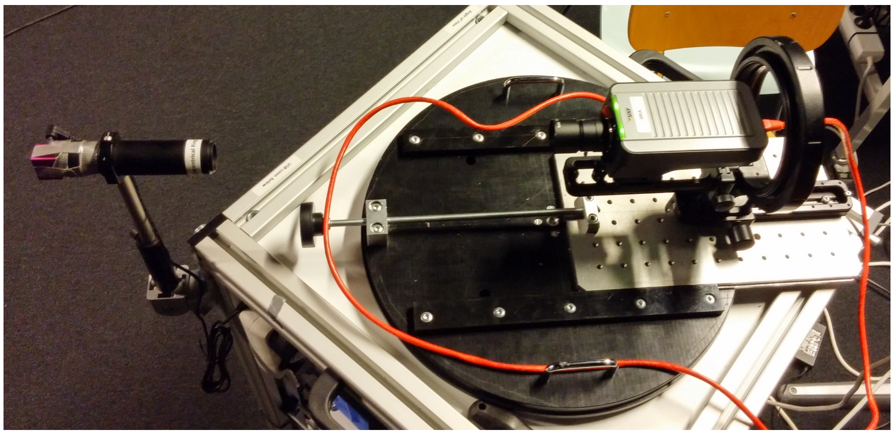


Figure 3.6: The camera is mounted to this construction, which can rotate the camera around its optical axis as well as relative to the floor. This enables us to capture images of the light source at different positions in the image plane. The tube-looking item to the left is a collimator, which is used to obtain the field of view of the camera.

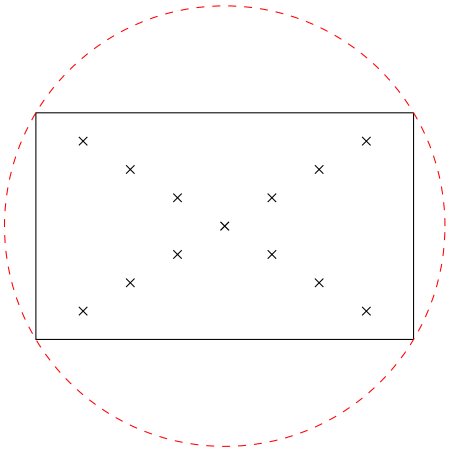


Figure 3.7: Illustration of all the 13 diagonal PSF-measurements. The image plane is represented by the rectangle and the red circle represents the lens. The diagonal FOV is 67.32 degrees. Due to the assumed rotational symmetry with respect to the optical axis, it should suffice to measure along the diagonals, which allows us to cover as much of the stray light characteristics as possible. In fact, if rotational invariance truly holds, then we would only need to measure along one of the diagonals. However, more measurements tend to yield better parameter estimates. Also, the measurements on the other diagonal may be useful to verify rotational invariance. This is done in Chapter 6; see Figure 6.1.

several different locations in the image plane using the setup shown in Figure 3.6. This setup lets the camera be rotated relative to the floor and around its optical axis, which allows us to measure the PSF along the two diagonals of the image plane. We choose to only measure along the diagonals, hoping that this gives enough information about the PSF due to the assumed rotational invariance of the system. We do 7 measurements along each diagonal, at 8.42 degrees apart. The measurements are illustrated in Figure 3.7 and shown in Figure 3.8.

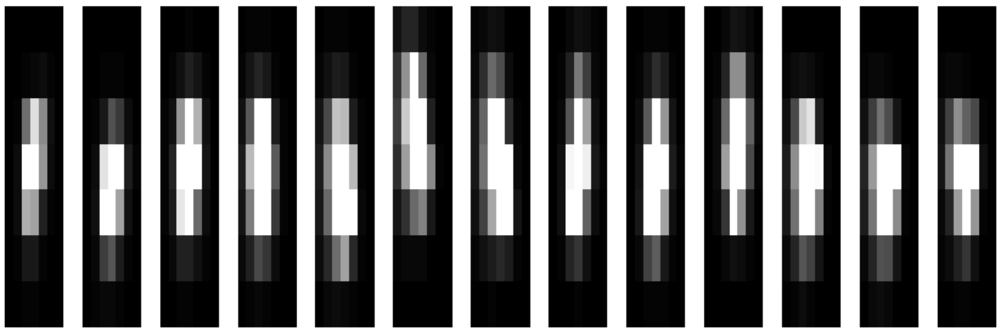


Figure 3.8: The 13 PSF-measurements, normalized to the range $[0, 1]$. These are used to estimate the model parameters (see Chapter 4). Note that we show only a 7×7 pixel area around the center of the PSF-measurements. The measurements are ordered as follows. From the left, the first seven measurements correspond to the main-diagonal from top to bottom in Figure 3.7. The remaining six measurements correspond to the off-diagonal (not including the center point once more) from bottom to top. Color axis: $[0, 0.002]$.

3.2.2 Acquiring precise measurements

Using only one exposure time, it is not possible to capture an image of a light source in which each pixel becomes meaningful. That is, some pixels will become saturated or lie below the noise floor. Therefore, to get a precise approximation of the PSF, we have to capture images of the light source at several different exposure times.

From our discussion on how sensors work, we can find the following relation between intensities at different exposure times. If a constant flux of light is assumed while capturing an image at a certain exposure time, then it holds true that

$$I(t, x, y) = I(t_0, x, y)\frac{t}{t_0}. \tag{3.1}$$

Here, $I(t, x, y)$ is the intensity in a pixel (x, y) at exposure time t, and t_0 is a reference exposure time. Note that the assumption holds true for the experimental setup described in the previous section, except for minor fluctuations caused by noise. To see why (3.1) makes sense, we return to the water tap analogy. Assume that we fill the glass with a constant stream of water for a specific amount of time t_0. Now, suppose we fill another glass with the same constant stream of water, but for some other time t. Then it is reasonable that the glass contains t/t_0 of the amount of water in the original glass (unless the water ran over in any of the glasses).

To ensure maximum precision when approximating the PSF in one pixel, one should ideally choose the referential exposure time t_0 such that the intensity in the pixel containing the point source is just below the saturation threshold. However, we do not spend time on such a calibration and instead use exposure time settings provided by our co-supervisors at Axis, which uses a referential exposure time reasonably close to the

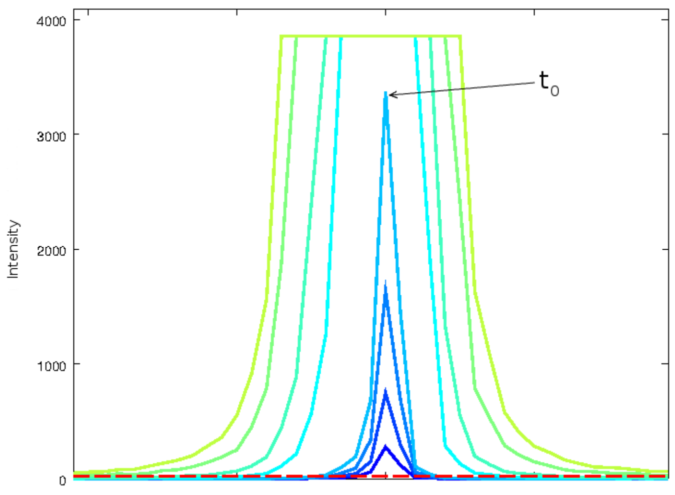


Figure 3.9: Slices from raw data of measurements using different exposure times. The dashed red line is the noise floor, located at 20 intensity levels. As can be seen, some exposure times yield saturated data close to the light source, but more information in the tails. Note that the zero level reference offset of 240 intensity levels has been subtracted.

optimum. Thereafter, we take measurements using a range of exposure times such that we obtain meaningful pixels in different regions of the image; see Figure 3.9. Finally, we enhance the PSF in the pixel captured with the referential exposure time by using a weighted average of all the measurements, where we for each exposure time throw away meaningless pixels. The weights are chosen according to (3.1).

Let $I^*(t_0, x, y)$ be the enhanced version of $I(t_0, x, y)$, N the number of different exposure times and W a rectangular window such that $W\{I\} = I$ for $20 < I < 3856$ and 0 else. Then the enhanced image is formally given by

$$I^*(t_0, x, y) = \frac{1}{N}\sum_{i=1}^{N}\frac{t_0}{t_i}W\{I(t_i, x, y)\}. \tag{3.2}$$

Furthermore, to reduce read and shot noise, we choose to perform several measurements at each exposure time and take the average. Let M be the number of measurements at each exposure time and $I_j(t_i, x, y)$ the jth measurement at the ith exposure time. Then we can refine (3.2) as

$$I^*(t_0, x, y) = \frac{1}{N}\sum_{i=1}^{N}\frac{t_0}{t_i}\left(\frac{1}{M}\sum_{j=1}^{M}W\{I_j(t_i, x, y)\}\right). \tag{3.3}$$

Finally, note that (3.3) can be used on any image and not just on PSF-measurements.

Chapter 4

Parameter estimation

So far, we have introduced the parametric model for the point spread function (Chapter 2) as well as the PSF-measurements (Chapter 3). In this chapter we explain how to estimate the model parameters in (2.6) given the measurements of the point spread function. In particular, we introduce the error criteria to be minimized in order to obtain the parameters, and describe the optimization method we use. The content of this chapter mainly follows the work done in [2] and [3].

4.1 Error criteria

Suppose we have access to an underlying ideal image $f_i(x, y)$ corresponding to the observed image $f(x, y)$. Then, according to (2.2), an estimate $\hat{f}(x, y)$ of $f(x, y)$ is given by

$$\hat{f}(x, y) = \sum_{x_0} \sum_{y_0} f_i(x_0, y_0) s(x_0, y_0, x, y; \beta, \sigma, a, b, d, f, g), \tag{4.1}$$

where s is given in (2.6). In [2], the authors suggest using the following error,

$$\epsilon = \sum_{k=1}^{N} \sum_{(x,y)\in I_k} \left[w(x, y) \left(\hat{f}_k(x, y) - f_k(x, y) \right) \right]^2 . \tag{4.2}$$

Here, N is the number of PSF-measurements, $w(x, y)$ is a weighting factor for the pixel (x, y), I_k is the set of selected pixels for the kth image, $\hat{f}_k$ is the kth estimated image, and f_k is the kth PSF-measurement. The sets I_k are introduced to improve the computational efficiency and are usually set to relatively small rectangular cut-outs containing the PSF in the center. It is of course possible to let I_k represent the entire images, so that as much information as possible is used in the parameter estimation.

4.2 Optimization method

The parameters are found by minimizing (4.2) in Octave using the package `minFunc` developed by Mark Schmidt at University of British Columbia. This package is similar to the built-in unconstrained optimization toolbox `fminunc`, but it contains some additional useful optimization methods. The optimization is initialized the same way as in [2], using $\beta = \sigma = 1$, $a = b = 0$, $d = g = 1$ and $f = 2$, making the PSF shift-invariant.

We use the default optimization method in `minFunc`, which is limited-memory BFGS (L-BFGS). Standard BFGS is a quasi-Newton method which is described in detail in [11, Ch. 8.1]. Quasi-Newton methods are similar to the Newton method [12, Ch. 9.5], but replace the computations of Hessians with approximations of Hessians, which are computed more efficiently. BFGS is a quasi-Newton method which uses approximations of the inverse Hessians instead, to avoid solving expensive linear systems of equations. While the original BFGS stores a dense approximation of the inverse Hessian, L-BFGS stores only a few vectors which describe the approximation sufficiently well [11, Ch. 9.1]. For this reason, L-BFGS is particularly useful for large problems.

4.3 Improving optimization efficiency

Directly computing the cost function (4.2) is expensive due to the shift-variance of the PSF. However, the computation can be done much more efficiently using the sparsity of the ideal images. First, suppose that no normalization is performed in (2.6). Let N_k denote the number of non-zero pixels in the kth ideal image, and let M_k denote the number of pixels contributing to the cost (i.e., M_k is the number of elements in I_k). Then the cost for the kth image becomes $\mathcal{O}(M_k N_k)$. Recall from our discussion in Chapter 3 that the ideal images consist of one non-zero pixel each, which is why the complexity per image becomes only $\mathcal{O}(M_k)$.

Even with a sparse representation, the actual complexity is bounded from below by the normalization in (2.6), as it requires M_k summations over the entire image. To deal with this bottleneck, we take a similar approach as proposed in [3], where the PSF is assumed to be locally shift-invariant inside the target region I_k. The shift-invariant PSF inside the target region is obtained by evaluating the shift-variant PSF in the center of the region. The authors then compute (4.1) more efficiently using fast Fourier transform (FFT); see [13, Ch. 18]. In our case, we speed up the normalization process by computing the normalizing factor only at the center of the target region. See Chapter 5.4.1 for more details on approximations using local shift-invariance. In Chapter 6.1, we compare the full and fast optimization methods.

Chapter 5

Deconvolution

In the context of digital stray light compensation, deconvolution concerns finding the ideal image $o(x, y)$ in (2.2) given the corrupted image $i(x, y)$ and the shift-variant point spread function $s(x, y; x', y')$. We begin this chapter by providing a brief overview of different deconvolution methods. First, we discuss fast, non-iterative approaches for deconvolution (Sections 5.1 - 5.2), and then we move on to computationally heavier but more robust and adaptive iterative approaches (Section 5.3). Chapters 5.1 - 5.3 summarize parts of [14, Ch. 4 - 5], in which a much more comprehensive treatment of the subject is given. To speed up the computations for the iterative methods, suitable approximations must be considered. A few approximation strategies are discussed in Section 5.4.

5.1 Direct filter inversion

It is well-known that a shift-invariant convolution in the space domain corresponds to a multiplication in the Fourier domain. That is,

$$i(x, y) = s(x, y) * o(x, y) \overset{\mathcal{F}}{\longleftrightarrow} I(f_x, f_y) = S(f_x, f_y)O(f_x, f_y). \tag{5.1}$$

For notational convenience, we henceforth omit the coordinates, as they are implicitly understood. The desired image is obtained by solving for O and applying the inverse Fourier transform,

$$o = \mathcal{F}^{-1}\left\{\frac{I}{S}\right\}. \tag{5.2}$$

This can be extended to the shift-variant case by using suitable approximations; see Section 5.4.1.

The straightforward filter inversion technique works well in a theoretical scenario in which the image is completely absent of noise, but tends to work poorly in practice. To explain this, we introduce noise into (5.1),

$$i = s * o + n \overset{\mathcal{F}}{\longleftrightarrow} I = SO + N. \tag{5.3}$$

Solving for O yields an estimate $\hat{O}$,

$$\hat{O} = \frac{I - N}{S} = O - \frac{N}{S}, \tag{5.4}$$

where O denotes the desired ideal image in (5.1). In most applications, S tends to be small for large spatial frequencies, which causes the noise term N/S in (5.4) to explode, rendering the direct filter inversion approach unsuitable on real data. Next, we look at one way to handle the noise-sensitivity.

5.1.1 Wiener inverse filter

The main drawback of direct filter inversion is its sensitivity to noise. It would be convenient if there existed an approximate inverse filter h taking the noise into account. That is, we want a filter h yielding an, in some sense, optimal estimated image $\hat{o}$ in (5.3) via $\hat{o} = h * i$. It turns out that if the noise is Gaussian and additive with known mean and variance, then one can construct a filter which is the linear minimizer of the *mean squared error* (MSE) between the desired image o and the estimated image $\hat{o}$. The filter is best explained in the frequency domain. Let H denote the Fourier transform of h. Then the optimal filter H^* is given by the argument minimizing the MSE,

$$H^* = \arg\min_H \mathrm{E}\left\{\left|O - \hat{O}\right|^2\right\} = \arg\min_H \mathrm{E}\left\{|O - H(SO + N)|^2\right\}. \tag{5.5}$$

Solving the minimization problem (5.5), one arrives at the Wiener inverse filter,

$$H^* = \frac{1}{S}\left(\frac{|S|^2}{|S|^2 + \frac{N}{O}}\right). \tag{5.6}$$

The effect of the this filter becomes intuitive when realizing that the term N/O in the denominator of (5.6) is the inverse signal-to-noise ratio. As the SNR approaches infinity, (5.6) approaches the direct inverse given in (5.2). When the noise increases the SNR decreases, which implies an attenuation in the Wiener filter. Therefore, H^* does indeed take the noise characteristics into account, as desired.

Note that filter inversion techniques do not guarantee that the restored image becomes physically sound, as opposed to some of the iterative methods described in Section 5.3.

For example, some entries may become negative. When noise is present, a ringing effect often appears in the restored image. This can occur even when using the Wiener filter, although the effect is usually considerably smaller if the noise characteristics are well estimated.

5.2 Convolution as a matrix-vector product

In this section we look at a matrix representation of (2.2), which gives a slightly different perspective on what was discussed in Section 5.1. Moreover, the matrix representation is convenient when explaining the iterative methods in Section 5.3.

Using vector notation, (2.2) may be written shorter as

$$i(x,y) = s\left(x,y;\boldsymbol{x}',\boldsymbol{y}'\right)^T o\left(\boldsymbol{x}',\boldsymbol{y}'\right), \tag{5.7}$$

where $\boldsymbol{x}'$ and $\boldsymbol{y}'$ are vectors containing each x' and y' in the summation (2.2), respectively. Note that in (5.7), s and o are computed element-wise. Since this is done for each point (x,y) in the image plane, it is convenient to consider the column vectors $\boldsymbol{i}$ and $\boldsymbol{o}$ corresponding to $i(x,y)$ and $o(x,y)$, respectively. Then (5.7) can be written in matrix form as

$$\boldsymbol{i} = \boldsymbol{S}\boldsymbol{o}, \tag{5.8}$$

where $\boldsymbol{S}$ is a square *convolution matrix* with rows $s\left(x,y;\boldsymbol{x}',\boldsymbol{y}'\right)^T$. Hence, deconvolution can be stated as a matrix inversion problem. If the image plane is represented by $m \times n$ pixels, $\boldsymbol{S}$ will be of size $mn \times mn$, making direct inversion intractable for mega pixel-resolution images. If s is shift-invariant, $\boldsymbol{S}$ becomes block Toeplitz with Toeplitz blocks, allowing fast computation using FFT [13, Ch. 18]. This was discussed in Section 5.1, albeit in a different framework. Since s is generally shift-variant, $\boldsymbol{S}$ will be dense and lack structure. In particular, speeding up the computations using FFT is not applicable in the shift-variant case. In [4], a method for computing the dense matrix inverse more efficiently is proposed, which is based on wavelet-based lossy source coding of the shift-variant convolution matrix $\boldsymbol{S}$.

Another major issue with (5.8) is that a given row in $\boldsymbol{S}$ is likely similar to nearby rows, which results in an ill-conditioned system of equations. This is unavoidable since it naturally follows that neighboring pixels in an image are similarly affected by stray light. Therefore, even a small amount of noise may corrupt the image estimate significantly.

In the next section, iterative methods for computing the inverse are described. Some of these methods have the advantage of allowing control over spurious fluctuations by introducing constraints on the restored image, which tends to make these iterative methods less sensitive to noise and more likely to produce physically sound results.

5.3 Iterative methods

This section considers iterative methods for inverting the linear system (5.8). We begin by deriving a widely used linear method developed by Van Cittert. This method is intuitively understood, but sometimes suffers from the same drawbacks as filter inversion. For example, it occasionally produces results with no physical meaning, such as images with some negative entries. After considering Van Cittert's method, we show some existing non-linear alternatives, which are able to correct some of the flaws associated with linear methods [14, Ch. 4].

5.3.1 Van Cittert's method

It is well-known that systems of linear equations in which the diagonal elements are the largest of their respective rows can be solved iteratively. Assuming $\boldsymbol{S}$ is such a matrix, consider the computation of the nth pixel,

$$i_n = \sum_m S_{nm} o_m. \tag{5.9}$$

Moving the diagonal element outside the sum, we get

$$i_n = S_{nn} o_n + \sum_{m \neq n} S_{nm} o_m. \tag{5.10}$$

Solving for o_n yields

$$o_n = \frac{1}{S_{nn}} \left(i_n - \sum_{m \neq n} S_{nm} o_m \right). \tag{5.11}$$

This means that we can determine o_n if we know all its neighbors. Hence, given that we have somehow obtained estimates $\hat{o}_m^{(k)}$ of all its neighbors, we can get a refined estimate of o_n as

$$\hat{o}_n^{(k+1)} = \frac{1}{S_{nn}} \left(i_n - \sum_{m \neq n} S_{nm} \hat{o}_m^{(k)} \right). \tag{5.12}$$

Repeating this process for all pixels, we will obtain a full set of $\hat{o}_n^{(k+1)}$. Finally, since we have old estimates of o_n, we can rewrite (5.12) as

$$\hat{o}_n^{(k+1)} = \hat{o}_n^{(k)} + \frac{1}{S_{nn}} \left(i_n - \sum_m S_{nm} \hat{o}_m^{(k)} \right). \tag{5.13}$$

Repeated use of (5.13) is known as the point-simulation relaxation method, the point Jacobi method, or the method of simultaneous displacements. Note that there is no guarantee that the correction term is adequately scaled, which may result in slow or no convergence at all. It is therefore suitable to introduce a scaling factor κ, yielding

$$\hat{o}_n^{(k+1)} = \hat{o}_n^{(k)} + \frac{\kappa}{S_{nn}} \left(i_n - \sum_m S_{nm} \hat{o}_m^{(k)} \right). \tag{5.14}$$

It is convenient to note that (5.14) can be written using the convolution notation,

$$\hat{o}^{(k+1)} = \hat{o}^{(k)} + C \left(i - s * \hat{o}^{(k)} \right), \tag{5.15}$$

where $C = \frac{\kappa}{S_{nn}}$ constitutes the scaling factor. The iteration is commonly initialized using $\hat{o}^{(0)} = i$. Setting $C = 1$ in (5.15) finally gives us Van Cittert's method,

$$\hat{o}^{(k+1)} = \hat{o}^{(k)} + \left(i - s * \hat{o}^{(k)} \right). \tag{5.16}$$

In [14, p. 96 - 97] it is shown that when Van Cittert's method converges, it converges to the estimate obtained using the inverse filter, and that convergence occurs precisely if $|1 - S| < 1$, where S is the Fourier transform of s.

Looking at (5.16), we see that iterates may become negative due to the subtraction. Another drawback is that present additive noise will grow with each iteration, which is explained in Appendix A. In [14, p. 131] it is proposed that smoothing i and the first few iterates $\hat{o}^{(k)}$ may be useful in order to reduce the unwanted noise propagation over the iterations, as well as to speed up the convergence.

Non-linear alternatives are discussed next, some of which are able to circumvent the aforementioned issues by setting constraints on the restored image.

5.3.2 Non-linear methods

Several alternative iterative approaches have arisen after the work by Van Cittert, most of which include non-negativity constraints on the restored image. Some methods are even more advanced in that they allow other types of constraints, such as forcing values to fall within specific bounds. We begin our discussion of non-linear methods by looking at Gold's method, which uses multiplicative corrections, as opposed to Van Cittert's additive corrections. We then look at another approach, known as Jansson's method. Several other alternatives exist; the reader is referred to [14, Ch. 4] for a comprehensive treatment.

5.3.2.1 Gold's method

Consider again the formulation $i = s * o$. Given an earlier estimate $\hat{o}^{(k)}$, Gold suggested refining the estimate as

$$\hat{o}^{(k+1)} = \hat{o}^{(k)} \frac{i}{s * \hat{o}^{(k)}}. \tag{5.17}$$

Gold's method is usually initialized the same way as Van Cittert's method, i.e., $\hat{o}^{(0)} = i$. If i and s are non-negative, we see that $\hat{o}^{(k)}$ is also non-negative for all $k \geq 0$. Note that even if i is non-negative in the absence of additive noise, it may become partially negative when noise is present. In such cases there is no guarantee for $\hat{o}^{(k)}$ to be non-negative.

As for convergence properties, Gold showed that proper convergence occurs for (5.17) if $\boldsymbol{S}$ is positive definite. In fact, the positive definiteness constraint can be somewhat relaxed, but we omit such details here. Unlike Van Cittert's method, this method does not in general converge to the inverse-filter estimate due to the non-negativity imposed.

5.3.2.2 Jansson's method

Using the previously described methods, the restored image may sometimes contain spurious peaks due to noise. Therefore, it would be useful to impose an upper bound on allowed values, in addition to the non-negativity constraint. A first approach to achieve this is to simply set non-physical parts of the image equal to zero, where non-physical refers to values outside a pre-specified interval. Formally, this is done by replacing the constant C in (5.15) by a threshold function, which depends on the previous estimate $\hat{o}^{(k)}$. Doing this we obtain

$$\hat{o}^{(k+1)} = \hat{o}^{(k)} + r\left(\hat{o}^{(k)}\right)\left(i - s * \hat{o}^{(k)}\right), \tag{5.18}$$

where

$$r\left(\hat{o}^{(k)}\right) = \begin{cases} r_0 & \text{for } \hat{o}^{(k)} \text{ physical} \\ 0 & \text{elsewhere} \end{cases} \tag{5.19}$$

Jansson found that one should generally choose r_0 large enough so that $\hat{o}^{(k)}$ will be significantly corrected in each iteration, making it approach o more quickly. Choosing such a large r_0 may result in some initial overcorrections, but these usually damp out after a few more iterations. One must be careful not to choose r_0 too large, however, since then the method risks not converging at all. Trial-and-error is usually performed to find a good r_0, even though there exist theorems regarding the eigenvalues of $\boldsymbol{S}$ establishing the optimum r_0.

Choosing the function r according to (5.19) has a major disadvantage. Due to the strict border between physical and non-physical in (5.19), values which are once overcorrected will remain constant for all subsequent iterations. Therefore, it is justified to seek other candidates for r, which involve a smoother transition between physical and non-physical. Such functions are known as *relaxation functions*.

Jansson argued that it should be possible to simply flip the sign of the correction corresponding to non-physical regions of the image. Let $P = [0, d]$ denote the physical interval, where $d \geq 0$. Jansson then introduced a functional form for the relaxation function that varies linearly from a maximum value at $d/2$, to zero at the bounds 0 and d, and negative beyond. Formally, such a function looks as follows,

$$r\left(\hat{o}^{(k)}\right) = r_0\left(1 - \frac{2}{d}\left|\hat{o}^{(k)} - \frac{d}{2}\right|\right). \tag{5.20}$$

There is no reason to confine r to be a linear relaxation function. For example, a power function can be used,

$$r\left(\hat{o}^{(k)}\right) = 4r_0\left(\hat{o}^{(k)}\left(d - \hat{o}^{(k)}\right)\right)^n, \tag{5.21}$$

where n is an integer exponent. An exponential relaxation function may also be used,

$$r\left(\hat{o}^{(k)}\right) = c_1 e^{-c_2\left(x-\frac{d}{2}\right)^2} + c_3, \tag{5.22}$$

where $c_2 > 0$, $c_1 = r_0(1 - \exp(-c_2/4))^{-1}$ and $c_3 = r_0 - c_1$. Illustrations corresponding to (5.19) - (5.22) are shown in Figure 5.1.

Note that there is no universal guarantee for convergence using Jansson's method. However, experimental results indicate that convergence occurs for many choices of r [14, p. 124]. Jansson's method is usually initialized as Gold's and Van Cittert's methods, i.e., $\hat{o}^{(0)} = i$.

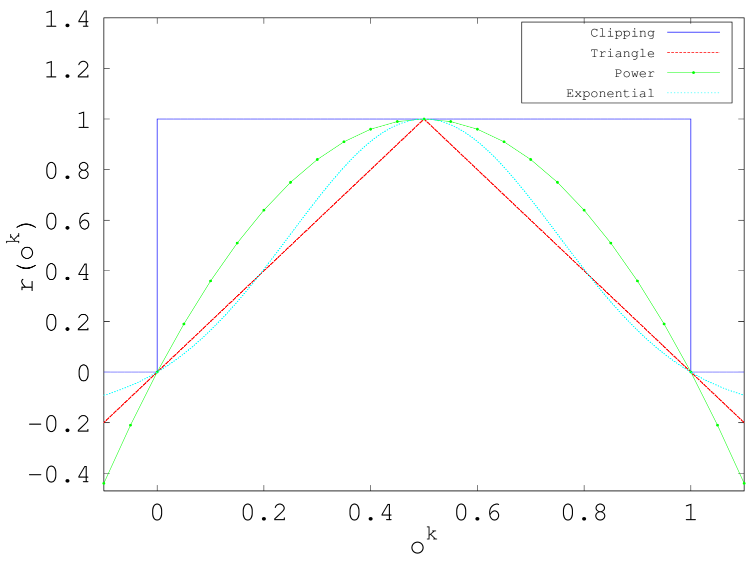


Figure 5.1: Different relaxation functions in Jansson's method. The physical range is $[0, 1]$ and $r_0 = 1$. For the power function, $n = 1$ and for the exponential function, $c_2 = 8$.

5.4 Improving efficiency of iterative methods

All the iterative methods are on the form

$$\text{next estimate} = \text{previous estimate} \oplus \text{correction}, \tag{5.23}$$

where $\oplus$ denotes addition or multiplication; see for example (5.16) and (5.17). The correction term involves computing a prohibitively expensive shift-variant convolution $s * \hat{o}^{(k)}$ in each iteration. Specifically, we realize by looking at the matrix formulation (5.8) that the cost for computing $s * \hat{o}^{(k)}$ for an image of size $m \times n$ is $\mathcal{O}\left((mn)^2\right)$. In this section we discuss several techniques for performing this convolution more efficiently.

5.4.1 Approximating shift-variance using shift-invariant patches

An intuitively appealing approximation technique is to divide the image into rectangular patches in which shift-invariance is assumed; this is done in [2] and [3]. The shift-invariant kernel for a specific patch is obtained by evaluating $s\left(x, y; x', y'\right)$ in the center of the patch, as shown in Figure 5.2. Once we have obtained such shift-invariant kernels, patch-wise convolution can be performed much more efficiently[1] using FFT, as discussed in Section 5.2.

[1]It is well-known that for a shift-invariant PSF, the cost for computing $s * \hat{o}^{(k)}$ for an $m \times n$ image using FFT is $\mathcal{O}\left(mn \log(mn)\right)$. Compare this to $\mathcal{O}\left((mn)^2\right)$, which is the cost for the shift-variant case.

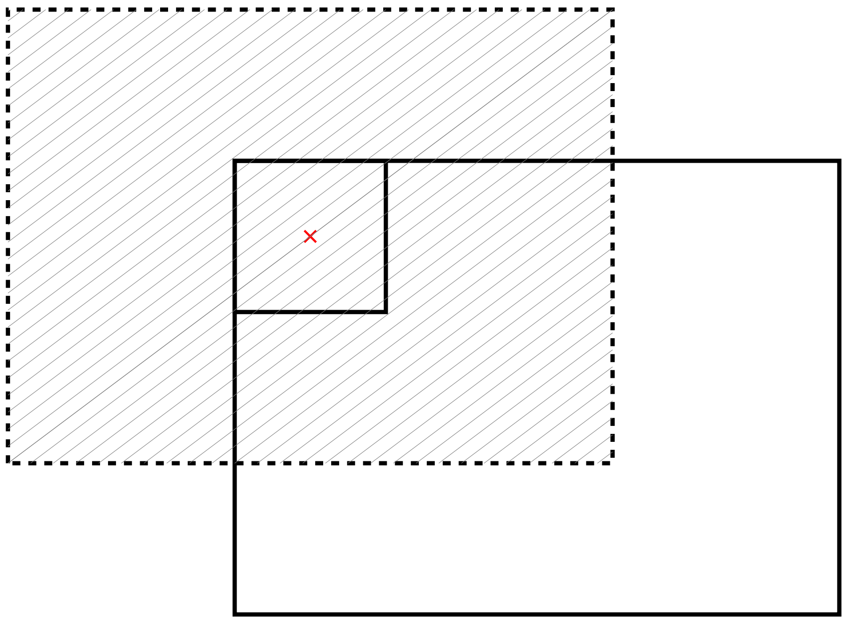


Figure 5.2: In order to reduce the computational complexity, local shift-invariance is assumed within each patch (small solid-line rectangle). The shift-invariant kernel, which is shown as the dashed rectangle with diagonal lines, is obtained by evaluating $s(x, y; x', y')$ in the center of the patch (red cross). The solid-line large rectangle is the image.

Simply dividing the image into non-overlapping patches might result in unwanted edge effects appearing in the restored image. In the *overlap-add method* this is remedied by allowing the patches to overlap [13, Ch. 18]. In the overlapping regions a weighted average is computed. We use linear weights, but it is also possible to apply e.g. Gaussian weighting. In total, four different weighted regions must be considered; see Figure 5.3. The overlap-add method lends itself very well to parallelization, since the work carried out in each patch is independent.

Note that we are not restricted only to iterative methods when restoring a patch, but we can choose any deconvolution method. This includes the filter inversion techniques described in Section 5.1. Hence, the overlap-add method allows shift-invariant techniques to be applied on shift-variant problems.

5.4.2 Kernel thresholding

Even when assuming local shift-invariance, performing the convolution $s * \hat{o}^{(k)}$ can be very time-consuming. To speed up the convolution, one may use *kernel thresholding*. The PSF will generally be fast-decaying and quickly become small, and therefore one could choose to omit image pixels deemed to be insignificant for the computation of the convolution.

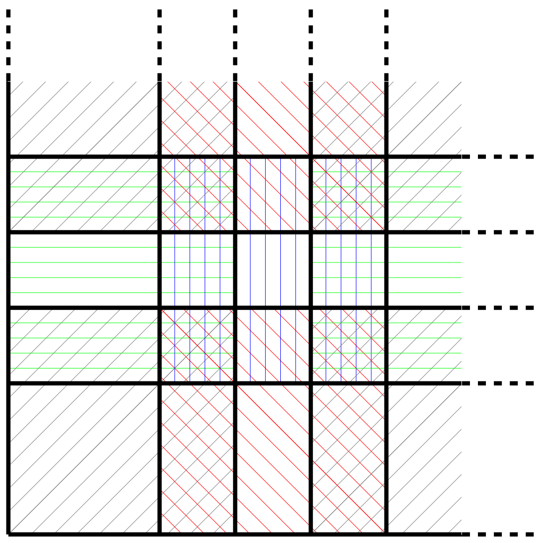


Figure 5.3: Illustration of the overlapping regions in the overlap-add method. In regions with only one type of stripe, no overlap occurs; in regions with two types of stripe, overlap occurs between two patches; and in regions with four types of stripe, overlap occurs between four patches. Linear averaging is used in the overlapping regions.

One way to determine if a pixel is insignificant is to check if its corresponding PSF-weight is below some chosen threshold. This kind of thresholding might be troublesome if a large number of pixels have small corresponding PSF-weights, since the cumulative sum may be significant. To avoid this issue, one can instead consider an area (e.g., a rectangle) around the peak of the PSF, and check if the sum of the intensities in the area is larger than some threshold fraction of the sum of the intensities in the entire PSF. For example, one can choose to keep only the part of the PSF lying within a rectangle around the peak of the PSF containing 99% of all the intensities in the PSF. Since the PSF is quickly decaying, this means that the PSF will be replaced by a very small approximate PSF.

Important to note is that one should not blindly focus on the PSF-weights. Looking at 2.2, it is clear that the product of the PSF-weights and the underlying image is what matters in the end. In a relatively constant image, this should not pose any issues, but it might yield some unwanted artifacts in images with regions of significantly different intensities. Moreover, if there are many terms in (2.2), the sum may still be significant even though each term is insignificant by itself.

5.4.3 Image downsampling

The cost for computing $s * \hat{o}^{(k)}$ is intimately connected to the image size. Therefore, downsampling the observed image will significantly increase the computational efficiency.

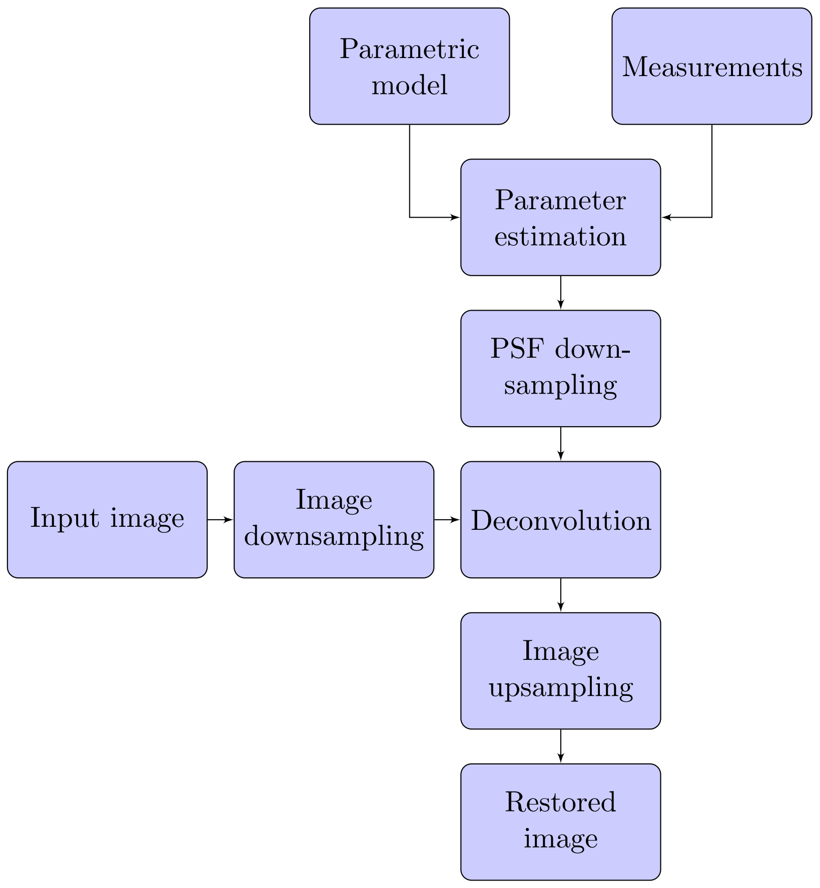


Figure 5.4: Extended stray light compensation framework, now including down- and upsampling steps. Compare with Figure 1.5.

This strategy is illustrated in Figure 5.4. Note that the PSF is downsampled as well, in order to make sense relative to the smaller image.

Downsampling low-pass filters the image and thus the high-frequent portion of the image is lost and cannot be corrected. For this reason, the compensation might yield some unwanted artifacts in regions with rapid changes in intensity. In more slowly varying parts of the image, the stray light compensation will hopefully be almost unaffected by the downsampling (except for the unavoidable decrease in image quality owing to the downsampling itself). An additional advantage of downsampling is that high-frequent noise is reduced, which should be beneficial for the more noise-sensitive deconvolution methods.

5.4.4 Selected-ordinates sampling

In US patent [15], a certain non-uniform sampling method called *selected-ordinates sampling* is used to speed up the convolution $s * \hat{o}^{(k)}$. For simplicity, we describe the method in one dimension; generalizations to higher dimensions are straightforward.

Begin by considering the weighted integral

$$g(x) = \int w(x; x')f(x')\mathrm{d}x', \tag{5.24}$$

where w is a weight function (e.g., a point spread function) and f is a function to be integrated. Assume we want to evaluate $g(x)$ on some discrete range $X = \{x_1, x_2, \ldots, x_M\}$. In the stray light compensation framework, X denotes the digital image plane, and $x_1, \ldots, x_M$ are its M pixels. Given a pixel $x_k \in X$, standard rectangular approximation (see Figure 5.5) yields

$$g_k = \int w(x_k; x')f(x')\mathrm{d}x' \approx \sum_{i=1}^{N_k} w_k^{(i)} f^{(i)} \Delta x_k'^{(i)}, \tag{5.25}$$

where $\Delta x_k'^{(1)}, \Delta x_k'^{(2)}, \ldots, \Delta x_k'^{(N)}$ are the spacings between the N_k samples $x_1', \ldots, x_{N_k}'$. Note that we use the convenient notation $g_k = g(x_k)$, $w_k^{(i)} = w(x_k; x_i')$ and $f^{(i)} = f(x_i')$. Since the number of samples N_k may be different for each $x_k \in X$, the sum is balanced by dividing with the number of samples, i.e. (5.25) is replaced by

$$g_k \propto \frac{1}{N_k} \sum_{i=1}^{N_k} w_k^{(i)} f^{(i)} \Delta x_k'^{(i)}, \tag{5.26}$$

where the proportionality relation $\propto$ is now more appropriate. For example, if this normalization is not performed, the overlap-add method might wrongly yield patches of different mean magnitudes in the restored image.

For fix x, one can interpret (5.24) as computing the area under the function $q(x') = w(x; x')f(x')$, which is discretized in (5.25). If we assume uniform sampling we have to use some minimum number of samples to obtain a certain precision. Since q rarely is a constant function, we probably have to use an excessive amount of samples to sufficiently capture its variational characteristics, leading to a large computational burden. On the other hand, if we assume non-uniform sampling, we can shift some of the work to the sampling process. One way to achieve this is to set

$$\Delta x_k'^{(i)} = \frac{1}{w_k^{(i)}}, \tag{5.27}$$

which implies that we sample more frequently where q varies a lot. This is because variations in q are attenuated when w is small and increased when w is large. Choosing $\Delta x_k'^{(i)}$ according to (5.27), we see that (5.26) reduces to

$$g_k \approx \frac{1}{|S_k|} \sum_{i \in S_k} f^{(i)}, \tag{5.28}$$

where S_k denotes the set of selected samples for the kth pixel and $|S_k|$ denotes the number of elements in S_k. If the kernel is shift-invariant we only need to compute S_k for one k^* since the remaining sample sets can obtained by an appropriate shift of S_{k^*}. We see from (5.28) that given each S_k, the cost for computing $s * \hat{o}^{(k)}$ for a $m \times n$ patch is $\mathcal{O}(mn|\tilde{S}_k|)$, where $\tilde{S}_k$ denotes the mean number of elements of the sets S_k.

It should be noted that we do not have access to the continuous image, since the digital camera has already sampled the scene to a finite set X of pixels. Hence, (5.24) should instead be replaced by

$$g_k = \sum_{i \in X} w_k^{(i)} f^{(i)}, \tag{5.29}$$

Now, an approximation formula as that in (5.25) can be found by noting that

$$g_k = \sum_{i \in X} w_k^{(i)} f^{(i)} = \sum_{i \in X} w_k^{(i)} f^{(i)} d_k^{(i)}, \tag{5.30}$$

where $d_k^{(i)} = 1$ for all $i \in X$ is a uniform step-length. An approximation can be obtained by allowing non-uniform $d_k^{(i)}$ and using a different sample set X. Setting $d_k^{(i)} = 1/w_k^{(i)}$ we arrive at (5.28). See Appendix B for a discussion on this kind of sampling.

5.4.4.1 Generating samples

In [15], several methods are suggested for approximately achieving the sampling given in (5.27). We propose using *rectangular sampling*, which satisfies (5.27) up to rounding[2]. To explain this method, we begin by considering the two-dimensional analogue of (5.27),

$$\Delta x_k'^{(i)} \Delta y_k'^{(i)} = \frac{1}{w_k^{(i)}}, \tag{5.31}$$

where $\Delta y_k'^{(i)}$ has been included since cuboids are used to approximate the volume, as opposed to rectangles approximating the area in (5.25); see Figure 5.5. In the following we assume that the PSF has been evaluated at some pixel i; denote this by $s_i(x, y)$. Rectangular sampling starts by selecting the sample (x_{max}, y_{max}) at the peak of the PSF. Next, the four neighbors at distance $d = \left\lceil \sqrt{1/w_0}/2 \right\rceil$ are selected, so that the sample-hold area comes as close to satisfying (5.31) as possible, while at the same time yielding a symmetrical neighborhood. All five coordinates are inserted to a coordinate

[2]Assuming an image of infinite size. As we will see, the image boundaries will pose some unavoidable issues.

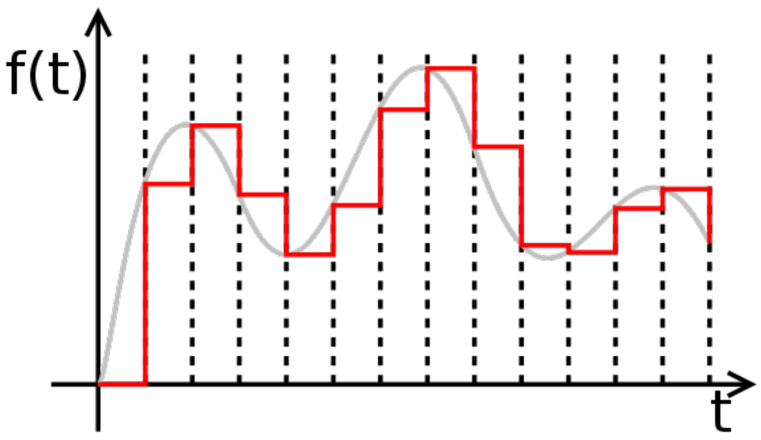


Figure 5.5: Illustration of sample-and-hold, one variant of rectangular sampling, with uniform sampling distance in the one-dimensional case. Line steps are taken, yielding rectangles approximating the area under the curve. In the two-dimensional case, steps are taken along two coordinate axes, yielding rectangular patches as seen from above. These rectangular steps yield cuboids approximating the volume under the two-dimensional surface. Image source: http://en.wikipedia.org/wiki/Sample_and_hold.

matrix $\boldsymbol{C}$, so that

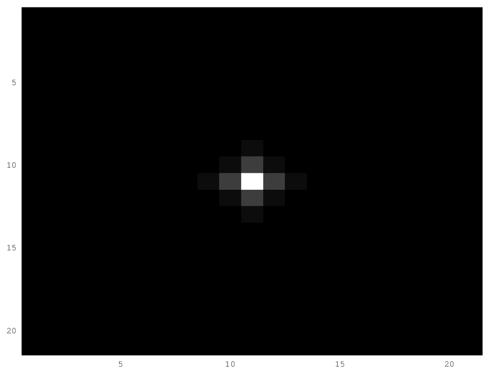
$$\boldsymbol{C} = \begin{pmatrix} x_{max} & y_{max} \\ x_{max} + d & y_{max} \\ x_{max} & y_{max} + d \\ x_{max} - d & y_{max} \\ x_{max} & y_{max} - d \end{pmatrix} \tag{5.32}$$

where $w_0 = s_i(x_{max}, y_{max})$. See Figure 5.6 for an illustration of the initialization process, as well as the rest of the sampling procedure. The rest of the process is also described in pseudo-code in Algorithm 1.

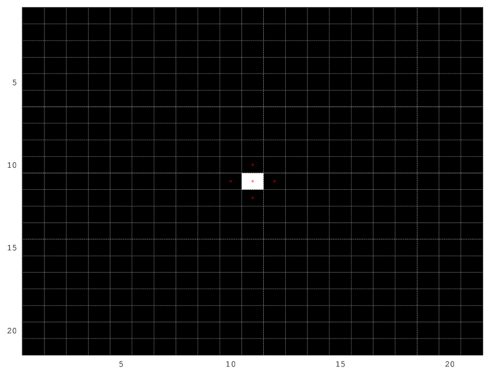
As can be seen in Figure 5.6, some parts are not covered by any area. This is due to the finite extent of the image. To remedy this issue we can loosen the constraint that the area generated by a pixel must lie completely within the image boundaries. A weaker constraint is to allow a proposed sample if a specific fraction of its corresponding area is inside the image. Figure 5.7 shows an alternative result to that shown in Figure 5.6f. This result is obtained by setting the constraint that only 25% of the area of generated pixels must lie within the image boundaries. Note however, that this will sometimes violate (5.31).

5.4.4.2 Handling edge-effects

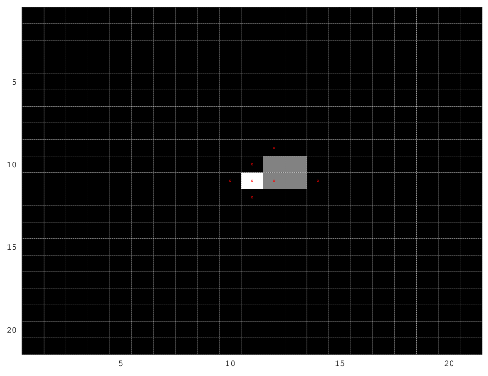
A problem when combining the approximation (5.28) with the iterative compensation methods described in Section 5.3 is illustrated in Figure 5.8. Unwanted edge-effects



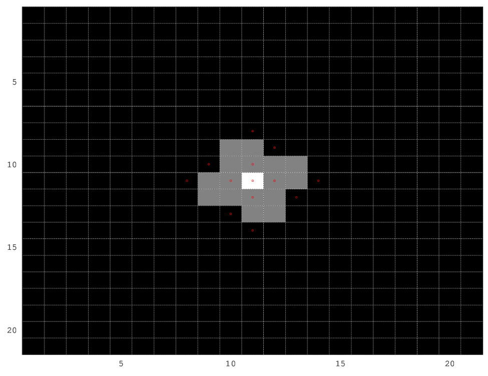
(a) Example of s_i as seen from above. The peak is the white pixel in the middle and has value $w_0 = 1$. The four neighbors have values $w_1 = 1/4$.



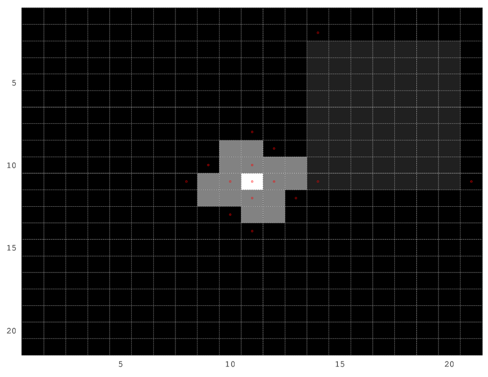
(b) Rectangular sampling is initialized by selecting the peak coordinate and the neighbors at distance $\left\lceil \sqrt{1/w_0}/2 \right\rceil = \lceil 1/2 \rceil = 1$. All five pixels are inserted to a matrix $\boldsymbol{C}$. The area covered by the peak pixel is 1.



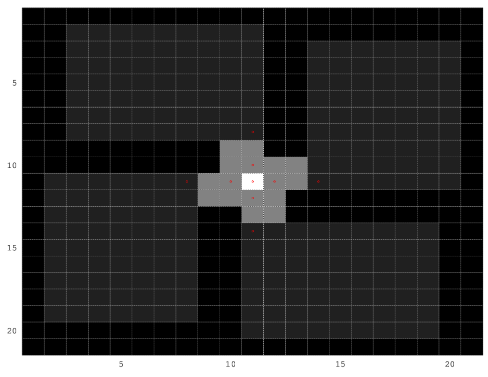
(c) Each neighbor now tries to take a rectangular step, i.e., a step satisfying (5.31) up to rounding. The area generated by a rectangular step is not allowed to overlap with any previous area, and it must fit within the image. In this case, the rectangular step succeeds and yields two new pixels, which are inserted to $\boldsymbol{C}$. The area covered is $1/w_1 = 4$.



(d) The three remaining neighbors to the peak pixel do the same, yielding in total 6 additional pixels, which are inserted to $\boldsymbol{C}$. All the 8 new pixels generated by the 4 neighbor pixels have value $w_2 = 1/64$.



(e) The same thing is repeated for the new pixels. In this case, a rectangular step succeeds, yielding two new pixels which are inserted to $\boldsymbol{C}$. The area covered is $1/w_2 = 64$.



(f) The process is repeated for all pixels that have been inserted to $\boldsymbol{C}$. The pixels that fail to take a rectangular step are excluded from the final set of samples. This figure shows the 9 pixels which are chosen at the end. Note that two of the rectangles are not squares, which is fine as long as the area approximately satisfies (5.31).

Figure 5.6: Illustration of rectangular sampling.

Algorithm 1: Rectangular sampling. Initialization of $\boldsymbol{C}$ shown in (5.32).

```
r = 2;
while r ≤ number of rows in C do
    (x, y) ← rth row of C;
    w ← s_i(x, y);
    if possible to take rectangular step then
        take precisely one rectangular step to obtain two additional coordinate pairs;
        for each of the two coordinate pairs not already present in C do
            insert the new coordinate pair to C;
        end
    end
    if no coordinates were inserted to C then
        remove rth row of C;
    end
    r ← r + 1;
end
```

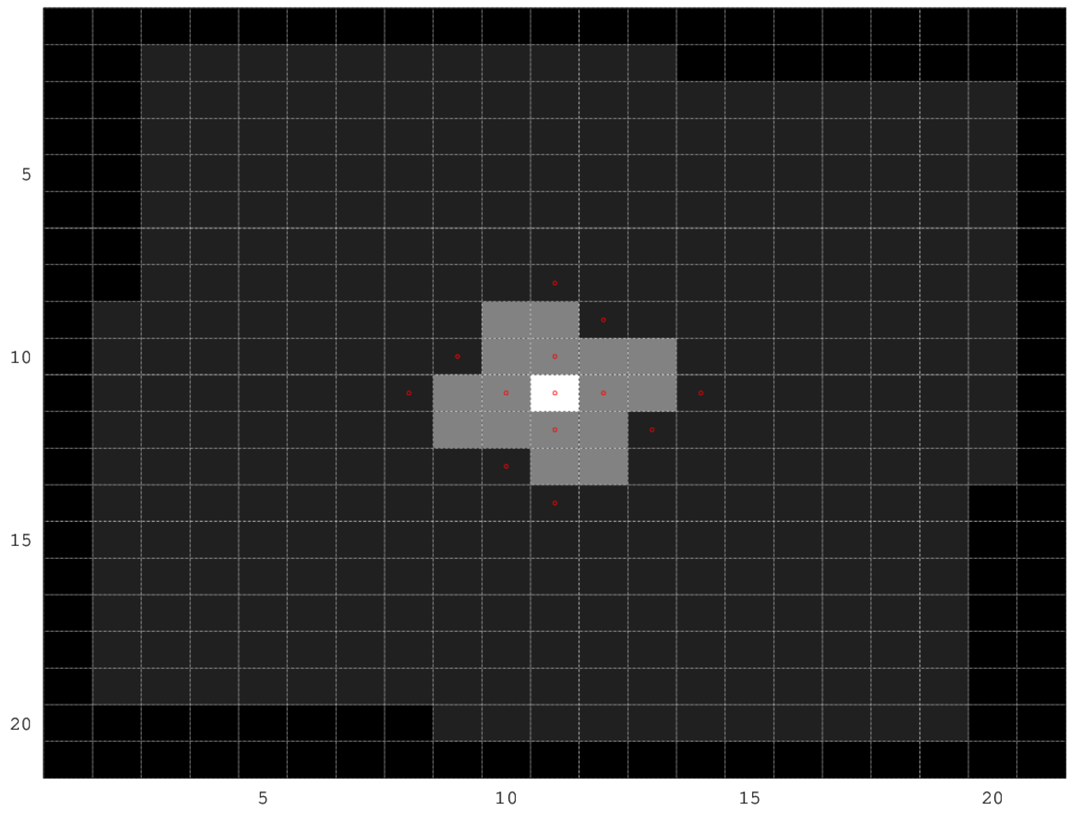
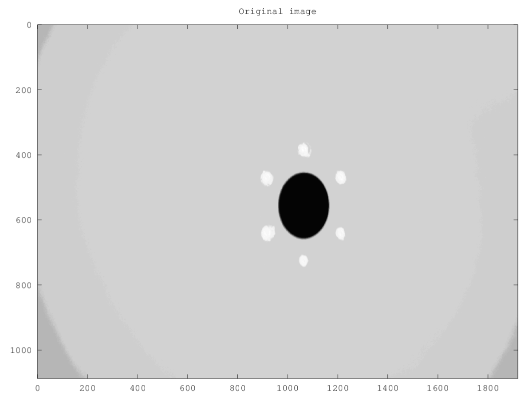
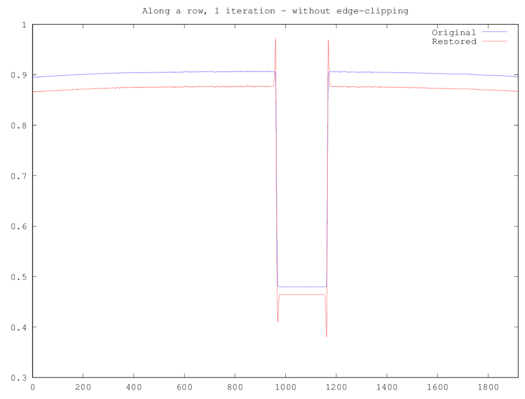


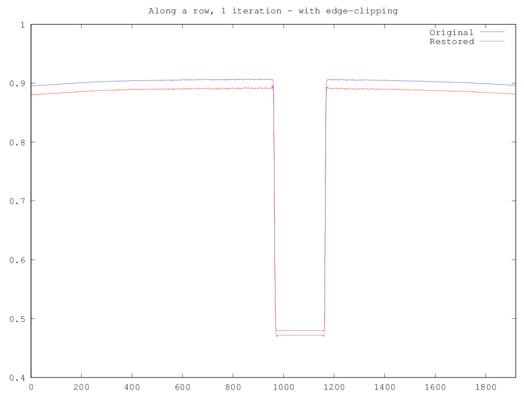
Figure 5.7: By allowing sample candidates to have at least 25% of their corresponding areas within the image, 4 additional samples are added to the final sample set. This should be compared to Figure 5.6f.



(a) Observed image, normalized to [0, 1].



(b) Restored image along one row, without edge-clipping. Notice the edge-effects occurring between the dark and light area of the image.



(c) Restored image along one row, with edge-clipping. Notice how the edge-effects have been significantly reduced. However, the effect of the stray light compensation decreases slightly as well.

Figure 5.8: Illustration of edge-effects and how they can be reduced using edge-clipping.

appear between pixel regions of significantly different intensities (in this case, at the borders between the dark ellipse and the much lighter background in Figure 5.8a). It is possible to remedy this by applying *edge-clipping*, which is explained in Figure 5.9. Edge-clipping refers to excluding pixel values which have significantly higher or lower intensity compared to the pixel being corrected. This affects specific elements of the correction term in (5.23), such that under- and over-corrections are avoided in the restored image. Note that there is a trade-off between edge-clipping and compensation. By excluding many samples, most of the edge-effects disappear, but it also seems to result in slightly decreased stray light reduction; see Figures 5.8b - 5.8c.

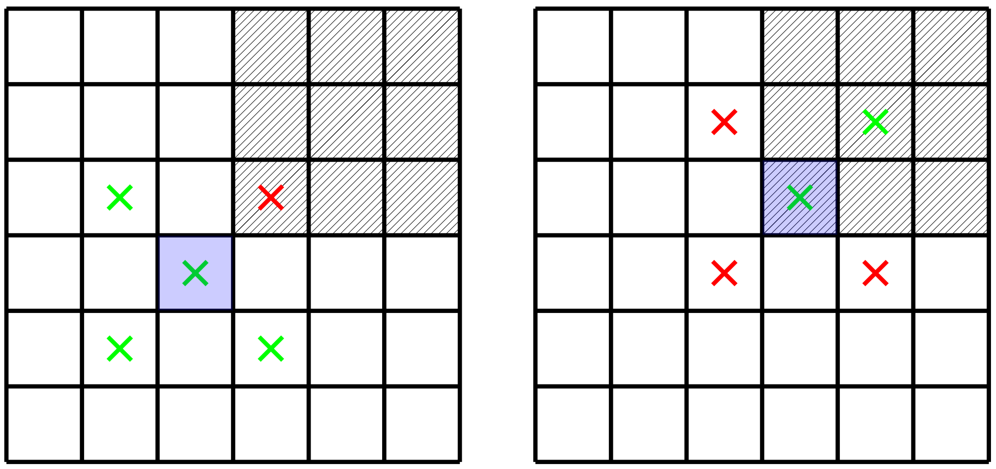


Figure 5.9: To avoid edge-effects in borders between high- and low-intensity pixels, edge-clipping can be applied. Suppose that we want to correct the shaded pixel using the shown neighborhood consisting of five samples. For the left case, this would imply a reduced average in (5.26) due to the pixel inside the dark region, which has significantly lower intensity compared to the shaded pixel. This in turn would imply a high peak in the restored image (under-correction). In the right case, we instead obtain an increased average compared to the value in the shaded pixel, which implies a low peak in the restored image (over-correction). Edge-clipping refers to excluding pixel values which have significantly higher or lower intensity compared to the pixel being corrected. Pixels which should be included in the mean have been marked with green crosses and the rest have been marked with red crosses.

Chapter 6

Results

In this chapter we present results from applying the ideas and methods discussed in earlier chapters. Recall from Chapter 3.1 that the PSF is spectrum dependent. Therefore, we choose to present results only for the green channel. Each green channel image is 544×960 pixels.

When measuring the PSF as discussed in Chapter 3.2.2, we use the $N = 13$ exposure times 0.015, 0.030, 0.059, 0.12, 0.24, 4.7, 9.5, 19, 38, 57, 80, 160, and 320 ms. For averaging out the noise, we perform $M = 100$ measurements at each exposure time. These numbers were suggested to us by our co-supervisors. Recall that this technique can be used for any image, and not just for PSF-measurements. Therefore, to be able to study noise separately, all images shown in this chapter are captured using this technique. Note that this technique yields high dynamic range (HDR) images.

A common problem with image enhancement algorithms is that it is difficult to objectively compare results and measure performance. In hope of achieving as objective comparisons as possible, we do as follows. When presenting the results, we normalize the images to the range $[0, 1]$ and set the color axis to show intensities in the range $[0, 1]$. In some cases it is impractical to use $[0, 1]$ as range for the color axis, for example when high intensity light is present in a relatively small part of the image. If we use a different color axis it will be explicitly stated.

In some experiments the ideal/ground-truth image is available, in addition to the observed and the restored images. This allows us to compute the decrease in error for the restored images relative to the error of the observed image. That is, we define a measure of improvement as

$$\tau_i = \frac{\epsilon^{obs} - \epsilon_i^{res}}{\epsilon^{obs}}, \tag{6.1}$$

Table 6.1: Approximate runtimes per iterations for the different deconvolution methods that are evaluated in this chapter. Note that Wiener deconvolution is the only non-iterative method and therefore its corresponding time/iteration should be interpreted as its total runtime. Also note that all the iterative methods (Gold, Van Cittert and Jansson) have similar runtimes per iteration. For kernel thresholding, image downsampling and selected-ordinates sampling, shift-invariant Gold's method is used.

Type of deconvolution	**Time/iteration (sec)**
Wiener (shift-invariant)	0.3
Iterative (shift-variant)	41400
Iterative (patch-invariant, 8 patches)	1500
Iterative (shift-invariant)	115
Kernel thresholding (99.9%)	3.3
Kernel thresholding (50%)	3.2
Image downsampling (25%)	40
Image downsampling (50%)	8.2
Image downsampling (75%)	0.6
Selected-ordinates sampling (0% edge-clipping)	0.2
Selected-ordinates sampling (80% edge-clipping)	0.3

where i denotes the number of iterations in the iterative methods (this index is of course omitted for non-iterative methods), ϵ^{obs} denotes the error for the observation, and ϵ_i^{res} denotes the error for the ith restored image. Note that τ_i can assume values in the range $(-\infty, 1]$, where 1 means that we have perfectly restored the ideal image. If τ_i becomes negative, it means that the restored image is further from the ideal image, compared to the observed image. We use the Frobenius norm of the difference between two images as measure of error. Note that τ_i will be multiplied by 100, to get the results in percentage.

Table 6.1 shows approximate runtimes per iteration for the different deconvolution methods that are examined. For kernel thresholding, the percentage within parentheses refers to how many percent of the sum of all intensities are within a rectangle centered at the peak of the PSF. Only the part of the PSF within that rectangle is used; values outside are set to zero, since they are considered negligible (recall Chapter 5.4.2). For selected-ordinates sampling, the percentage within parentheses refers to the threshold set for allowed pixels; recall Chapter 5.4.4.2. When 80% edge-clipping is used, only pixels that differ by at most 20% from the current pixel are used. All results were obtained using a 64-bit Ubuntu Gnome 3.4.2 computer, with 16 GB ram and an i7-4770 cpu @ 3.40 GHz x 8.

As can be seen, shift-variant and patch-invariant deconvolution are very time-consuming compared to the other methods. For this reason, we will use shift-invariant deconvolution, where the kernel is evaluated in the middle of the image, unless otherwise stated.

6.1 Full versus approximate optimization

To compare the full optimization scheme in Chapter 4 with its approximate counterpart in Chapter 4.3, we construct nine 27×27 ideal images, each having one pixel equal to 1 and the rest equal to 0, at uniform locations over the image plane. We then generate nine simulated PSF-measurements by filtering each ideal image with the shift-variant filter given in (2.6), with parameters $[\beta, \sigma, a, b, d, f, g] = [1.2, 1.1, 0.002, 0.0025, 0.9, 2.1, 0.9]$.

Results for the full optimization are shown in the first column of Table 6.2. Note that $\beta/d = 1.2/0.9$, so these parameters are correctly estimated up to scaling, which is sufficient since we normalize the kernel. As seen in the second column of Table 6.2, the parameters are not quite as accurate when using the approximate optimization routine, but they are reasonably close to the true parameters. Also, the execution time is significantly decreased.

Table 6.2: Comparison between full and fast optimization. Full optimization yields extremely accurate parameter estimates but takes about 23 times longer than the approximate optimization. The approximate optimization does not yield as accurate parameters, but they look reasonably correct. Note that with respect to β and d, it is the quotient between the two that matters.

Type of optimization	Full	Locally shift-invariant
Estimated parameters	$\beta = 1.1339$	$\beta = 1.1061$
	$\sigma = 1.1000$	$\sigma = 1.0964$
	$a = 0.0020$	$a = 0.0012$
	$b = 0.0025$	$b = 0.0021$
	$d = 0.8504$	$d = 0.8868$
	$f = 2.1000$	$f = 2.1937$
	$g = 0.9000$	$g = 1.0054$
Execution time (sec)	68.4	3.0

6.2 Investigating rotational invariance

In Figure 6.1 we see measurements of the PSF captured close to the corners of the image plane. In this figure we see that the camera seems to be rotationally invariant.

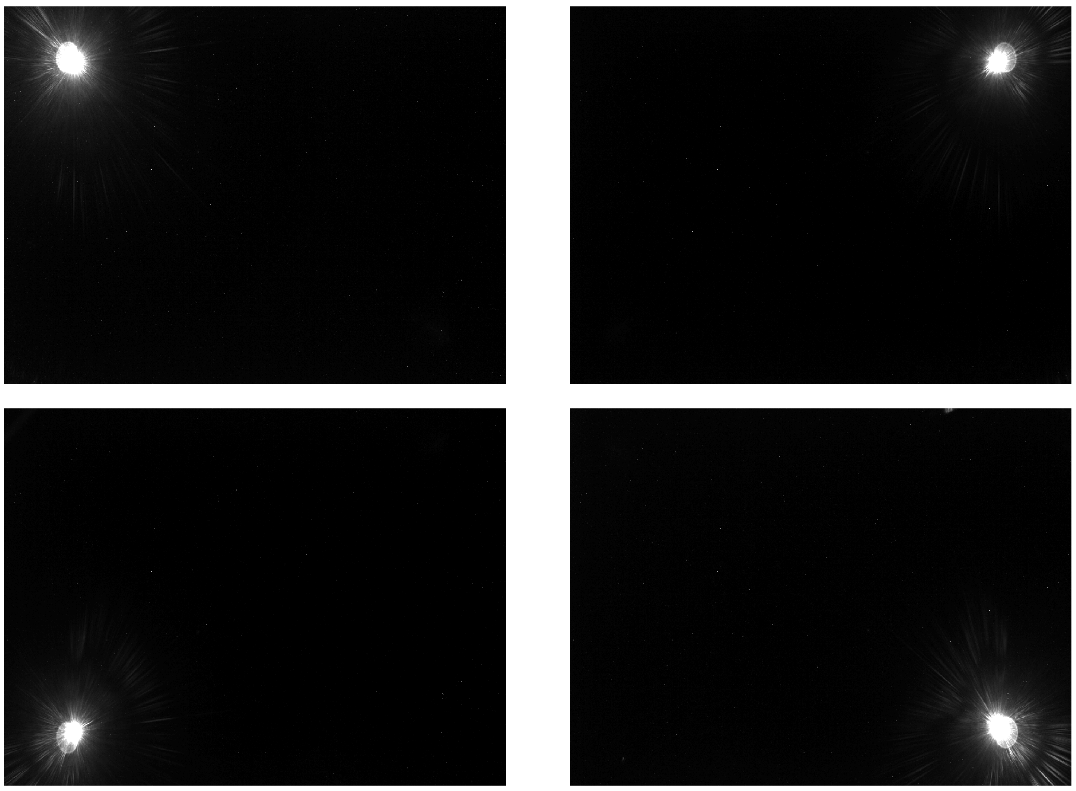


Figure 6.1: Measurements of the PSF from the corners of the image. From these we can conclude that the camera is rotationally invariant. Color axis: $\left[0, 10^{-5}\right]$.

6.3 PSF-parameters

Table 6.3 shows the obtained parameter estimates for the PSF. To obtain these estimates, we use the fast optimization routine in Chapter 4.3. As for (4.2), we set the weights w equal to 1. Moreover, we set I_k to be the entire images, in order to utilize as much PSF-information as possible. Using such large I_k is possible in our case, since the ideal images consist of only one pixel each, making the problem very sparse. Figures 6.2 - 6.3 show the ideal images and estimated point spread functions, respectively.

To see that the estimated parameters are reasonable, we deconvolve two measurements of the point spread function, one in the middle and one closer to the corner of the image plane. In particular, Figures 6.4 - 6.6 show the PSF measured in the middle and the results when deconvolving it using the shift-invariant versions of Gold's, Van Cittert's, Jansson's method and Wiener deconvolution. Accompanying these figures, we have Figure 6.11 showing the improvement using (6.1) (improvement here refers to reduced error with respect to the ideal image i.e., an image with precisely one non-zero pixel). In this figure we have omitted Janssons's method, which is motivated later. Figure 6.11 also contains results of applying shift-invariant deconvolution on a measurement of the PSF located near the corner of the image. Similarly, in Figures 6.8 - 6.10 and Figure

Table 6.3: Estimated PSF-parameters, obtained using fast optimization.

β	σ	a	b	d	f	g
0.826	1.063	-1.514×10^{-5}	-7.427×10^{-6}	1.175	2.127	0.758

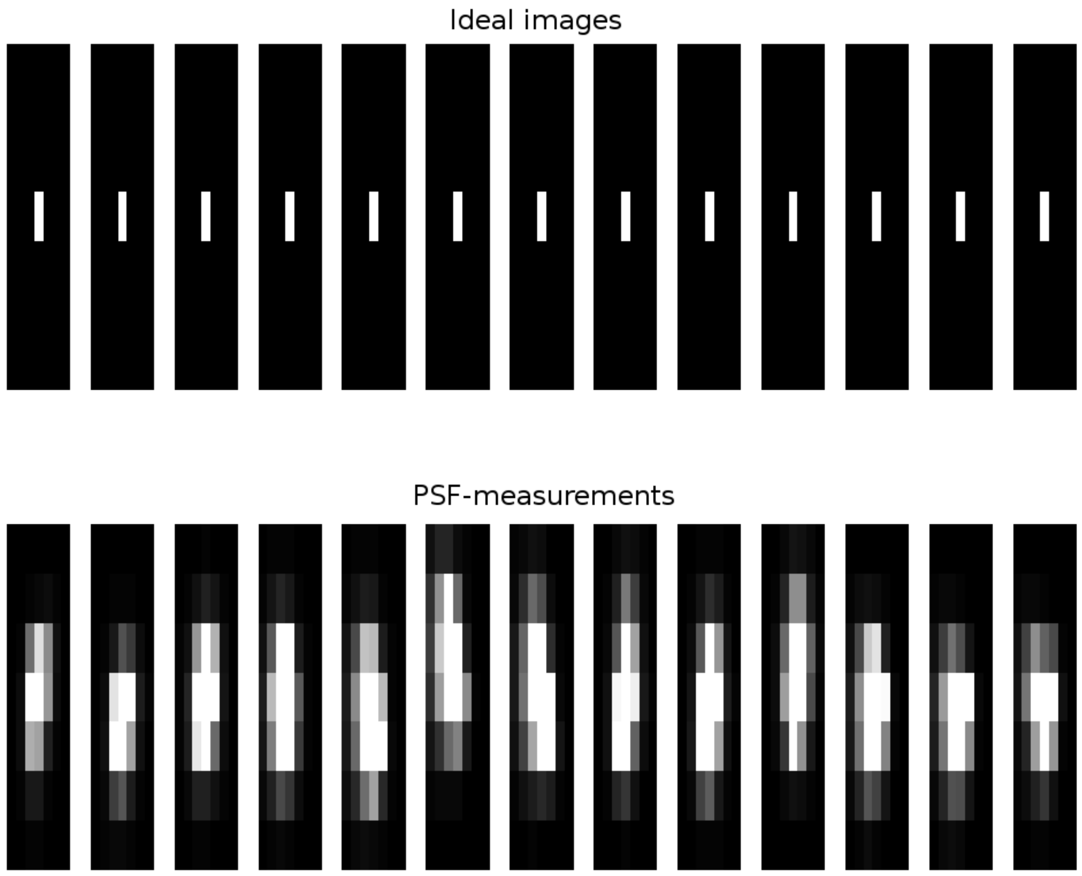


Figure 6.2: Ideal images (positioned at the peaks of the corresponding PSF-measurements) and measured point spread functions. Color axis: $[0, 0.002]$.

6.12 we show the PSF measured near the corner and the results when deconvolving it using shift-invariant, patch-invariant and shift-variant versions of Gold's method.

We see that all methods reduce the smearing, i.e., that the restored images become closer to the ideal image. An exception is seen in Jansson's method (Figure 6.7), which behaves somewhat unexpectedly; it is difficult to find suitable settings for it to yield reasonable results. The shift of the peak suggests that there may be some flaws in the implementation as well. Therefore, we henceforth omit using Jansson's method. We also note in Figure 6.11b that Van Cittert's method does not converge toward the ideal image when the PSF is placed near the corner of the image.

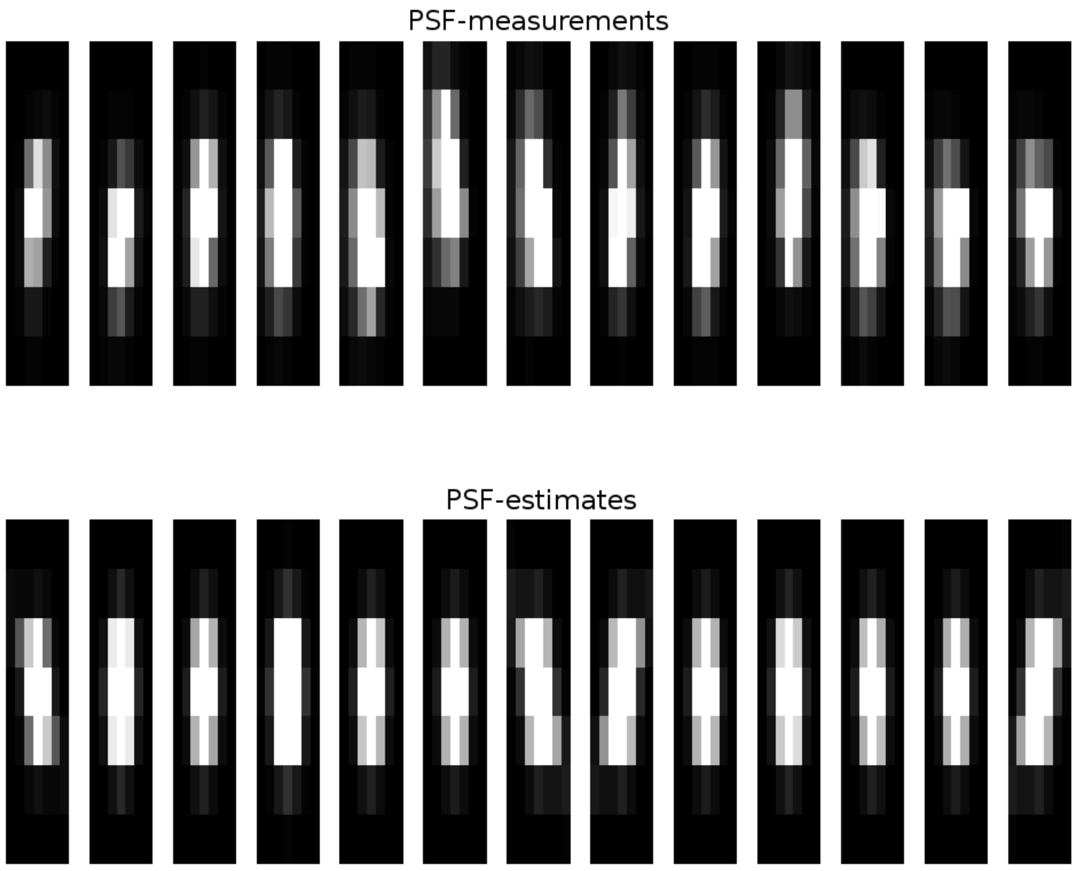
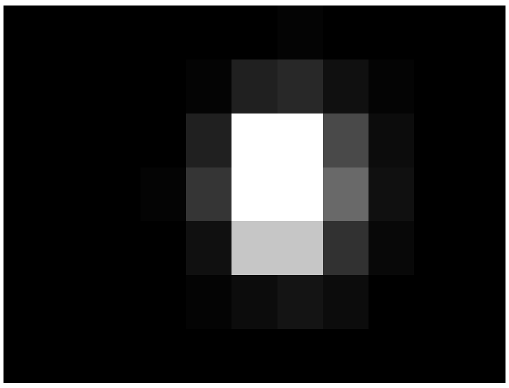
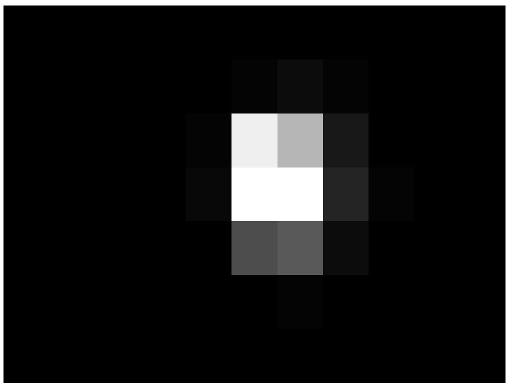
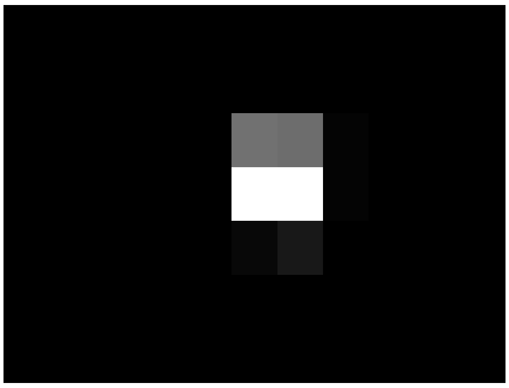
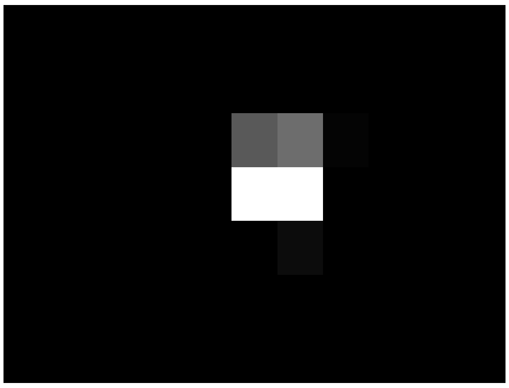


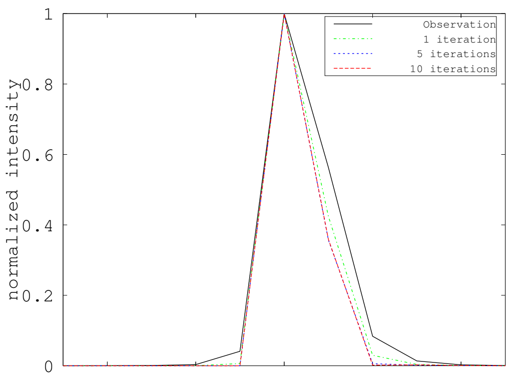
Figure 6.3: Measured and estimated point spread functions. The ideal images shown in Figure 6.2 were used to acquire these results. The estimated parameters are given in Table 6.3. Color axis: $[0, 0.002]$.


(a) Measured PSF.

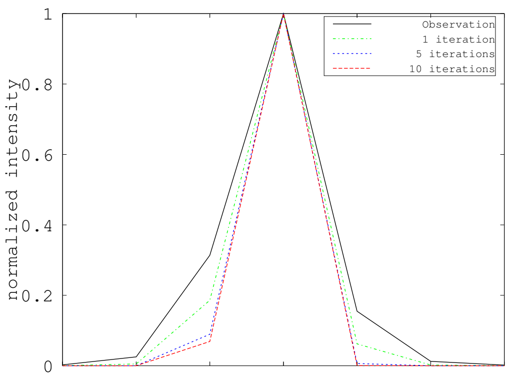

(b) Deconvolved PSF, 1 iteration.


(c) Deconvolved PSF, 5 iterations.


(d) Deconvolved PSF, 10 iterations.

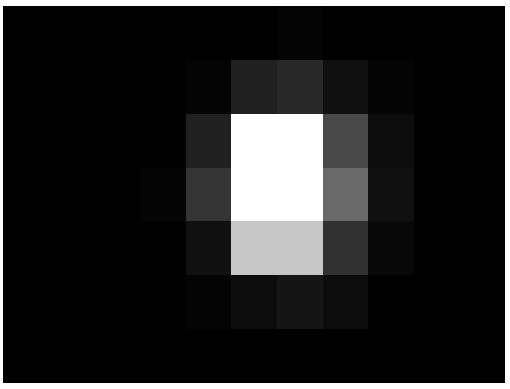


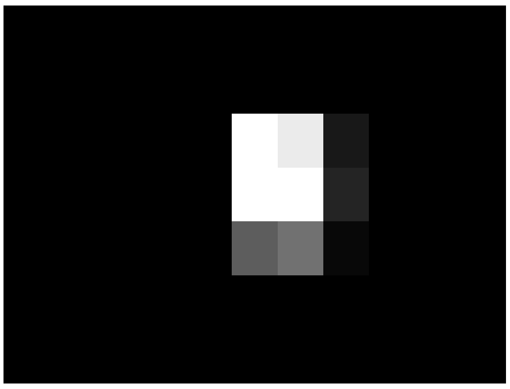
(e) Intensities along the row going through the maximum of the PSF.

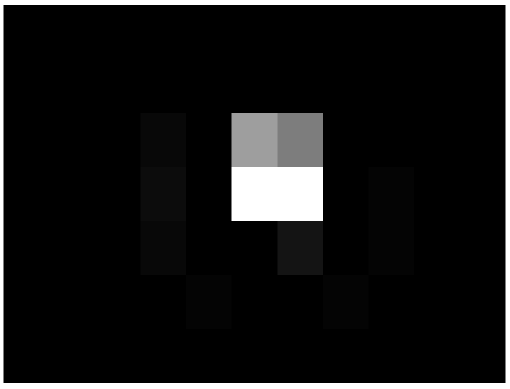


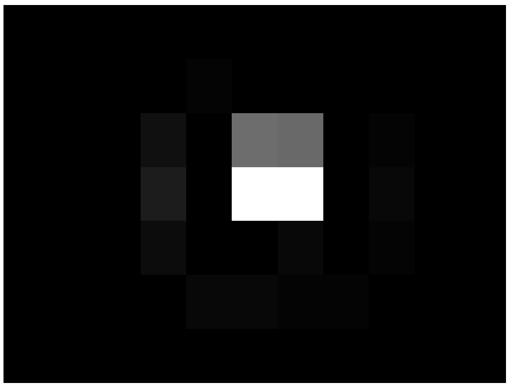
(f) Intensities along the column going through the maximum of the PSF.

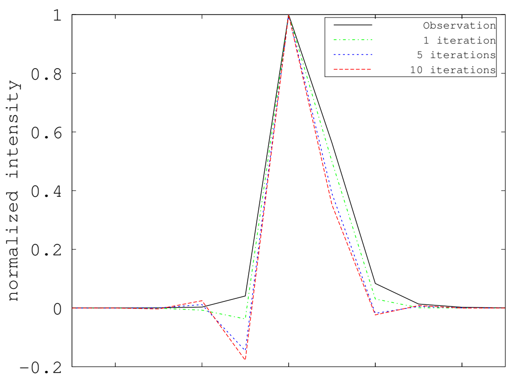
Figure 6.4: Result of the PSF measured in the middle of the image when it is deconvolved using Gold's method. Color axis: $[0, 0.2]$.


(a) Measured PSF.

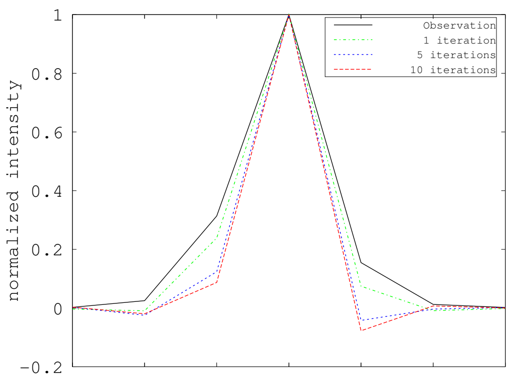

(b) Deconvolved PSF, 1 iteration.


(c) Deconvolved PSF, 5 iterations.


(d) Deconvolved PSF, 10 iterations.

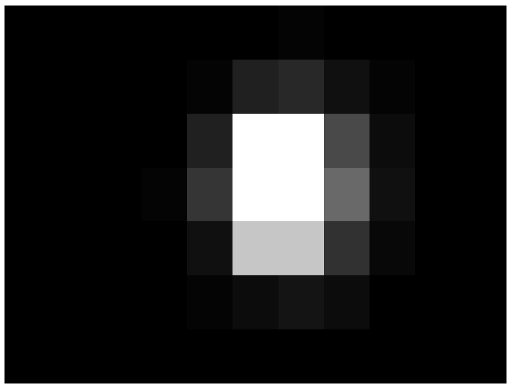


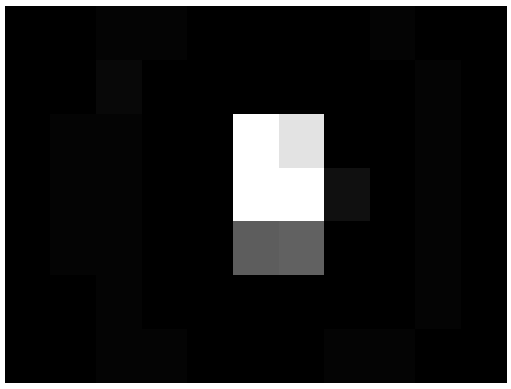
(e) Intensities in the row going through the maximum of the PSF. Note the negativity.

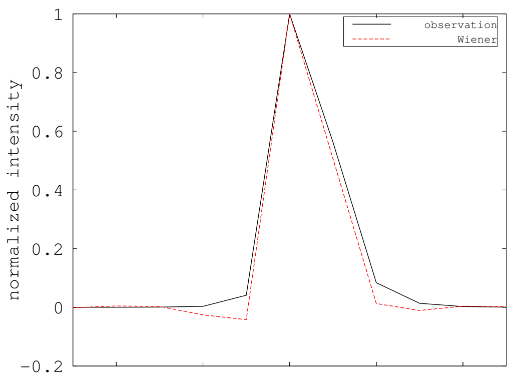


(f) Intensities along the column going through the maximum of the PSF. Note the negativity.

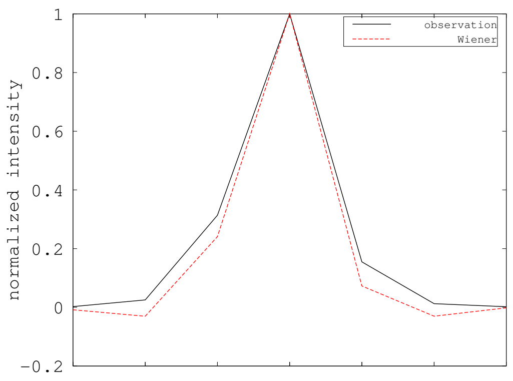
Figure 6.5: Result of the PSF measured in the middle of the image when it is deconvolved using Van Cittert's method. Color axis: $[0, 0.2]$.


(a) Measured PSF.


(b) Deconvolved PSF.

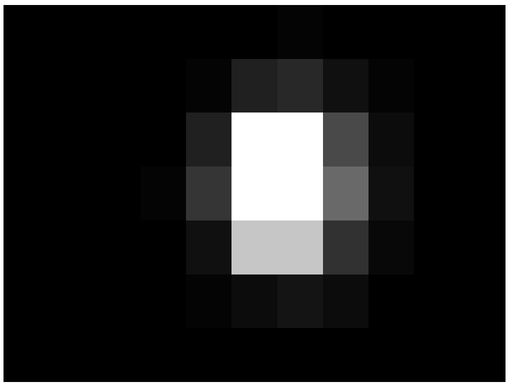


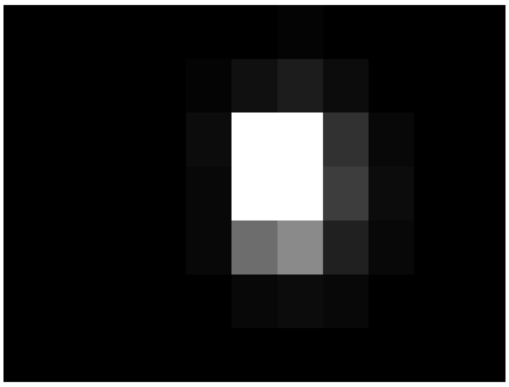
(c) Intensities along the row going through the maximum of the PSF. Note the negativity.

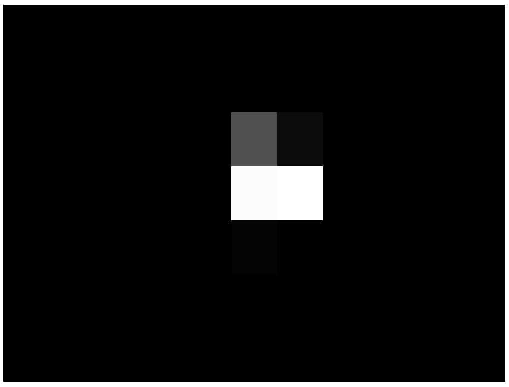


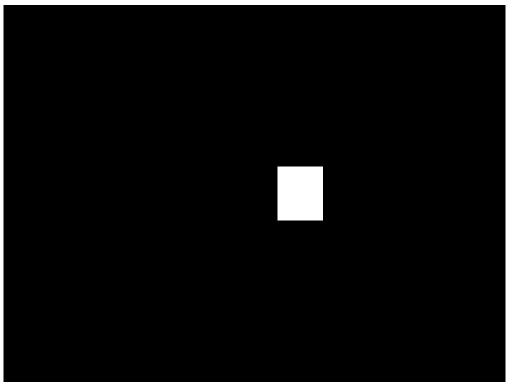
(d) Intensities along the column going through the maximum of the PSF. Note the negativity.

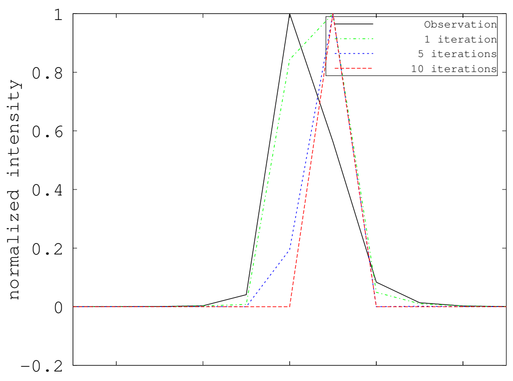
Figure 6.6: Result of the PSF measured in the middle of the image when it is deconvolved using Wiener deconvolution. Color axis: $[0, 0.2]$.


(a) Measured PSF.

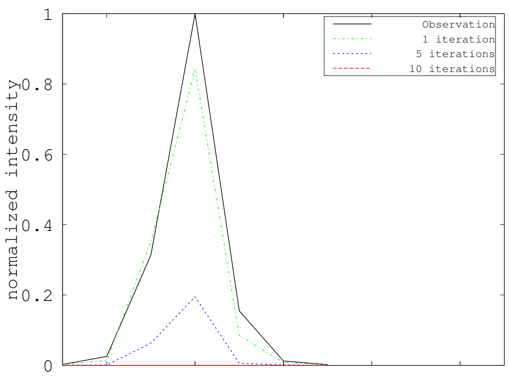

(b) Deconvolved PSF, 1 iteration.


(c) Deconvolved PSF, 5 iterations.


(d) Deconvolved PSF, 10 iterations.

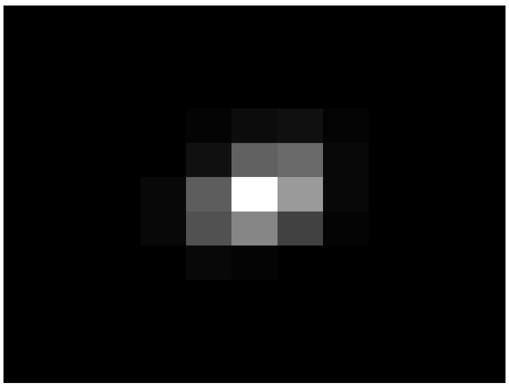


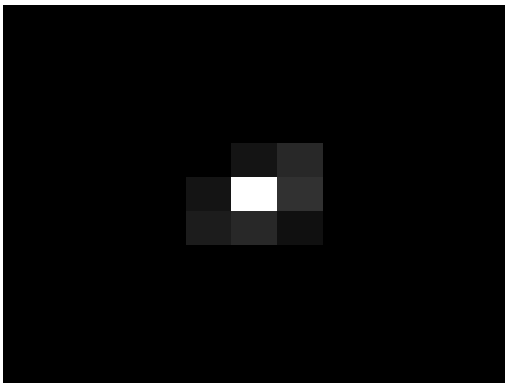
(e) Intensities along the row going through the maximum of the PSF.

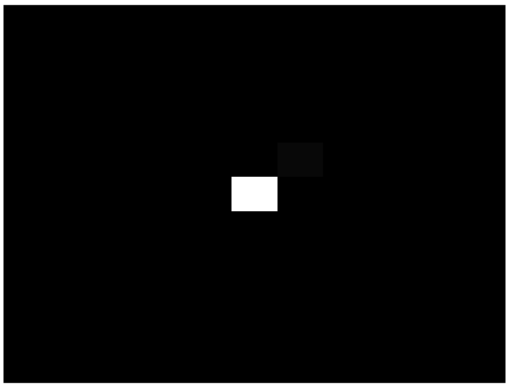


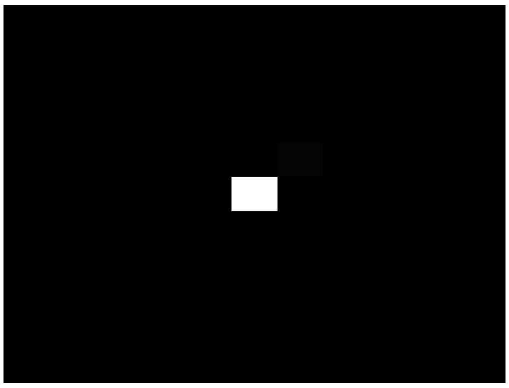
(f) Intensities along the column going through the maximum of the PSF.

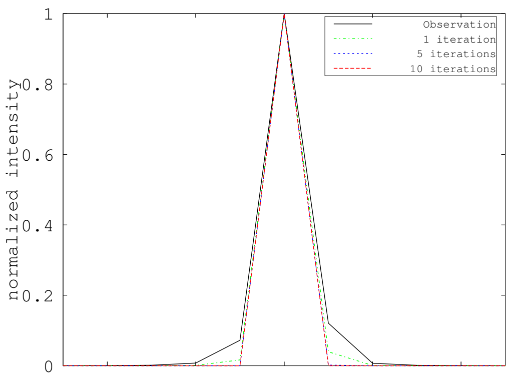
Figure 6.7: Result of the PSF measured in the middle of the image when it is deconvolved using Janssons's method. We used the triangular window given in (5.20), with $r_0 = 4$ and $d =$ maximum intensity in the observed image. Color axis: $[0, 0.2]$.


(a) Measured PSF.

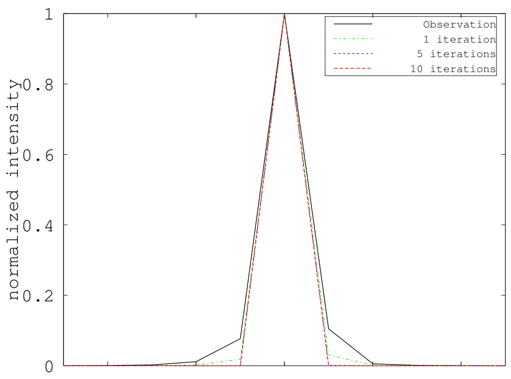

(b) Deconvolved PSF, 1 iteration.


(c) Deconvolved PSF, 5 iterations.


(d) Deconvolved PSF, 10 iterations.

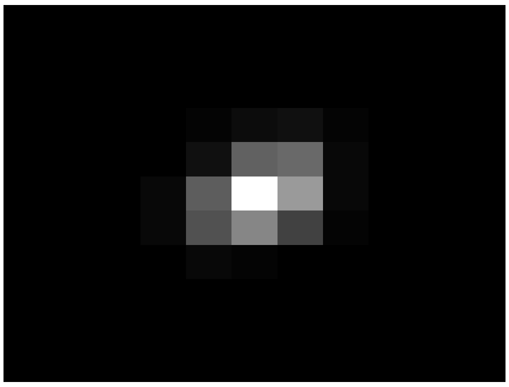


(e) Intensities along the row going through the maximum of the PSF.

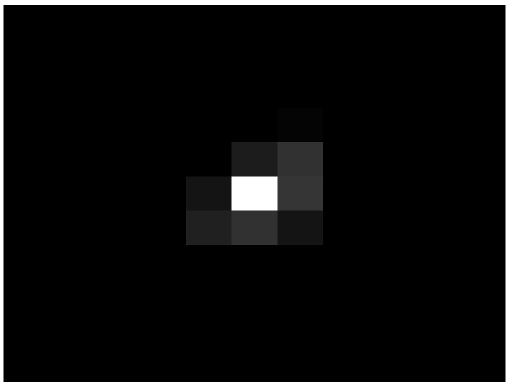


(f) Intensities along the column going through the maximum of the PSF.

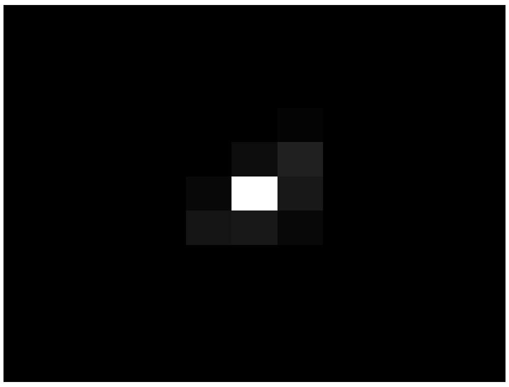
Figure 6.8: Result of the PSF measured close to the upper left corner of the image when it is deconvolved using shift-invariant Gold's method. Color axis: $[0, 0.2]$.



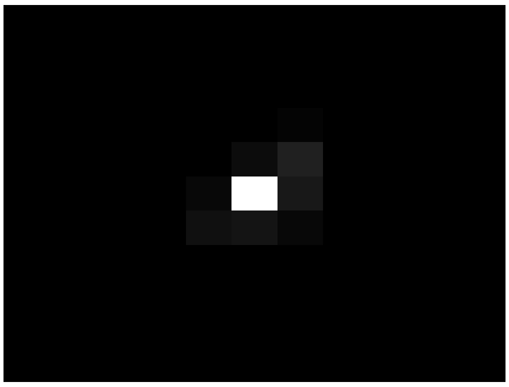
(a) Measured PSF.



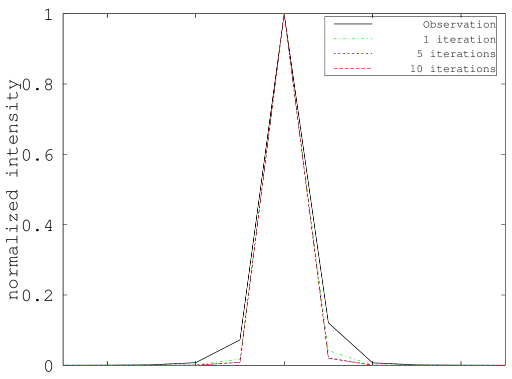
(b) Deconvolved PSF, 1 iteration.



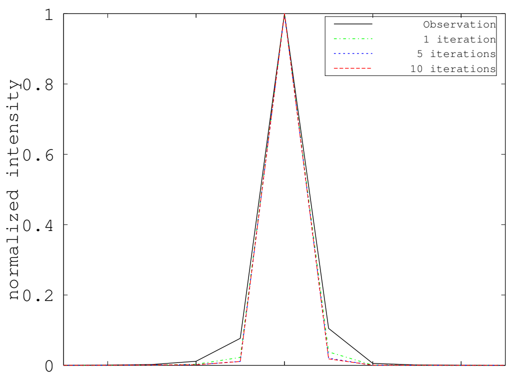
(c) Deconvolved PSF, 5 iterations.



(d) Deconvolved PSF, 10 iterations.

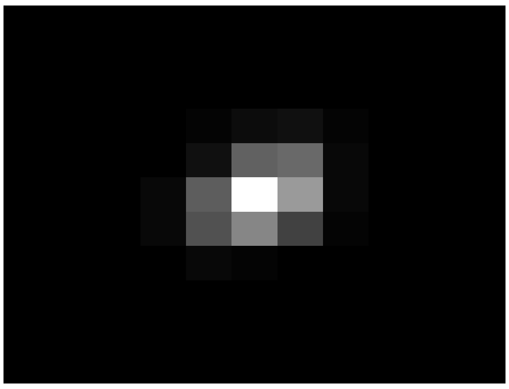


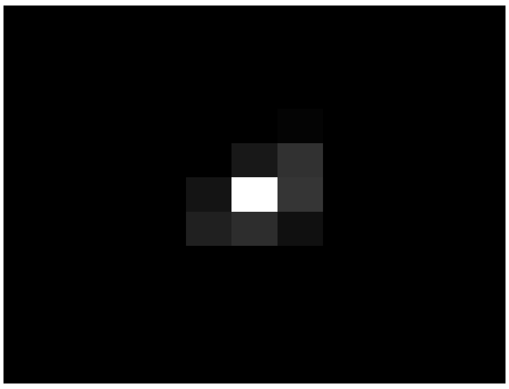
(e) Intensities along the row going through the maximum of the PSF.

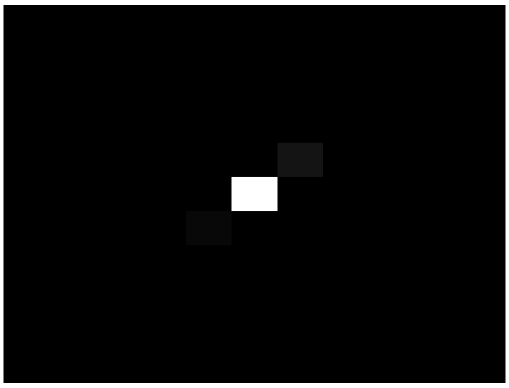


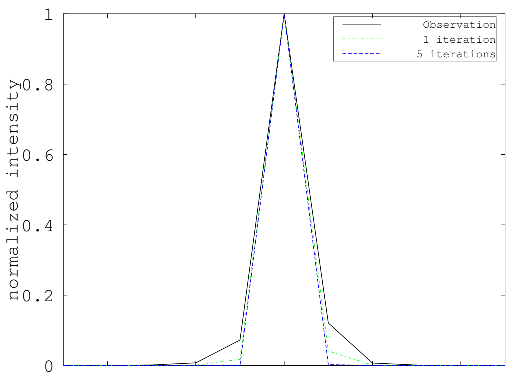
(f) Intensities along the column going through the maximum of the PSF.

Figure 6.9: Result of the PSF measured close to the upper left corner of the image when it is deconvolved using patch-invariant Gold's method, using 8 shift-invariant patches. Color axis: $[0, 0.2]$.

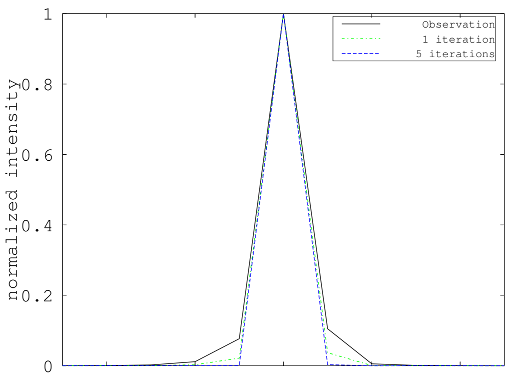

(a) Measured PSF.


(b) Deconvolved PSF, 1 iteration.


(c) Deconvolved PSF, 5 iterations.

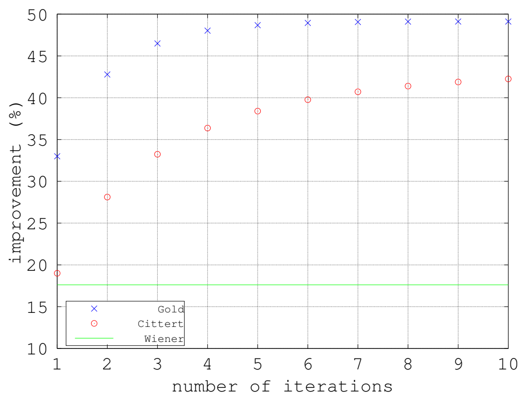


(d) Intensities along the row going through the maximum of the PSF.

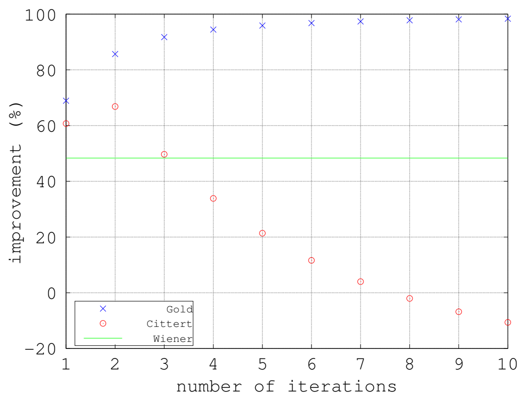


(e) Intensities along the column going through the maximum of the PSF.

Figure 6.10: Result of the PSF measured in the upper left corner of the image when it is deconvolved using fully shift-variant Gold's method. Color axis: $[0, 0.2]$.



(a) Deconvolution of the PSF when it is measured in the middle of the image.



(b) Deconvolution of the PSF when it is measured in the corner of the image.

Figure 6.11: Performance comparison (using (6.1)) between different shift-invariant deconvolution methods. Gold's method seems to yield the best performance for both the middle and the corner measurements of the point spread function.

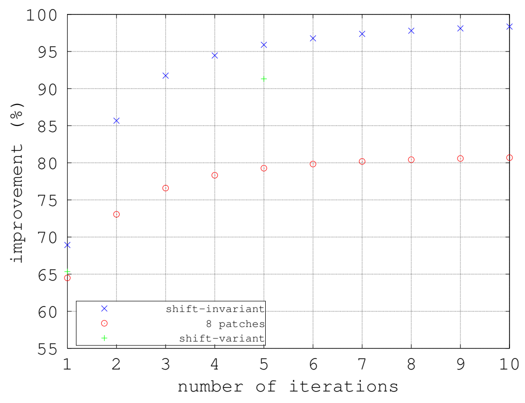


Figure 6.12: Performance comparison (using (6.1)) of different shift-variant deconvolution methods. The PSF is measured in the corner of the image. Shift-invariant deconvolution seems to yield the best performance.

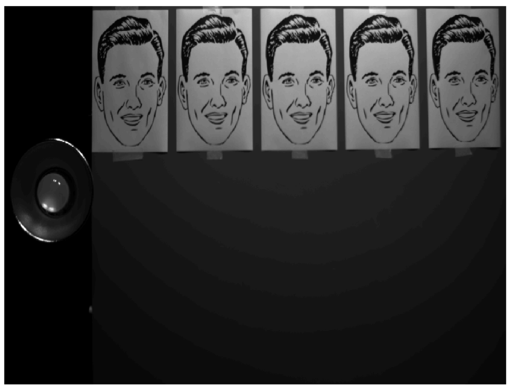
6.4 Objective experimental setup

In this section we present results of applying the stray light compensation framework (recall Figure 1.5) to images captured using a particular experimental setup. In Section 6.4.1, we show results for fully shift-variant, patch-invariant (with 8 shift-invariant patches; recall Chapter 5.4.1) and shift-invariant deconvolution. The faster deconvolution techniques (Chapters 5.4.2 - 5.4.4) are compared in Section 6.4.2. Due to the inefficiency of the shift-variant and patch-invariant deconvolution (see Table 6.1) they are used only rarely in the experiments. Instead, shift-invariant deconvolution using Gold's method is used unless otherwise stated.

To obtain a reasonably objective assessment of the stray light compensation performance, we construct the experiment shown in Figure 6.13. In Figure 6.13a we see the ground-truth, showing five cartoon faces positioned equidistantly apart. Figure 6.13b shows the same scene, but with a bright lamp turned on, causing a large increase of stray light in the image. This is the observed image. The observed image is compensated for stray light using the different compensation approaches. Next, each of the five faces are cut out in the ground-truth, the observed image and the restored images, respectively. Using these five cut-outs, we can compute the improvement (6.1) to obtain performance measurements at different distances from the light source.

6.4.1 Shift-variant, patch-invariant and shift-invariant deconvolution

Figures 6.14 - 6.15 shows results using shift-invariant deconvolution, patch-invariant deconvolution using 8 patches, and completely shift-variant deconvolution (Gold's method). For the shift- and patch-invariant deconvolution, we see that the improvement tends to


(a) Ground-truth. Color axis: $[0, 0.4]$.

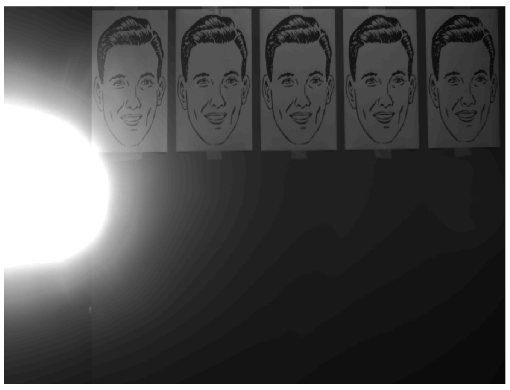

(b) Observed image. Color axis: $[0, 0.005]$.

Figure 6.13: The experiment setup for evaluation of stray light compensation. From left to right, the faces will be numbered 1 to 5.

increase as the number of iterations increases, and that it seems to converge after about 6 - 7 iterations. An exception is seen for the face closest to the lamp, in which the improvement decreases with the number of iterations. Moreover, we see that for all methods, the improvement is significantly weaker for this face, as compared to the rest (see specifically Figure 6.14f). Shift-invariant deconvolution seems to perform better compared to the patch-invariant and shift-variant deconvolution. Possible reasons for this are discussed in Chapter 7.1.2.

In Figures 6.16 - 6.17, Gold's method, Van Cittert's method and Wiener deconvolution are compared, all using shift-invariant deconvolution. The iterative methods seem to outperform Wiener deconvolution after a few iterations, except for the first face.

6.4.1.1 Noise sensitivity

Figures 6.18 - 6.19 show the stray light compensation performance when the observed image contains different amounts of noise, using shift-invariant deconvolution (Gold's method). The noise is mean-zero, additive and Gaussian. Different standard deviations are used for the noise. In particular, standard deviations 0.1%, 0.3% and 1% of the mean of the observed image are used.

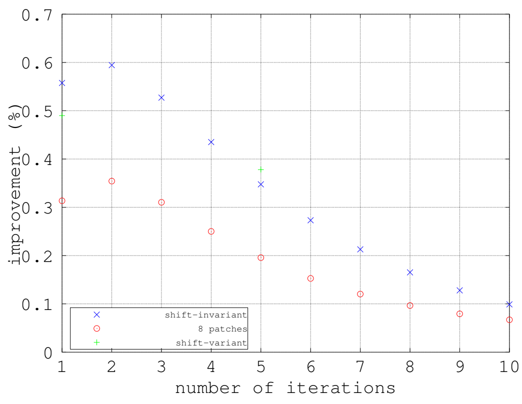
We see that the framework can handle smaller noise levels quite well, but starts to behave poorly with 1% noise. Specifically, in Figure 6.19 we see that the noise appears to be magnified by the iterative deconvolution.

6.4.1.2 Parameter robustness

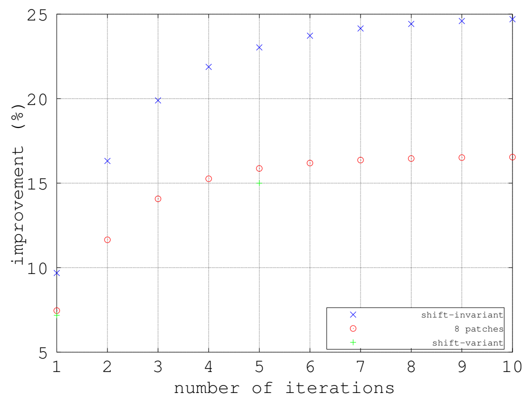
Figure 6.20 shows how the stray light compensation is affected when offsets are added to the estimated parameters. Specifically, 25%, 50% and 75% offset, respectively, are added (or subtracted) from the original parameter estimates. Table 6.4 shows the specific parameter setups evaluated.

Table 6.4: The parameter sets used in the parameter robustness evaluation.

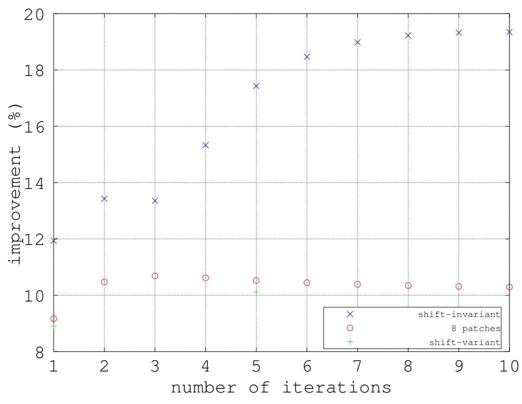
Amount off	β	σ	a	b	d	f	g
0%	0.826	1.063	-1.514×10^{-5}	-7.427×10^{-6}	1.175	2.127	0.758
25%	1.033	1.329	-1.136×10^{-5}	-5.571×10^{-6}	1.469	2.659	0.947
50%	1.240	1.595	-7.571×10^{-6}	-1.114×10^{-5}	0.588	1.063	1.137
75%	1.446	1.861	-2.650×10^{-5}	-1.857×10^{-6}	2.057	3.722	1.326



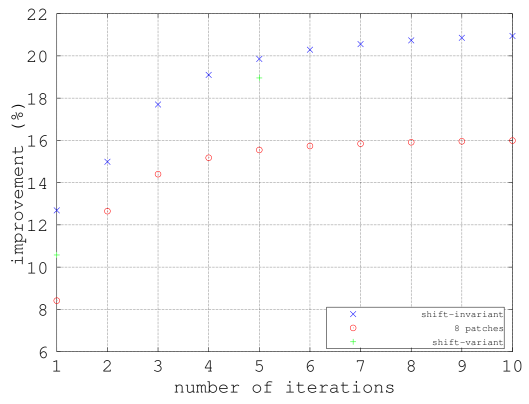
(a) Face 1.



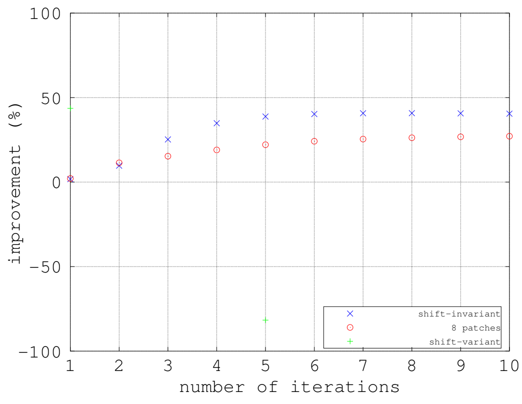
(b) Face 2.



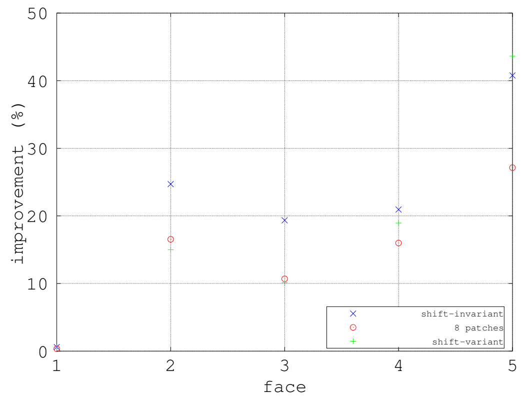
(c) Face 3.



(d) Face 4.



(e) Face 5.



(f) All five faces. For each face, the number of iterations yielding the most improvement in Figures 6.14a - 6.14e, respectively, is chosen.

Figure 6.14: Performance comparison between shift-invariant, patch-invariant (8 patches) and fully shift-variant deconvolution.

6.4.2 Fast, approximate deconvolution techniques

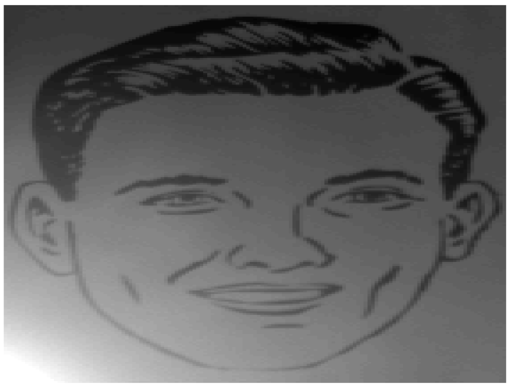
In this section we present results using the fast, approximate deconvolution techniques. For each method, shift-invariant deconvolution is used with Gold's method. Also, each approximate method is compared to normal shift-invariant deconvolution using Gold's method.

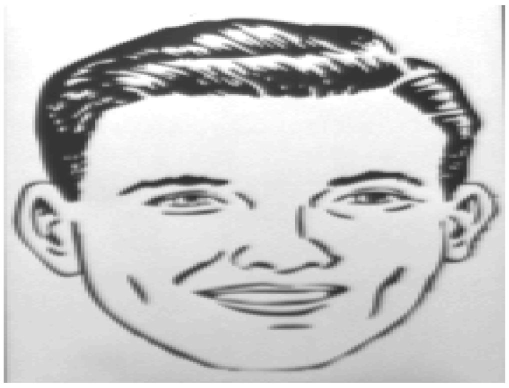
Figure 6.21 shows results using kernel thresholding (Chapter 5.4.2). We evaluate the performance for two different threshold levels. First, we keep the PSF within a box centered around the peak of the PSF, having 99.9% of all the intensity in the PSF within the box. Next, we lower the threshold to only require that 50% of all the intensity is confined within the box. We see that the general behavior remains the same when using the thresholds 99.9% and 50%, but the improvement decreases as the threshold decreases.

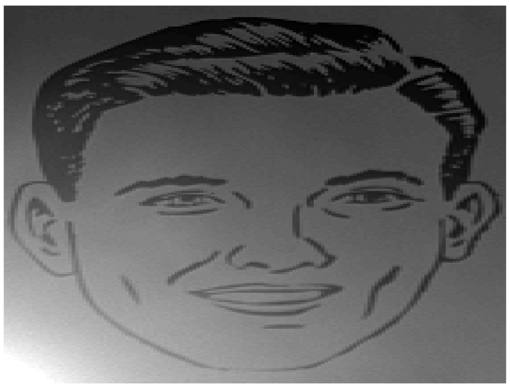
Figures 6.22 - 6.23 show results using image downsampling (Chapter 5.4.3). We evaluate three different levels of downsampling, 25%, 50% and 75%. We see that using any level of downsampling yields worse performance compared to no downsampling at all. However, the decrease in performance does not seem to be connected to the amount of downsampling in any straightforward way. For example, 75% downsampling yields better results than only 25% downsampling, as seen in Figure 6.22b. A possible explanation is given in Chapter 7.1.3.

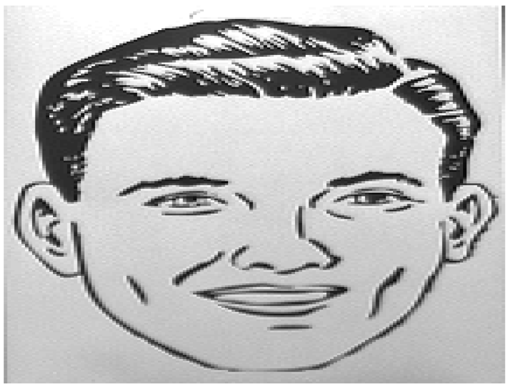
Figure 6.24 shows results using selected-ordinates sampling (Chapter 5.4.4). We evaluate selected-ordinates, both without any edge-clipping, and with edge-clipping (not including pixels which are more than 20% off compared to the pixel of evaluation). Without edge-clipping, the improvement becomes strongly negative after only a few iterations, but it seems to outperform the non-approximate deconvolution when using only one iteration. Including edge-clipping yields a behavior very similar to the non-approximate deconvolution. See Appendix B for a discussion on selected-ordinates sampling.

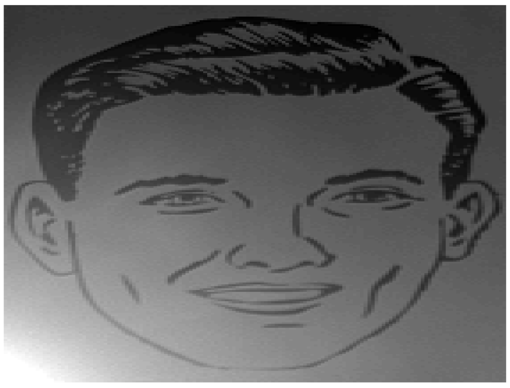
Finally, Figure 6.25 compares all three approximation techniques, and the non-approximate deconvolution, at the same time.

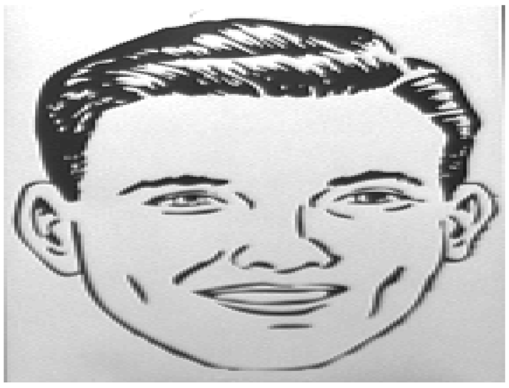

(a) Face 1, observation.

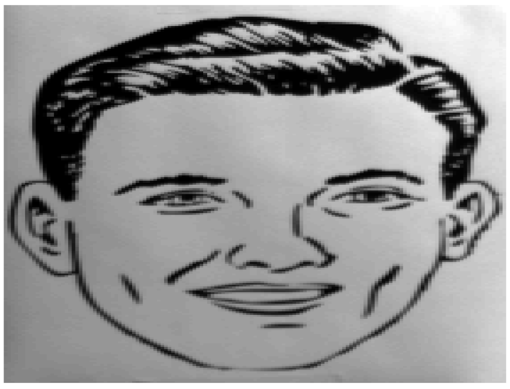

(b) Face 3, observation.


(c) Face 1, shift-invariant restoration.


(d) Face 3, shift-invariant restoration.


(e) Face 1, patch-invariant restoration.


(f) Face 3, patch-invariant restoration.


(g) Face 1, ground-truth.

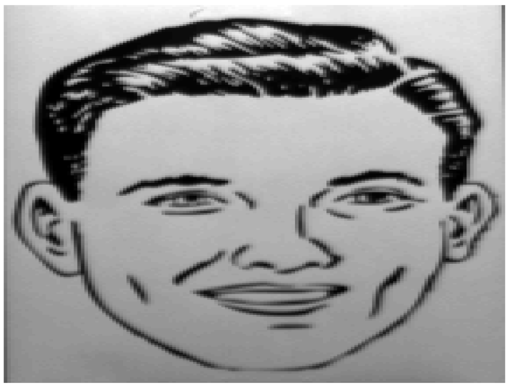
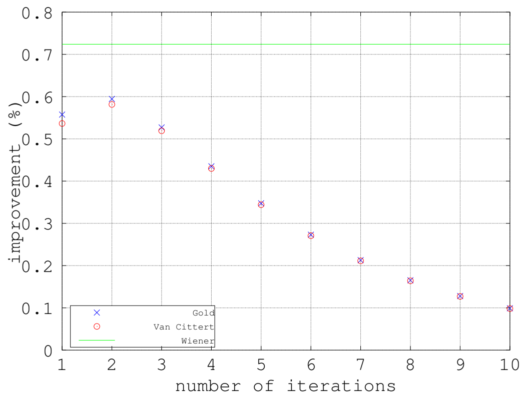
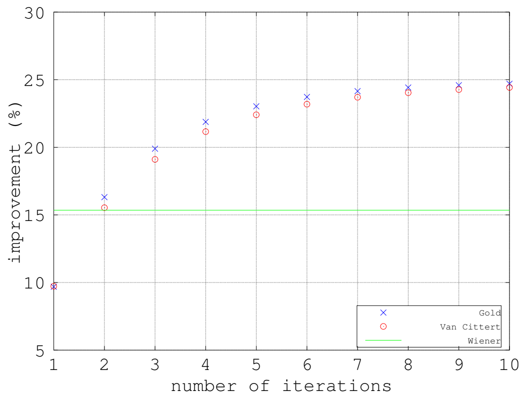

(h) Face 3, ground-truth.

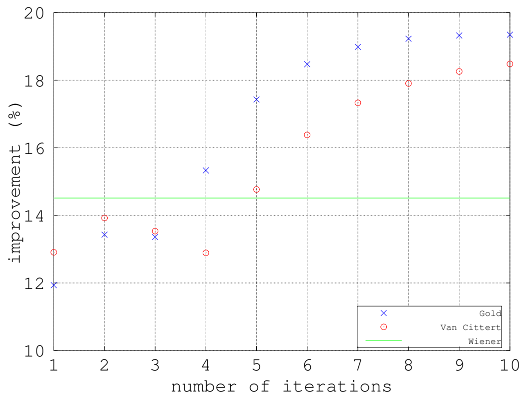
Figure 6.15: Comparison between shift-invariant and patch-invariant deconvolution using Gold's method. For each restoration, the number of iterations yielding the best result is used. Color axis: $[0.1, 0.85]$.



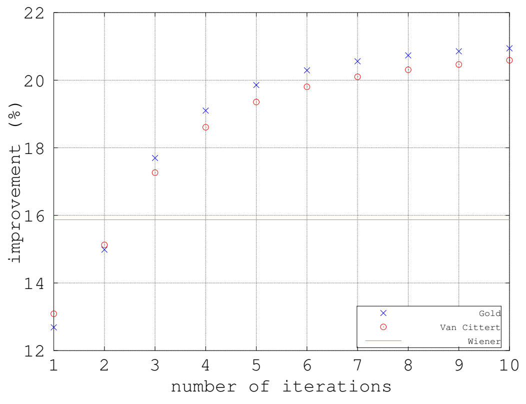
(a) Face 1.



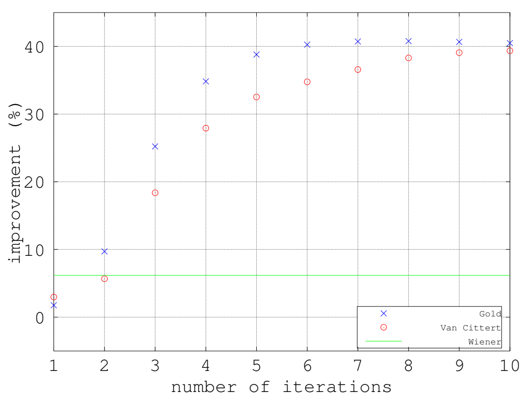
(b) Face 2.



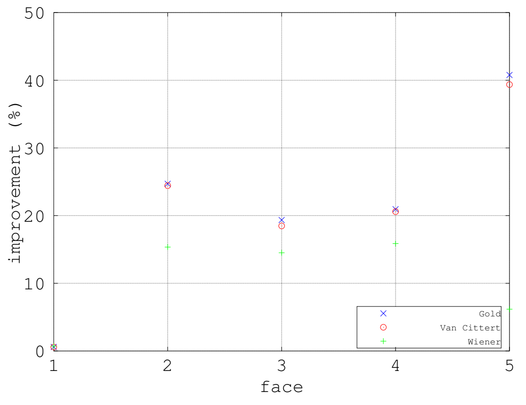
(c) Face 3.



(d) Face 4.

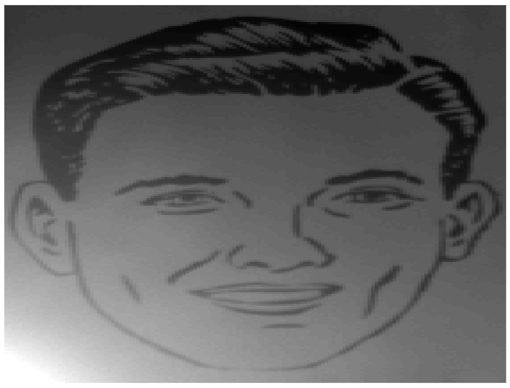


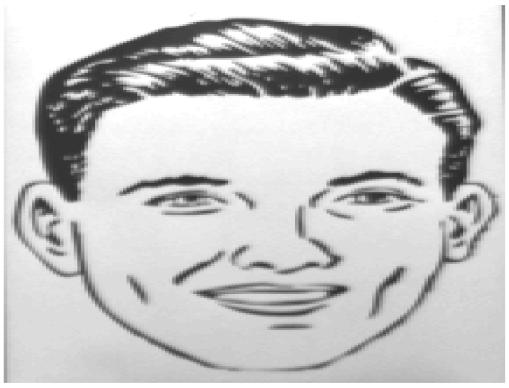
(e) Face 5.

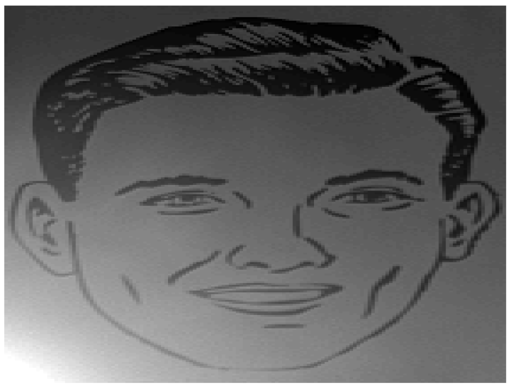


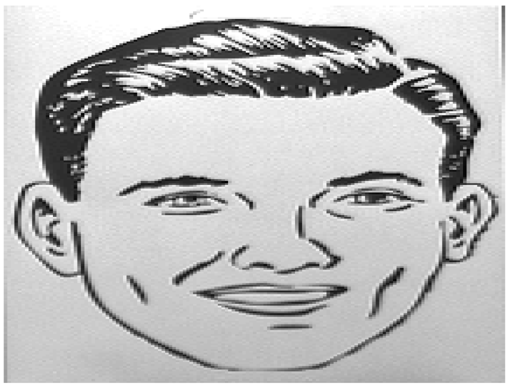
(f) All five faces. For each face, the number of iterations yielding the most improvement in Figures 6.16a - 6.16e, respectively, is chosen.

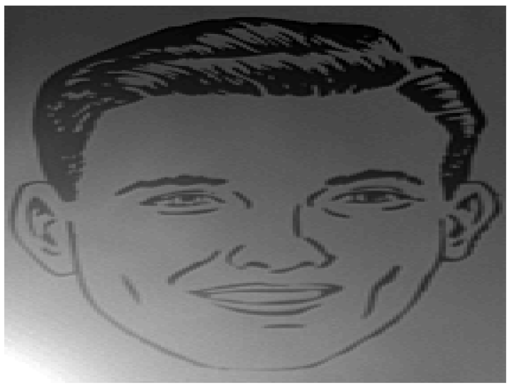
Figure 6.16: Performance comparison between shift-invariant Gold, Van Cittert and Wiener deconvolution.

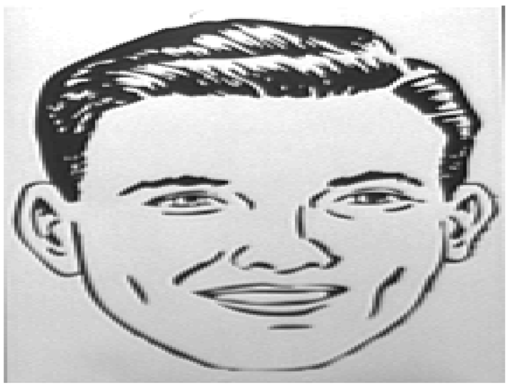

(a) Face 1, observation.

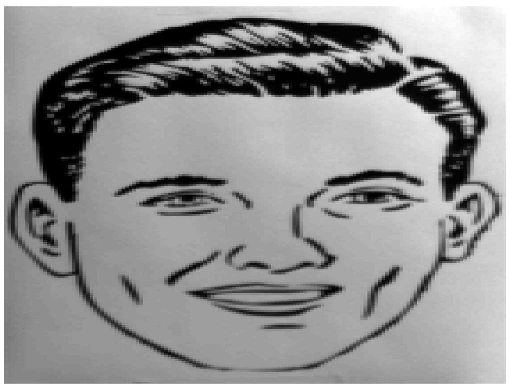

(b) Face 3, observation.


(c) Face 1, Van Cittert (shift-invariant).


(d) Face 3, Van Cittert (shift-invariant).


(e) Face 1, Wiener deconvolution.


(f) Face 3, Wiener deconvolution.


(g) Face 1, ground-truth.

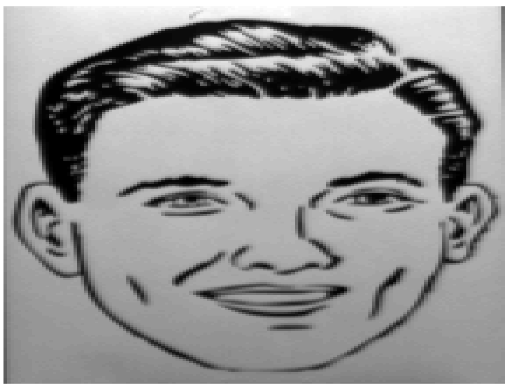
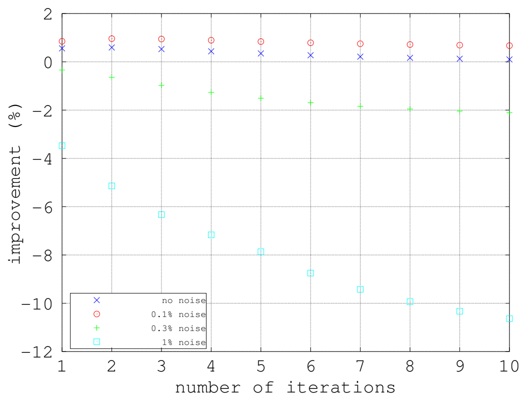
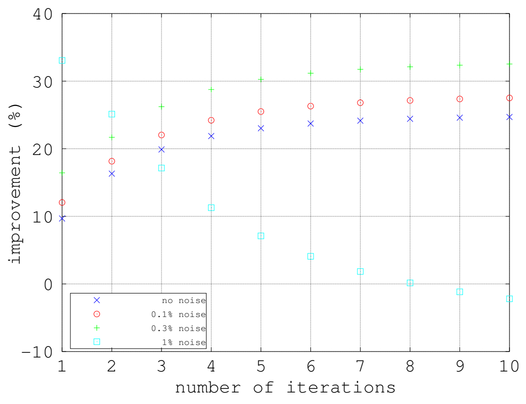

(h) Face 3, ground-truth.

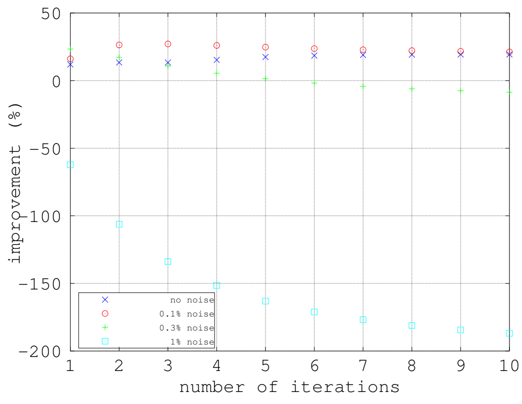
Figure 6.17: Comparison between shift-invariant deconvolution using Van Cittert's method and shift-invariant Wiener deconvolution. For each restoration using Van Cittert's method, the number of iterations yielding the best result is used. Color axis: $[0.1, 0.85]$.



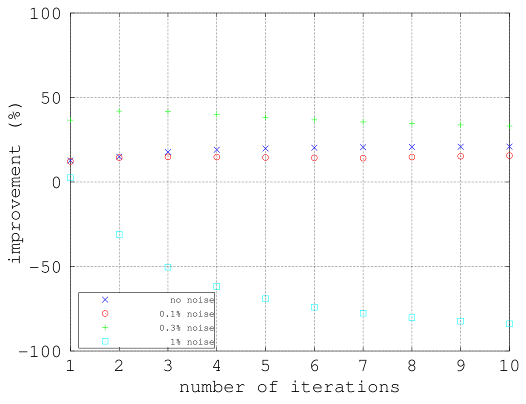
(a) Face 1.



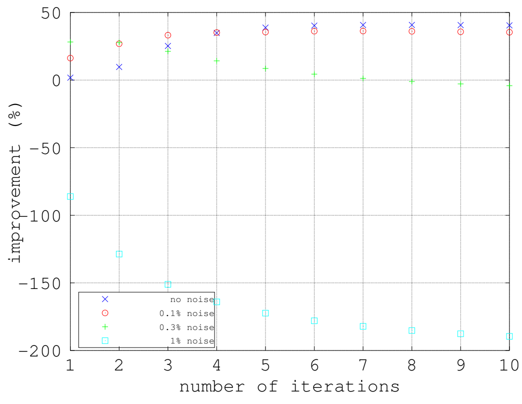
(b) Face 2.



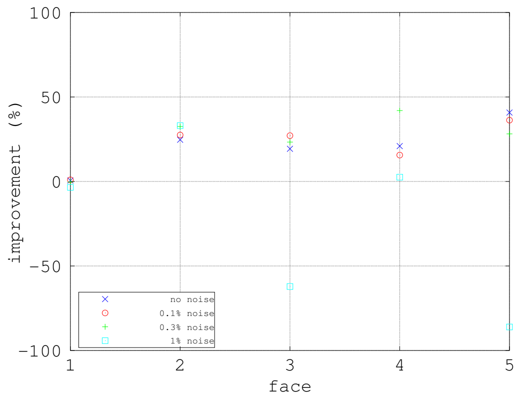
(c) Face 3.



(d) Face 4.

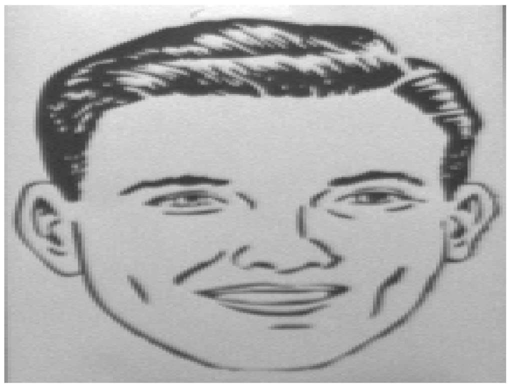


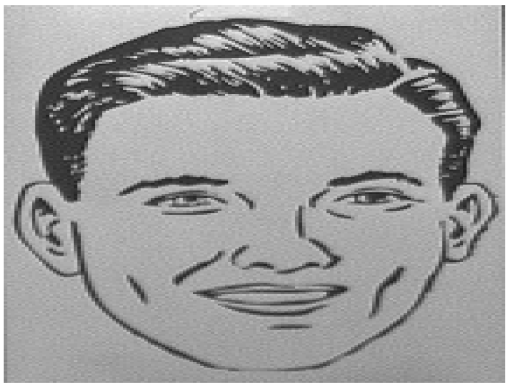
(e) Face 5.

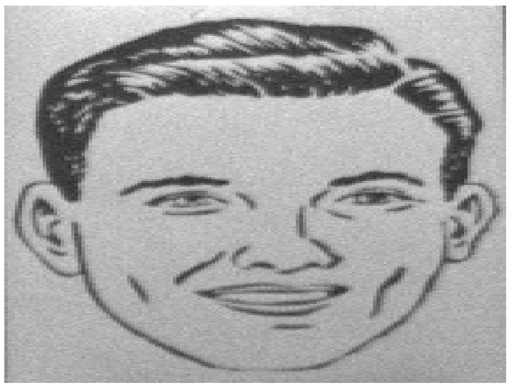


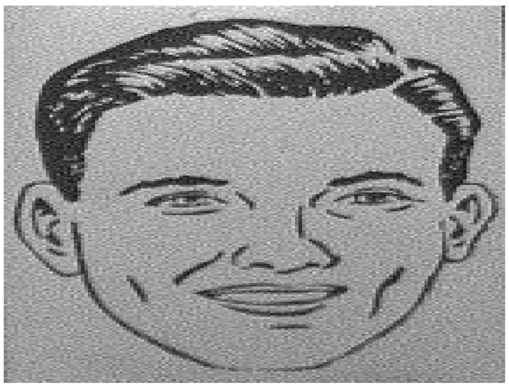
(f) All five faces. For each face, the number of iterations yielding the most improvement in Figures 6.18a - 6.18e, respectively, is chosen.

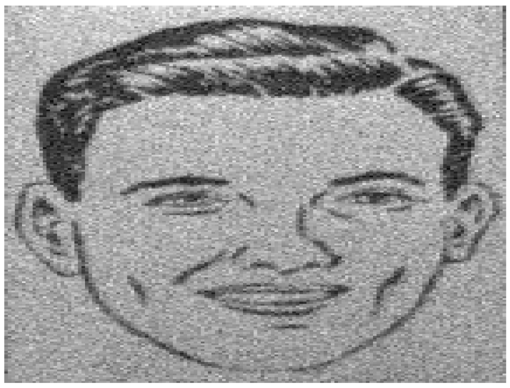
Figure 6.18: Results using different noise levels.

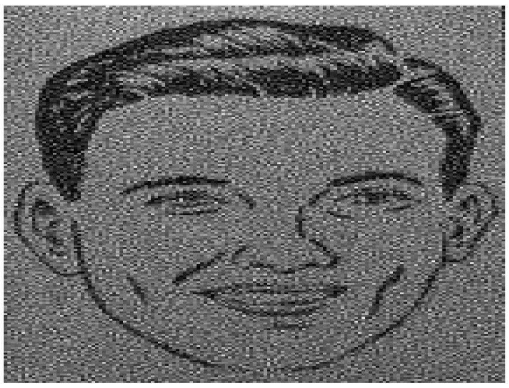

(a) Face 3, observation, 0.1% noise.


(b) Face 3, restored, 0.1% noise.


(c) Face 3, observation, 0.3% noise.


(d) Face 3, restored, 0.3% noise.


(e) Face 3, observation, 1% noise.


(f) Face 3, restored, 1% noise.

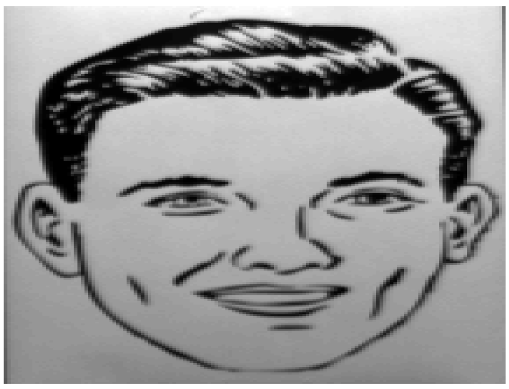
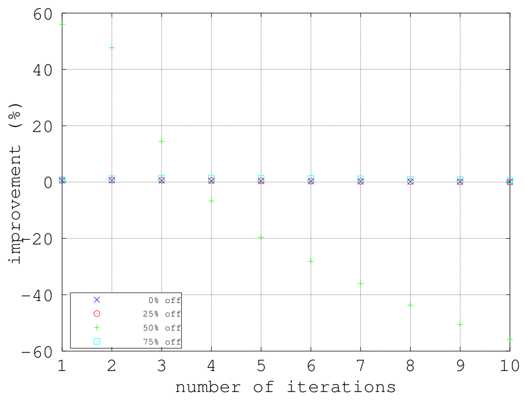
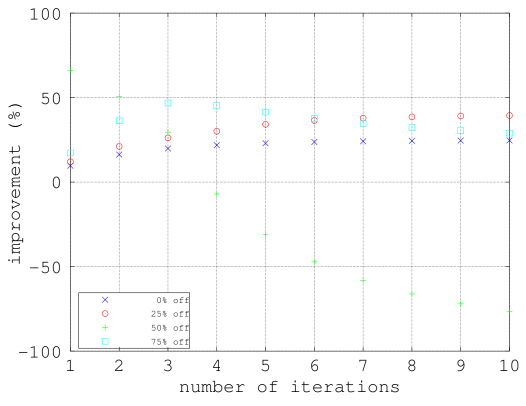

(g) Face 3, ground-truth.

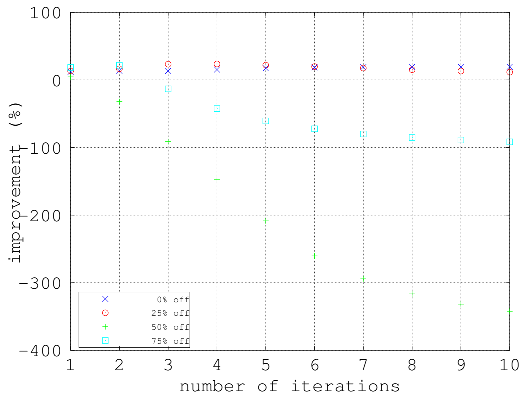
Figure 6.19: The effects of noise in the observation. Using shift-invariant deconvolution with Gold's method (5 iterations), the overall contrast seems to improve, but the noise seems to magnify. Color axis: $[0.1, 0.85]$.



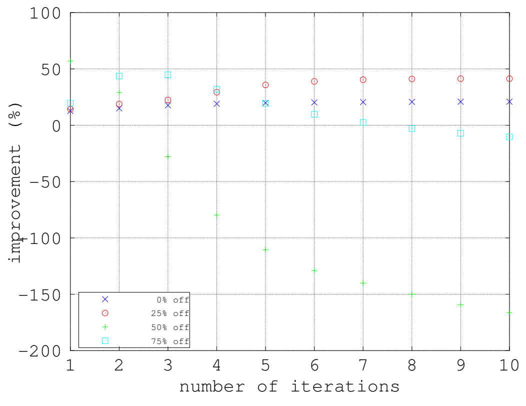
(a) Face 1.



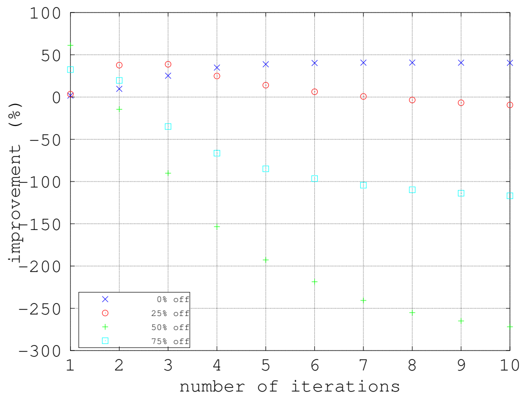
(b) Face 2.



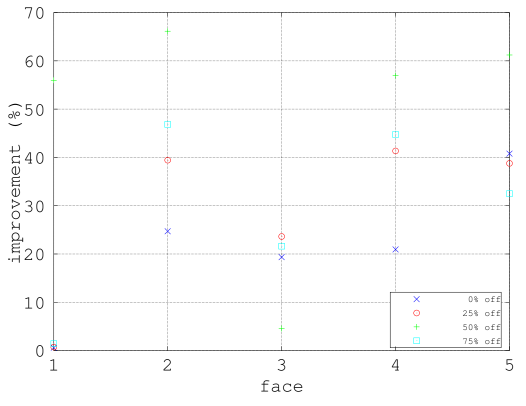
(c) Face 3.



(d) Face 4.

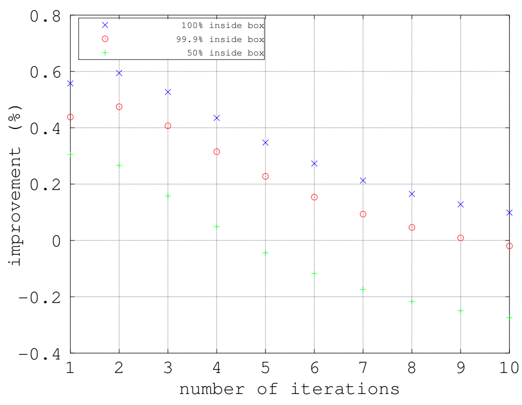


(e) Face 5.

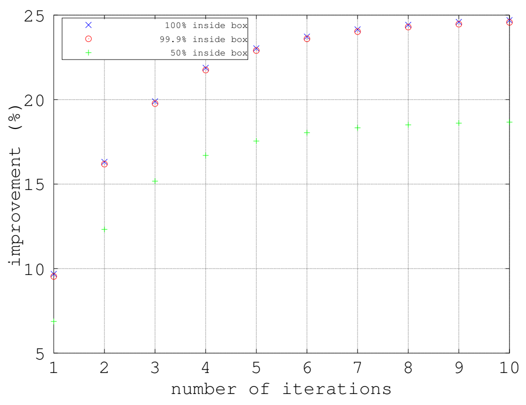


(f) All five faces. For each face, the number of iterations yielding the most improvement in Figures 6.20a - 6.20e, respectively, is chosen.

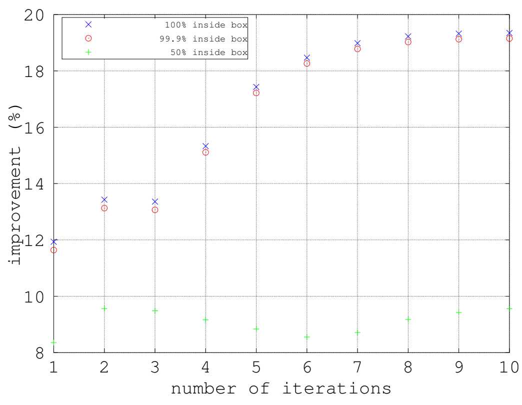
Figure 6.20: Results on parameter robustness. Adding parameter offset does not necessarily imply decreased performance. This is discussed in Chapter 7.1.2.



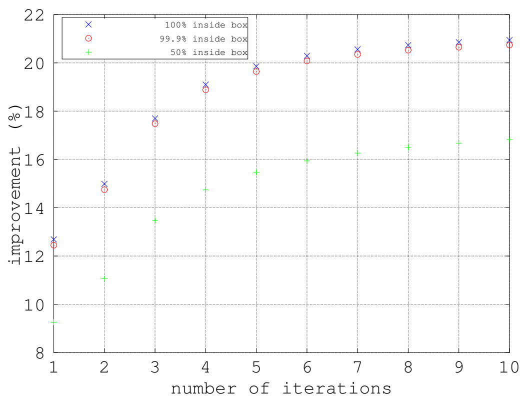
(a) Face 1.



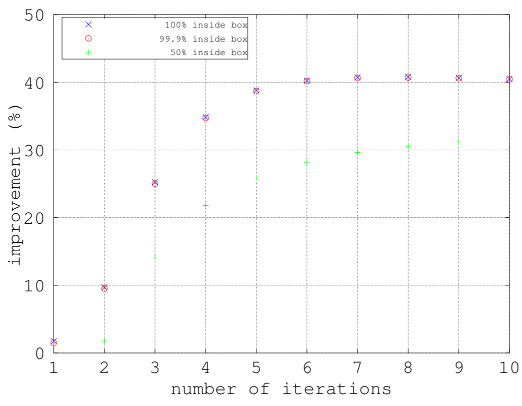
(b) Face 2.



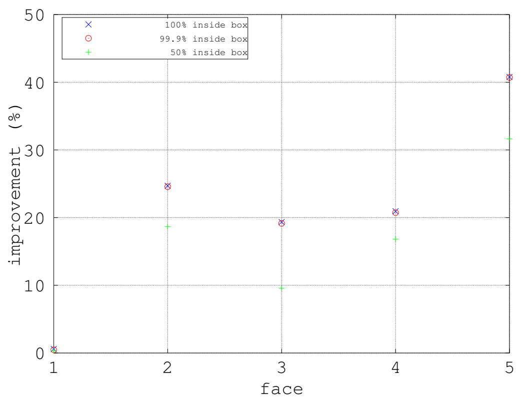
(c) Face 3.



(d) Face 4.

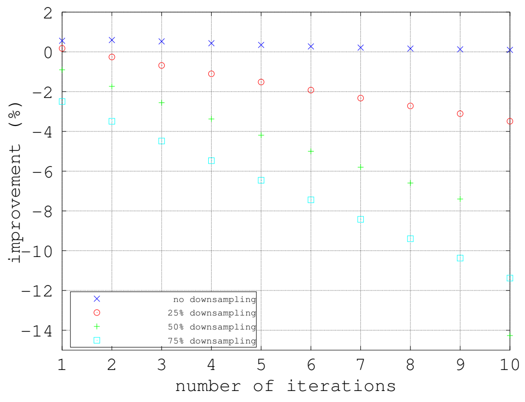


(e) Face 5.

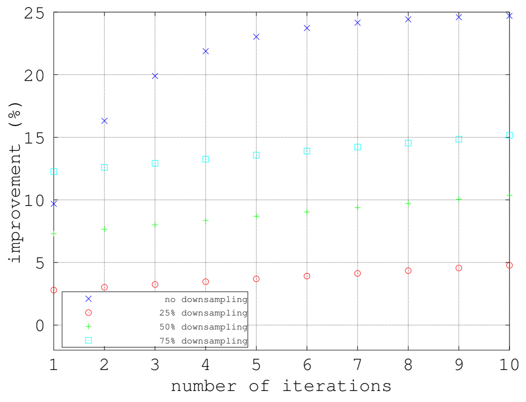


(f) All five faces. For each face, the number of iterations yielding the most improvement in Figures 6.21a - 6.21e, respectively, is chosen.

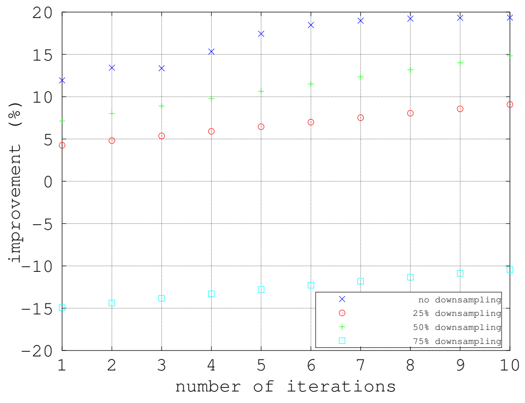
Figure 6.21: Results on kernel thresholding. The 100% inside box label refers to no kernel thresholding, i.e., the normal shift-invariant deconvolution.



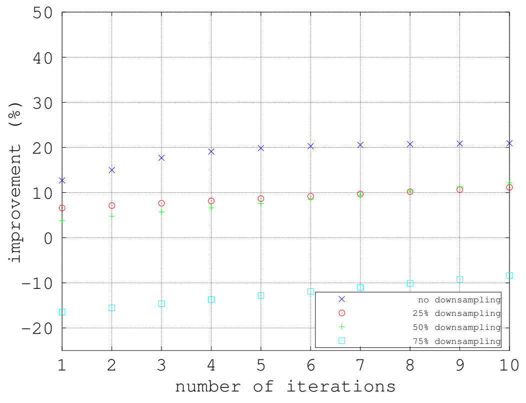
(a) Face 1.



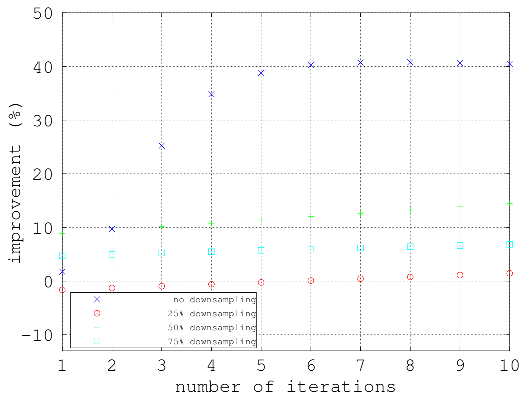
(b) Face 2.



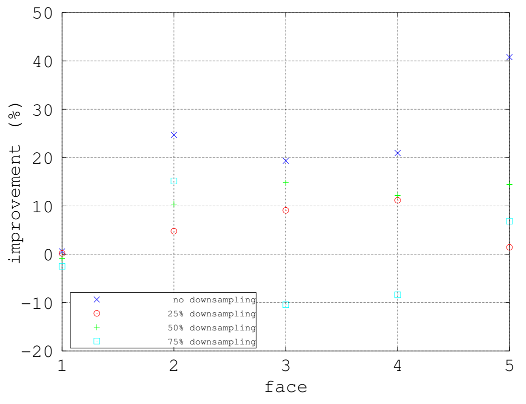
(c) Face 3.



(d) Face 4.

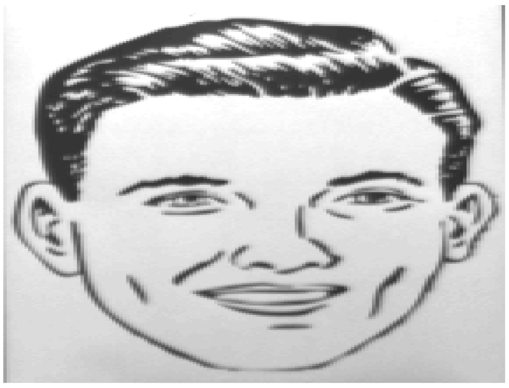


(e) Face 5.

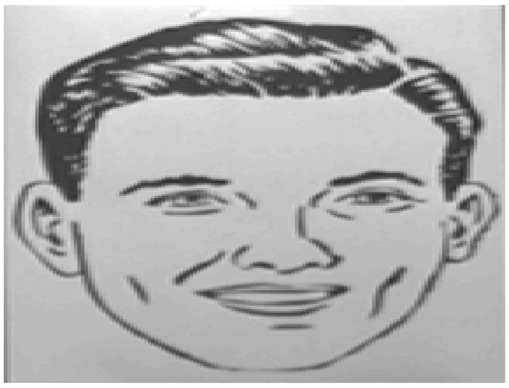


(f) All five faces. For each face, the number of iterations yielding the most improvement in Figures 6.22a - 6.22e, respectively, is chosen.

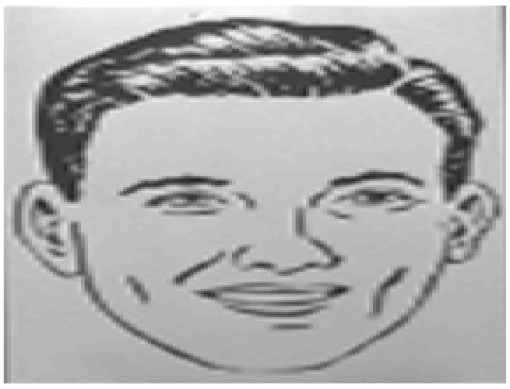
Figure 6.22: Comparison between different image downsampling levels. In each case, shift-invariant deconvolution is used.



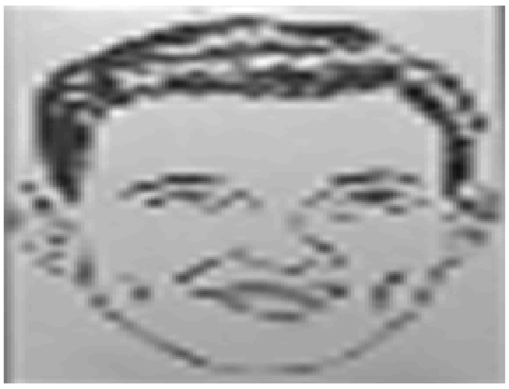
(a) Face 3, observation.



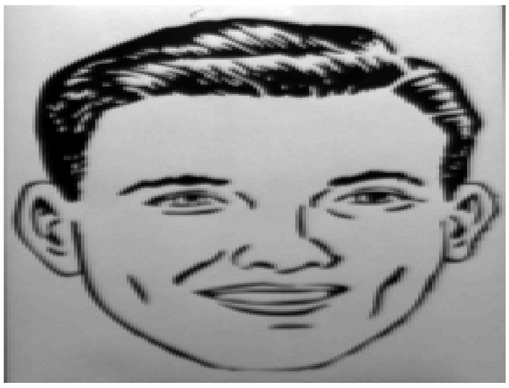
(b) Face 3, restored, 25% downsampling.



(c) Face 3, restored, 50% downsampling.

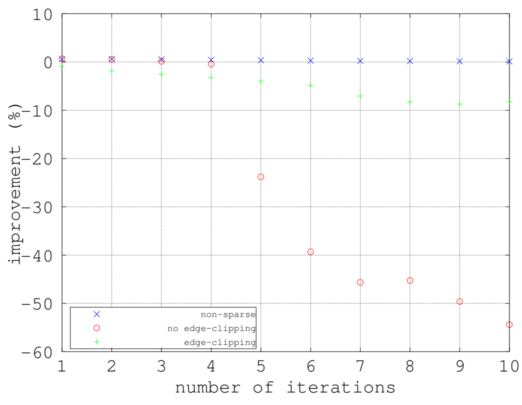


(d) Face 3, restored, 75% downsampling.

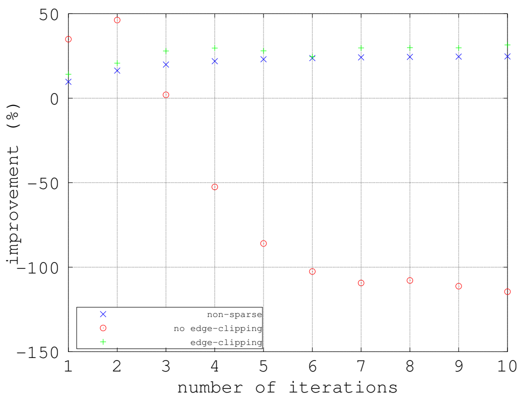


(e) Face 3, ground-truth.

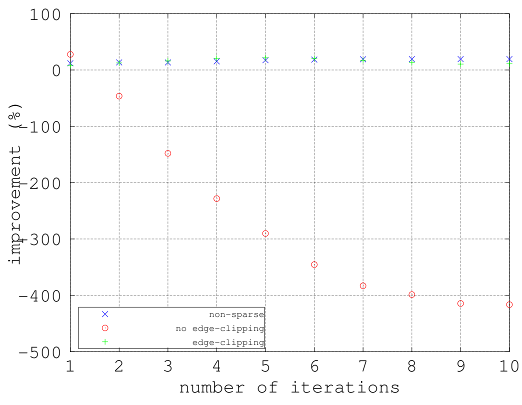
Figure 6.23: The effects of downsampling the input image and then using shift-invariant deconvolution with Gold's method (5 iterations). Color axis: $[0.1, 0.85]$.



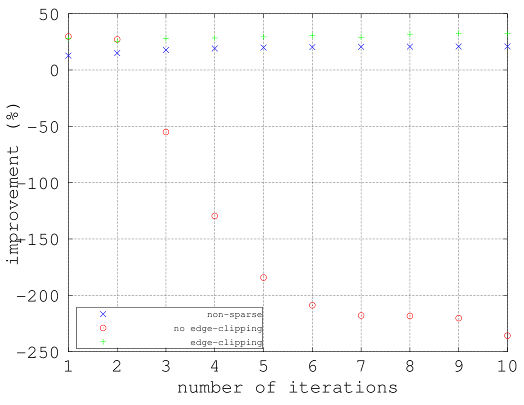
(a) Face 1.



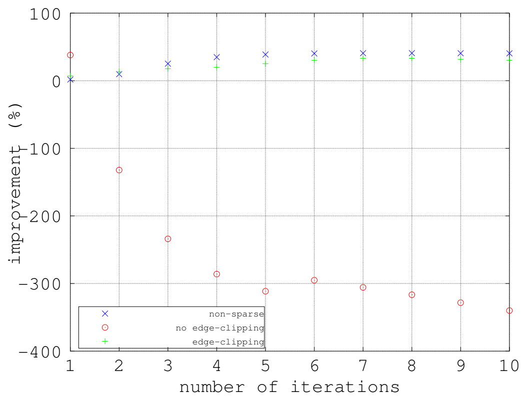
(b) Face 2.



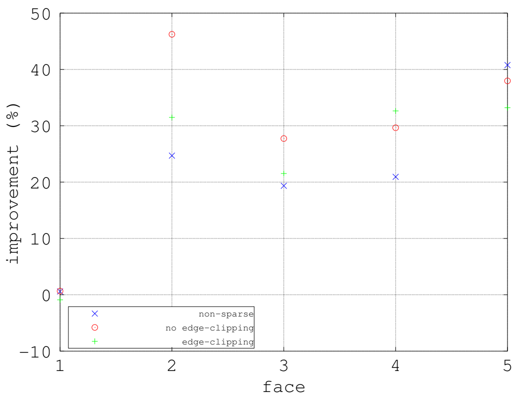
(c) Face 3.



(d) Face 4.

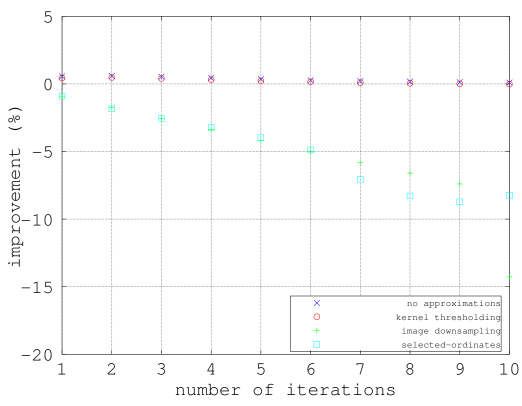


(e) Face 5.

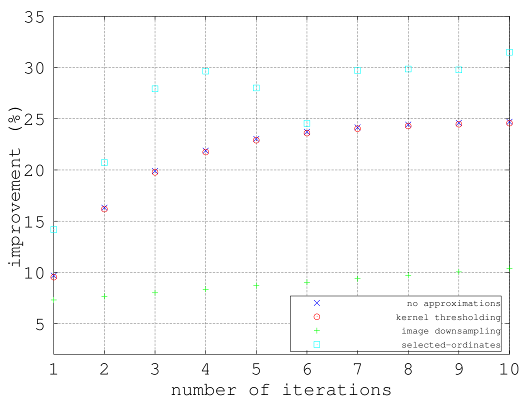


(f) All five faces. For each face, the number of iterations yielding the most improvement in Figures 6.24a - 6.24e, respectively, is chosen.

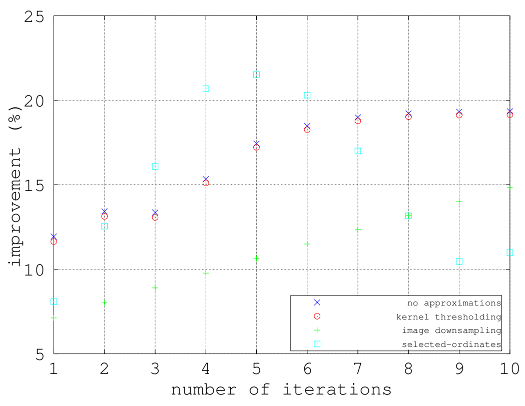
Figure 6.24: Results using selected-ordinates sampling. Non-sparse refers to the ordinarily sampled image, i.e., the non-approximate deconvolution. In each case, shift-invariant deconvolution is used.



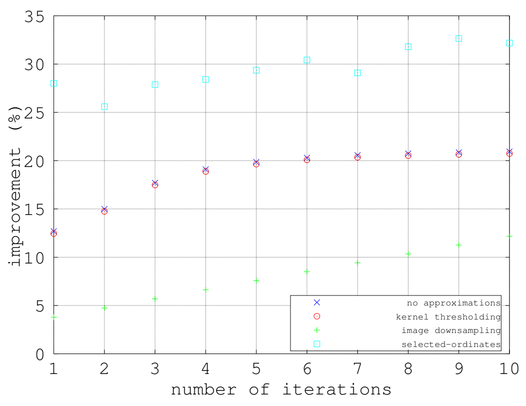
(a) Face 1.



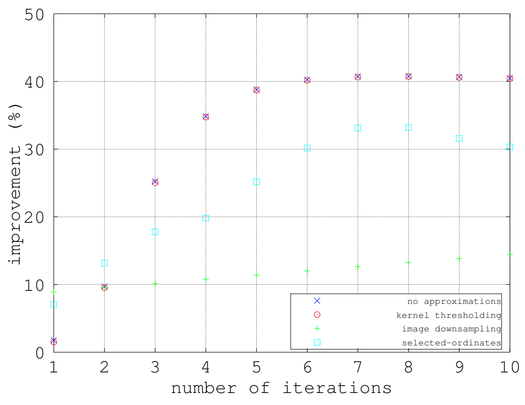
(b) Face 2.



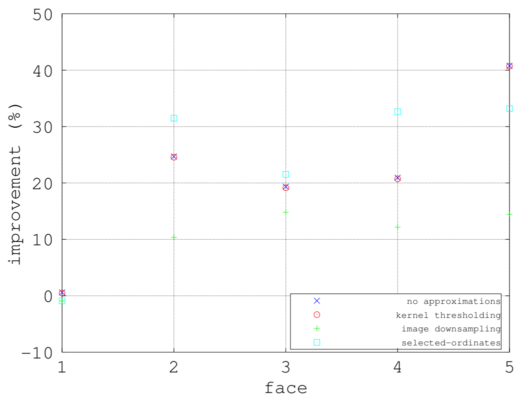
(c) Face 3.



(d) Face 4.



(e) Face 5.

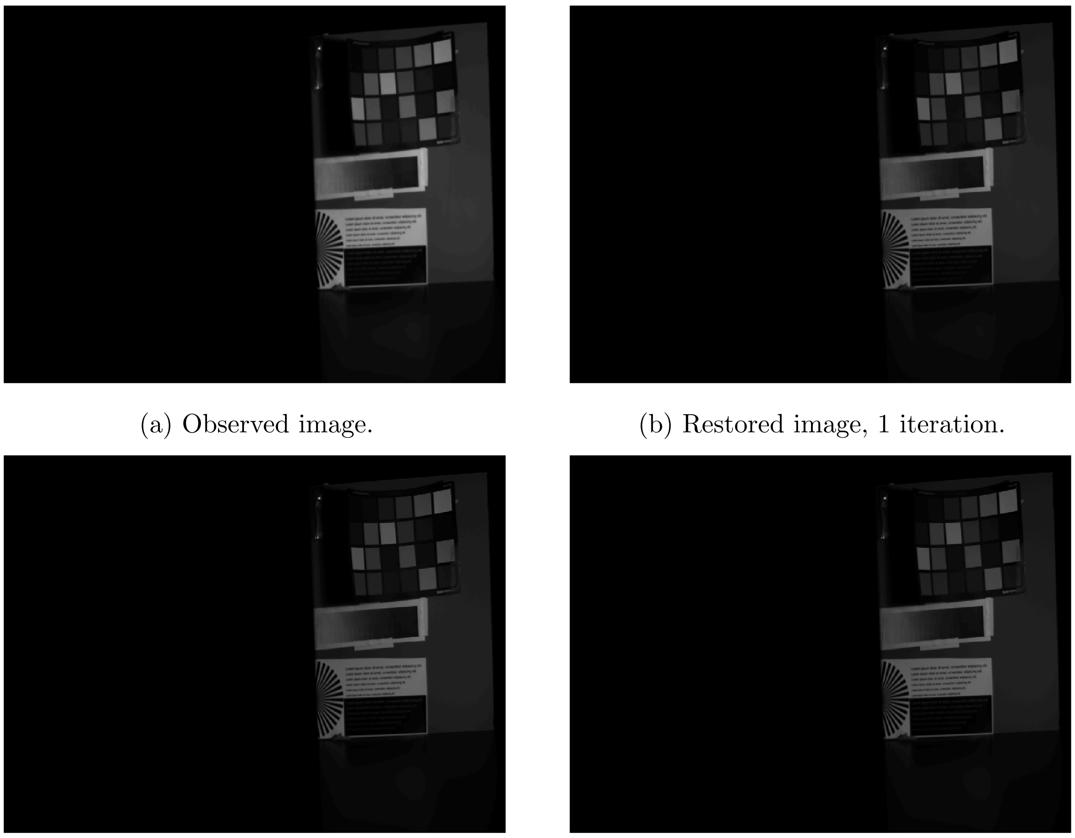


(f) All five faces. For each face, the number of iterations yielding the most improvement in Figures 6.25a - 6.25e, respectively, is chosen.

Figure 6.25: Comparison of all three approximation techniques, together with the non-approximate deconvolution. For kernel thresholding, 99.9% inside the box thresholding is used; for image downsampling, 50% downsampling is used; and for selected-ordinates, edge-clipping is used, not using pixels which deviate with more than 20% from the current pixel. In each case, shift-invariant deconvolution is used.

6.5 A few additional results

In this section we show a few more results; all results are obtained using shift-invariant deconvolution using Gold's method. Figures 6.26 - 6.27 should be studied jointly; the same goes for Figures 6.28 - 6.29.



(a) Observed image.

(b) Restored image, 1 iteration.

(c) Restored image, 5 iterations.

(d) Restored image, 10 iterations.

Figure 6.26: Not much can be concluded from this Figure, but in Figure 6.27, the dark area is zoomed in, which is more interesting.

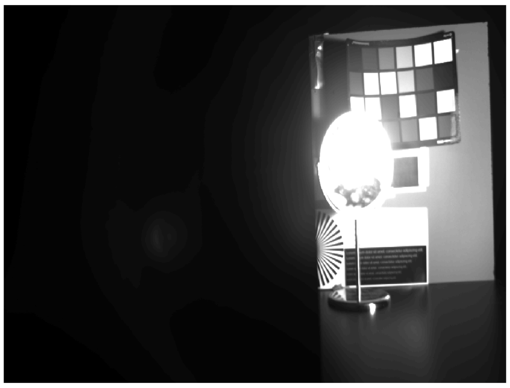
(a) Observed image.

(b) Restored image, 1 iteration.

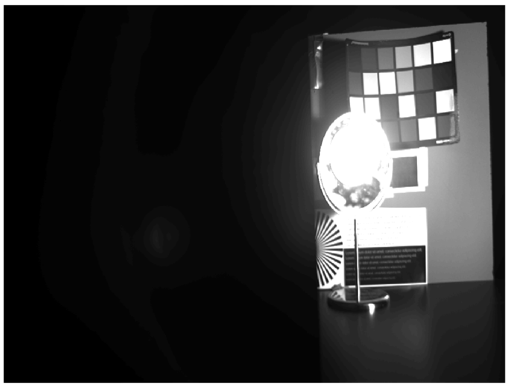
(c) Restored image, 5 iterations.

(d) Restored image, 10 iterations.

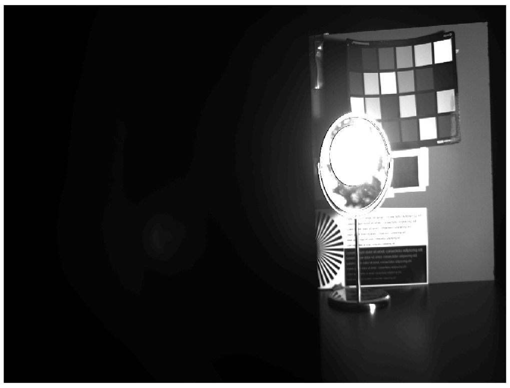
Figure 6.27: Zoomed in version of Figure 6.26. The monkey and other details become increasingly apparent as the number of iterations increases, and the level of detail seems to increase (see for example the black-white frame).



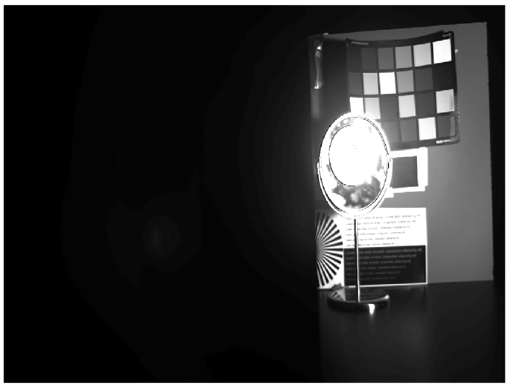
(a) Observed image.



(b) Restored image, 1 iteration.

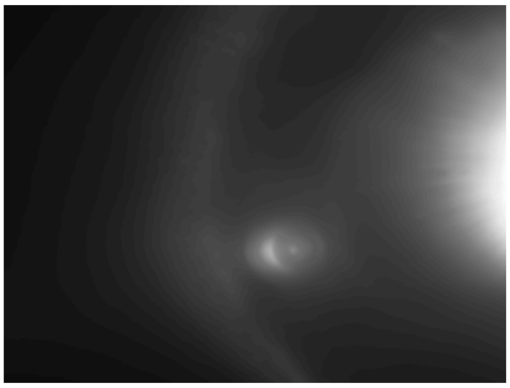


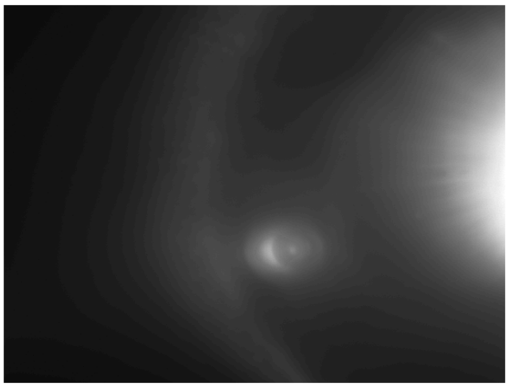
(c) Restored image, 5 iterations.

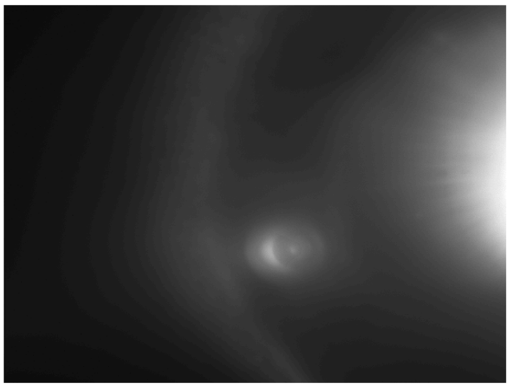


(d) Restored image, 10 iterations.

Figure 6.28: Note how the glow from the reflecting mirror becomes smaller as the number of iterations increases. Also, the text behind the mirror becomes apparent after about five iterations. In Figure 6.29, the dark area is zoomed in. Color axis: $[0, 0.008]$.


(a) Observed image.


(b) Restored image, 1 iteration.


(c) Restored image, 5 iterations.

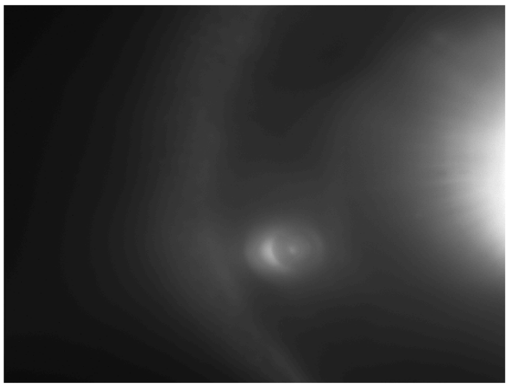

(d) Restored image, 10 iterations.

Figure 6.29: Zoomed in version of Figure 6.28. Nothing becomes visible, even after 10 iterations. This is likely because the light from the mirror is too strong.

Chapter 7

Discussion

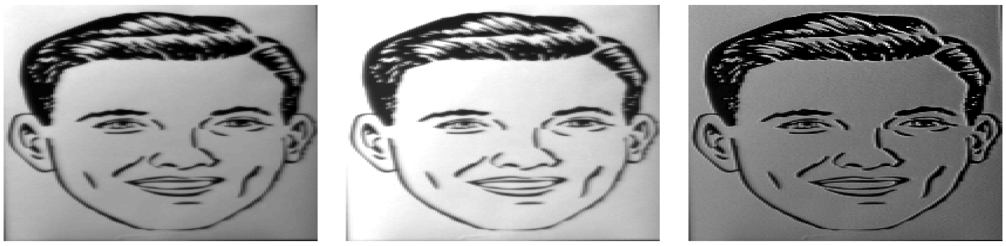
In this concluding chapter we discuss the work done in previous chapters. In Section 7.1, emphasis is put on analyzing the results from Chapter 6. The results are also compared to the preceding theoretical chapters. In Section 7.2, we try to answer the questions in Chapter 1.1. For convenience, the questions will be re-stated in this section. Finally, Section 7.3 contains suggestions on future work.

7.1 Analysis of results

7.1.1 Evaluation of the experimental setup

Finding plausible solutions to a problem is sometimes a lot easier than validating the solutions themselves. Image processing is haunted by this issue, and finding meaningful objective performance measures is difficult. We believe that the method introduced in Chapter 6.4 can be considered both meaningful and reasonably objective. It should be pointed out though, that the ground-truth used in the experiment also contains stray light, since it has been captured by a (non-ideal) camera. However, this stray light should hopefully be negligible compared to that caused by the bright lamp.

A drawback of the experimental setup is that we did not take any measures to reduce indirect light, caused by reflections off the walls, from entering the camera. To deal with this, we could have chosen a larger room and/or coated the walls with anti-reflective materials. Another drawback is seen in Figure 6.13a, where we clearly see that the scene lightning is brighter in the center. Ideally, this light should be evenly distributed over all the faces.



(a) Face 5, ground-truth. (b) Face 5, observation. (c) Face 5, shift-variant restoration (5 iterations).

Figure 7.1: Even though the improvement is negative according to (6.1) as seen in Figure 6.14e, the result does not look bad.

The setup also allows for evaluating how the stray light compensation performs at different distances from the light source. Studying Figures 6.14 - 6.25, we find that the distance only seems to matter if the object is very close to the light source. An intuitive explanation to this behavior might be that the close vicinity of the light source is naturally more corrupt, making it difficult to restore parts of the image with the proposed methods.

Sifting through the results we find an anomaly in Figure 6.14e. In particular, one may wonder why the fully shift-variant method yields a negative improvements for the fifth face after five iterations. To answer this we produce Figure 7.1, where we can see that the result is clearly not bad; the face appears more clearly than in the observed image. Hence, we draw the conclusion that (6.1) might be an insufficient measure of improvement. To improve it, one should look at alternatives to the Frobenius norm for measuring the error between images.

Important to note is that the error we use (the Frobenius norm) will give different results depending on how one normalizes the image. Therefore, the improvement measure should be used with caution.

7.1.2 Shift-variance compared to shift-invariance

Studying Figures 6.11 and 6.12, we see that shift-invariant (both iterative and Wiener), patch-invariant and fully shift-variant deconvolution all seem to successfully reduce stray light, at least in the PSF-measurements themselves. It is worth repeating that iterative methods do not always guarantee convergence, as can be seen in 6.11b, but they often obtain better performance compared to Wiener deconvolution. Note that iterative methods can still be employed, but one should either try to prove convergence or monitor the results from each iteration so that one can abort the process when maximum improvement seems to have been reached.

Furthermore, comparing results in Figures 6.8 - 6.10, Figure 6.12 and Figure 6.14 shows that shift-invariant deconvolution seems to outperform patch-invariant and shift-variant deconvolution. We believe this can be explained by the following two reasons, either separately or in combination. First, we might have poorly estimated the PSF-parameters. This could be because we have used too few measurements, that only using diagonal measurements is not enough, or that the optimization routine performs suboptimally. The reason why the optimization routine might work somewhat poorly is the extremely small values of the PSF. From the peak of the PSF, the decay is very fast, making most values in the goal function (4.2) insignificant. We tried circumventing this issue by altering the weight function in (4.2) to give more weight to small PSF-values, but without any success. The only thing that gave any effect was to multiply the entire PSF by a large constant; however, the tails of the PSF still remain small relative to the peak. The belief that the parameters are somewhat off is supported by the results in Figure 6.20, which shows that in some cases, a major offset in the parameters yields even better results than the original parameters. Second, it could be that the parametric model itself is not suitable for describing stray light, but we should be very careful in making such a conclusion since we do not have enough experience with the model.

To further investigate the problems related to the parametric model (or the parameter estimates), look at Figure 7.2, which is similar to Figure 6.3, but in full size and using different settings for the color axis. From these images, we clearly see that the estimated point spread functions are off compared to the PSF-measurements. In particular, the PSF-estimates seem to be much more concentrated than the PSF-measurements. One could argue that the values in the tails have a negligible effect on the end result. However, it is not the PSF-values themselves that matter. Looking at the sum (2.2), we see that it is the sum of the products of the PSF-values and the underlying image values that forms the result. Moreover, a large sum of small terms may become significant. Therefore, the estimated PSF can indeed be questioned.

We end this section by noting that patch-invariant deconvolution should perform better when using smaller patches, at least when using a correctly estimated PSF. We chose to use eight patches due to time limitations.

7.1.3 Fast, approximate methods

As for the speed-up approximations, we begin by stating that, intuitively, the results of any approximation should be close to or worse than the method being approximated. Now, it may be difficult to evaluate what is close, but one should pay particular attention to when the approximation performs better than the original method. In such a scenario, the approximation might rather be considered a method of its own, or a fortunate guess. Studying the results in Figures 6.21 - 6.24, we see that kernel thresholding and image downsampling seem to be proper approximations in that they perform worse than their original method, whereas selected-ordinates sampling (without edge-clipping) sometimes yields better performance compared to the original method.

Kernel thresholding seems to perform as expected, with decreasing performance as the approximation becomes coarser. Note in Figure 6.21 that kernel thresholding (99.9%) performs almost as well as when not using any thresholding, but it runs about 35 times faster (see Table 6.1). Downsampling does not seem to have such a behavior. Each level of downsampling performs worse than no downsampling at all, but comparisons within the downsampled images show that the most downsampled image does not always yield the worst performance. This is somewhat surprising, especially when considering Figure 6.23, where clearly the image quality reduces as the downsampling level increases. This reinforces the suspicion that the measure of improvement (6.1) is not sufficient.

We note that selected-ordinates sampling (without edge-clipping) exhibits a very aggressive behavior. Using very few iterations, it yields significant improvement, before wandering away in the completely wrong direction. One could conclude that selected-ordinates sampling significantly outperforms the non-approximate method when using only a few iterations. Therefore, it is questionable if it can even be called an approximation, as discussed earlier. Furthermore, it is difficult to rule out that its impressive performance is attributed to luck. In Appendix B we discuss the validity of selected-ordinates sampling. A question which arises regarding this method is, if we assume that the non-approximate method converges to some solution, does it converge to the same solution that the selected-ordinates method reaches in its early iterations? Unfortunately, we cannot answer this question, since we have not run enough iterations.

We end this section by pointing out that the behavior of the speed-up approximations could also be connected to our previous insight regarding the validity of the parameters and the parametric model. It may very well be that some approximate methods perform better if the parameters of the non-approximate methods are off.

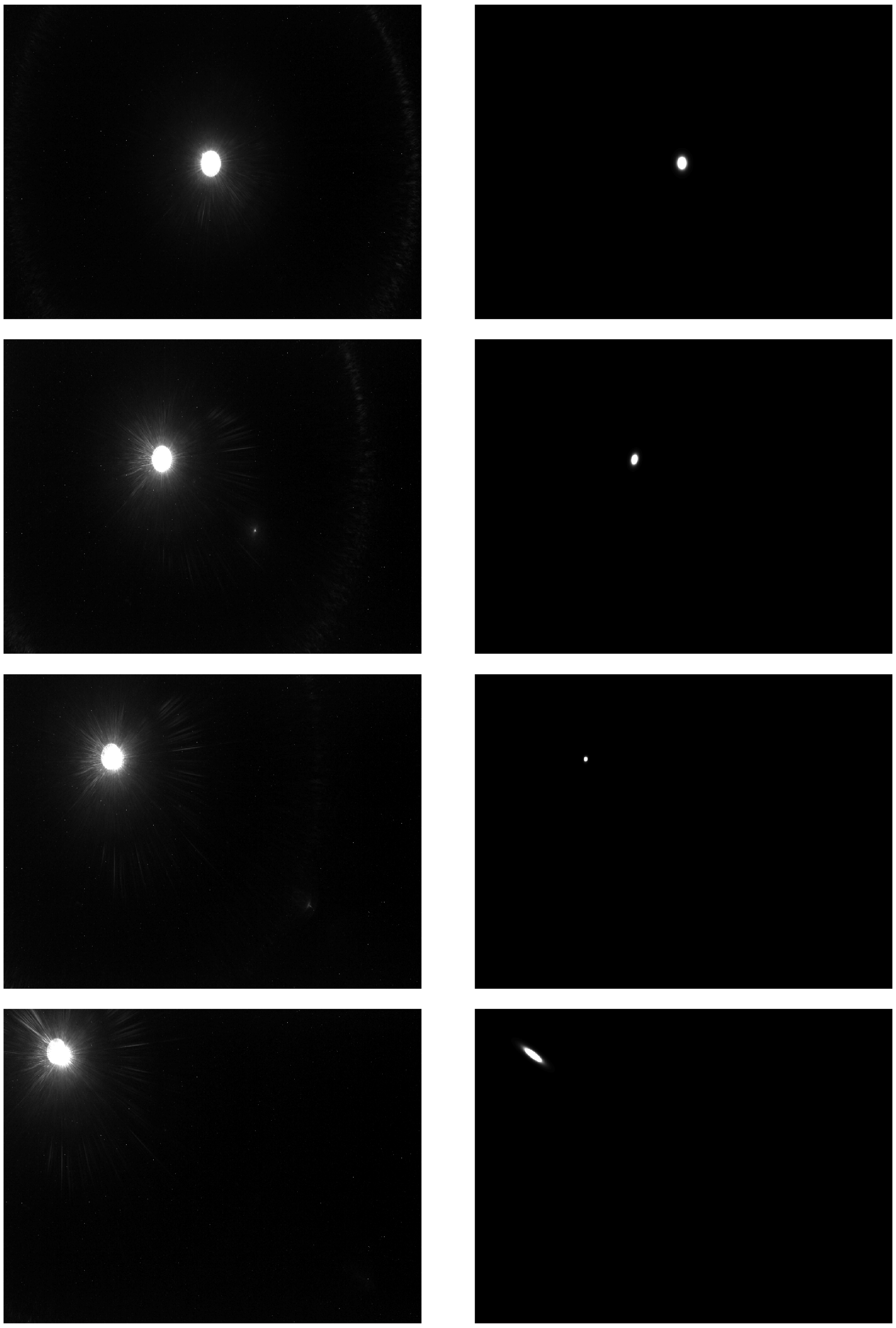


Figure 7.2: Measured PSF to the left and corresponding PSF-estimate to the right. The estimates are more concentrated and tilts more than the measurements. The PSF has been normalized to $[0, 1]$. Color axis: $[0, 10^{-5}]$

7.2 Conclusions

In Chapter 1.1 we stated the following questions.

- Does existing equipment at Axis suffice to measure the point spread function?
- Do the measurements depend on the rotation of the camera around the optical axis, or is the camera rotationally invariant?
- Can the camera be modelled as a shift-invariant system, or must one account for shift-variance?
- Given a stray light compensation strategy, is it feasible in practice, i.e., in the hardware of the surveillance camera?

Before leaving this treatment of stray light behind, at least for now, our answers to these questions are as follows.

- The measurement setup in Chapter 3 seems to be sufficient for measuring the point spread function.
- In Figure 6.1 we see that the camera is rotationally invariant.
- It is difficult to conclude whether the camera can be sufficiently modelled as a shift-invariant system. However, the results point to the fact that significant stray light compensation is achieved by using shift-invariant deconvolution.
- Implementing a shift-variant deconvolution in today's cameras is not viable due to the constrained performance of the system. However, there are promising approximations to remedy this problem. It should be noted that for some optics, a shift-invariant deconvolution might yield satisfactory results. Considering these alternatives we believe that a stray light compensation strategy similar to the one we have used is fully feasible in a camera.

7.2.1 Contributions of this master thesis

Below we list the contributions of this master thesis.

- The use of a convolutive model for describing stray light has been motivated.
- A novel framework for measuring the point spread function has been presented.

- Kernel thresholding and image downsampling have been suggested to be used for reducing the computational complexity.
- An experimental setup for objective performance comparisons has been introduced.
- Several comparisons between different deconvolution algorithms have been presented.
- Finally, this master thesis is a broad overview of stray light compensation, which should be particularly useful for persons uninitiated to stray light analysis.

7.3 Future work

This master thesis has answered several questions regarding stray light compensation. However, as is often the case when studying a phenomenon in detail, one often ends up with more questions than one started off with. For future work, the following questions may be interesting to consider.

- What are the differences when using other interpolation techniques for the PSF-measurements? How many measurements are needed? How do the interpolation techniques differ with respect to stray light compensation?
- How does one implement the stray light framework in the hardware of a camera?
- How do the proposed approximation techniques perform in practice? Can they be improved?
- Are there better experimental setups for measuring the PSF? For example, how does our setup compare to measurements from using a collimator?
- Is the experimental setup for measuring the stray light compensation (the face experiment) truly objective? Are there other experiments that could be performed?

Appendix A

Noise sensitivity of Van Cittert's method

We give a heuristic proof of why Van Cittert's method (Chapter 5.3.1) tends to be sensitive to noise. This is best explained in the Fourier domain. Let us consider the 1D case for simplicity; the extension to 2D is straightforward. We use the default initialization $\hat{o}^{(0)} = i$. Recalling (5.2), the observed image can be decomposed as $i = \tilde{i} + n$, where $\tilde{i} = s * o$ and n is additive noise. The first iteration of Van Cittert's method yields

$$\hat{o}^{(1)} = \hat{o}^{(0)} + \left(i - s * \hat{o}^{(0)}\right) = i - (i - s * i) = 2i - s * i.$$

Entering the Fourier domain, we get

$$\hat{O}^{(1)}(f) = 2I(f) - SI(f) = [2 - S(f)]\, I(f) = [2 - S(f)] \left[\tilde{I}(f) + N(f)\right].$$

The additive noise is usually high-frequent. Hence, since the information in a natural image generally is quite low-frequent, it should hold that $|\tilde{I}(f)| < |N(f)|$ for large f. Furthermore, for a typical PSF it holds that $|S(f)|$ is small for large f, whereas it is relatively large for small f. Following these heuristics, we conclude that

$$\hat{O}^{(1)}(f) \approx 2\left[\tilde{I}(f) + N(f)\right] \approx 2N(f) \text{ for large } f,$$

whereas

$$\hat{O}^{(1)}(f) \approx c\left[\tilde{I}(f) + N(f)\right] \approx c\tilde{I}(f) \text{ for small } f,$$

where c is significantly smaller than 2. The interpretation is that after one iteration, Van Cittert's method has increased the noise portion of the signal more relative to the signal portion. Therefore, the noise level has increased. This effect increases as the number of iterations increases, which can be shown analogously.

Appendix B

Selected-ordinates sampling

To see if selected-ordinates sampling yields a reasonable approximation of (5.24), we perform a simple experiment. The parametric PSF introduced in Chapter 2.1 is used; the parameters are given in Table 6.3.

We want to simulate how the sampling pattern looks for the computation of the convolution at the pixel $(0, 0)$, in the center of the image. To simplify this experiment, we only study the sampling along the row passing through the peak of the PSF. Furthermore, since the PSF is symmetric in the center, we will only consider the right half of the row. That is, we only consider the pixel set $X = [0, 1, \ldots, \frac{N}{2}]$ where N is the image size. We denote the PSF along the middle row by $w(x) = s(x, 0, 0, 0; \beta, \sigma, b, d, f, g)$. To simplify further, we assume that the underlying continuous image f is given by $f(x) = 1, \forall x$.

In Figure B.1 we see the continuous PSF $w(x)$, which in this case is equal to $w(x)f(x)$; the sampled PSF using uniform sampling ($x \in X$); exact selected-ordinate sampling based on the continuous PSF; and approximate selected-ordinate sampling based on the uniformly sampled PSF. Observe that the uniformly sampled PSF corresponds to the PSF captured by the camera. First, we note that the approximate selected-ordinate sampling based on the uniformly sampled PSF is not perfectly matching true selected-ordinate sampling. This implies that the implicit multiplications may be somewhat off. Second, we observe that the selected-ordinate sampling yields major errors when approximating the area. If this kind of miss-approximation occurs for other $f(x)$, one should consider if the loss in precision is worth the savings given by the implicit multiplications.

Note that one should not blindly focus on the PSF-weights. Looking at 2.2, it is clear that the product of the PSF-weights and the underlying image is what matters in the end. In a somewhat constant image, this should not pose any issues, but it might yield some unwanted artifacts in images with regions of significantly different intensities. Moreover,

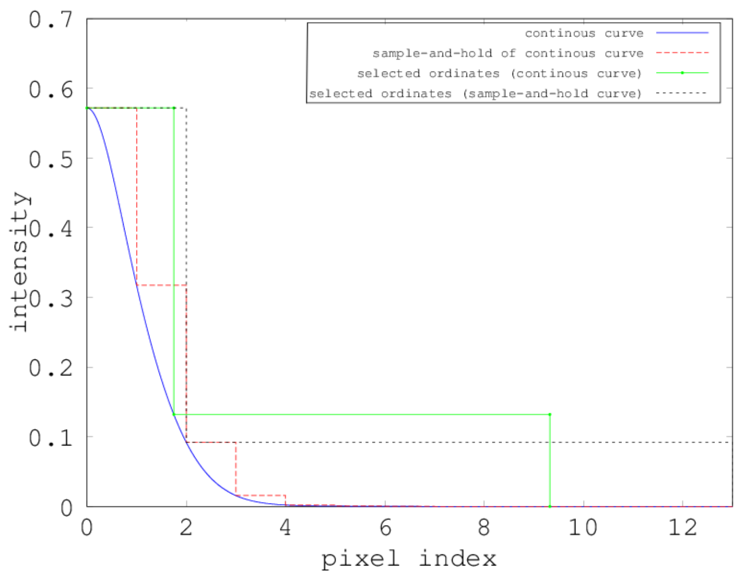


Figure B.1: Illustration of the selected-ordinates sample pattern. Note that the sample-and-hold curve corresponds to the sampling performed by a digital camera.

if there are many terms in 2.2, the sum may still be significant even though each term is insignificant by themselves.

We conclude this reasoning by adding that in practice, the errors we have shown here might be small enough to be negligible. Since we have not performed enough such experiments we remain cautious.

Bibliography

[1] Eric C Fest. *Stray Light Analysis and Control.* SPIE Press, 2013.

[2] Burak Bitlis, Peter A Jansson, and Jan P Allebach. Parametric point spread function modeling and reduction of stray light effects in digital still cameras. In *Electronic Imaging 2007*, pages 64980V–64980V. International Society for Optics and Photonics, 2007.

[3] Jianing Wei, Burak Bitlis, Alan Bernstein, Akila de Silva, Peter A Jansson, and Jan P Allebach. Stray light and shading reduction in digital photography: a new model and algorithm. In *Electronic Imaging 2008*, pages 68170H–68170H. International Society for Optics and Photonics, 2008.

[4] Jianing Wei, Charles A Bouman, and Jan P Allebach. Fast space-varying convolution using matrix source coding.

[5] Jianing Wei, Guangzhi Cao, Charles A Bouman, and Jan P Allebach. Fast space-varying convolution and its application in stray light reduction. In *IS&T/SPIE Electronic Imaging*, pages 72460B–72460B. International Society for Optics and Photonics, 2009.

[6] Feng Xiao, Jeffrey M DiCarlo, Peter B Catrysse, and Brian A Wandell. High dynamic range imaging of natural scenes. In *Color and Imaging Conference*, volume 2002, pages 337–342. Society for Imaging Science and Technology, 2002.

[7] Eino-Ville Talvala, Andrew Adams, Mark Horowitz, and Marc Levoy. Veiling glare in high dynamic range imaging. In *ACM Transactions on Graphics (TOG)*, volume 26, page 37. ACM, 2007.

[8] James G Nagy and Dianne P O'Leary. Restoring images degraded by spatially variant blur. *SIAM Journal on Scientific Computing*, 19(4):1063–1082, 1998.

[9] Bryce E. Bayer. Us patent 3,971,065. 1975.

[10] Mohd Awais Farooque and Jayant S Rohankar. Survey on various noises and techniques for denoising the color image.

[11] Stephen J Wright and Jorge Nocedal. *Numerical optimization*, volume 2. Springer New York, 1999.

[12] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[13] Steven W Smith et al. The scientist and engineer's guide to digital signal processing. 1997.

[14] Peter A Jansson. *Deconvolution of images and spectra*. Courier Corporation, 1997.

[15] P. A. Jansson. Us patent 6,829,393. 2004.

Master's Theses in Mathematical Sciences 2015:E21
ISSN 1404-6342

LUTFMA-3279-2015

Mathematics
Centre for Mathematical Sciences
Lund University
Box 118, SE-221 00 Lund, Sweden

http://www.maths.lth.se/