POLITECNICO DI MILANO
Corso di Laurea Specialistica in Ingegneria Informatica
Dipartimento di Elettronica e Informazione

RESTORATION OF BLURRED OBJECTS USING CUES FROM THE ALPHA MATTE

AI & R Lab
Laboratorio di Intelligenza Artificiale
e Robotica del Politecnico di Milano

Relatore:
Prof. Vincenzo CAGLIOTI
Correlatori:
Dr. Giacomo BORACCHI
Ing. Alessandro GIUSTI

Tesi di Laurea di:
Andrea Villa
matricola 673882

Anno Accademico 2006-2007

To my parents, Dede and Alberto.

Abstract

Very few works focus on the restoration (deblurring) of a blurred object when the surrounding background is sharp. This challenging task involves issues which are related to image segmentation, digital image matting and blind deconvolution.

In this scenario, this thesis proposes a technique for estimating the blur point spread function exploiting the information provided by the alpha matte, when a single image containing a blurred object is available. The alpha matte contains information about the apparent transparency of the blurred object. The main idea consists in estimating the unknown silhouette of the blurred object by thresholding the alpha matte and obtaining then the blur point spread function with the deconvolution between the blurred and the thresholded alpha map.

The thesis begins presenting an essential state-of-the-art of matting, image formation and point spread function estimation; then the problem is stated and the proposed approach is accurately described in all its steps. Implementation details and experiments conclude the work.

Our approach has been validated with experiments on both synthetic and camera images. On synthetic images the blur point spread function has been estimated with reasonable accuracy even in presence of additive white noise, while on camera images, the restoration performance using the estimated point spread function encourages further works in this direction.

Ringraziamenti

Desidero ringraziare caldamente il prof. Vincenzo Caglioti per avermi offerto l'opportunità di svolgere questa esperienza. Ringrazio specialmente Alessandro Giusti e Giacomo Boracchi per la paziente supervisione, i preziosi consigli, e le sconvolgenti caramelle finlandesi. Vorrei esprimere inoltre la mia gratitudine a coloro che hanno contribuito con delle fotografie che ho adoperato nelle prove sperimentali: Alberto Villa, Alice Sosio e symx_82 su *flickr*^{TM}.

Contents

Introduzione

L'argomento della presente tesi si colloca in due rami dell'informatica: l'elaborazione digitale delle immagini e la visione artificiale. In particolare, gli argomenti di ricerca più attinenti al lavoro svolto sono il *matting* e la *deconvoluzione blind*. Il matting consiste nell'estrarre degli oggetti dallo sfondo in un'immagine, ad esempio allo scopo di realizzare dei fotomontaggi. Con deconvoluzione blind si indicano varie tecniche ideate per ripristinare un'immagine affetta da sfocatura, perché fuori fuoco oppure mossa. Viene definita *blind* (cieca) in quanto si prescinde dalla conoscenza della funzione che descrive la sfocatura. Tale funzione prende nome di *point spread function* (PSF), e corrisponde all'immagine di una sorgente luminosa puntiforme.

Lo scopo della tesi è lo sviluppo di una tecnica per ripristinare immagini di oggetti fuori fuoco o mossi. Viene considerato ovvero il caso in cui la degradazione riguarda un oggetto isolato, il che accade quando l'oggetto si trova ad una profondità diversa dallo sfondo, o quando l'oggetto è il movimento. L'approccio proposto ha inizio con la selezione dell'oggetto da ricostruire e la scomposizione dell'immagine in tre componenti: il colore dell'oggetto (*color map*), lo sfondo e la mappa di trasparenza (*alpha map*). Il principale contributo della tesi è il peculiare approccio alla stima della PSF basato sull'alpha map. Nota la PSF, l'immagine dell'oggetto viene ripristinata usando algoritmi di deconvoluzione. L'approccio viene infine validato con immagini di prova, in parte sintetiche e in parte ottenute da fotografie digitali.

(a) (b)

Figura 1: (a) Un'immagine interamente sfuocata. (b) Immagine contenente un oggetto mosso.

La Figura 1(a) mostra il caso più frequentemente considerato in letteratura, in cui tutta l'immagine è affetta dalla degradazione che si intende correggere. Diversamente, in questa tesi, si considera il caso in cui la sfocatura o il movimento riguardano un oggetto isolato, come chiarito dalla Figura 1.1(b). Siccome l'esito del matting è di grande importanza per le fasi successive, viene utilizzato uno degli approcci di maggior successo a questo difficile compito, proposto da Levin, Lischinski e Weiss in [23]. Il loro metodo si fonda su ipotesi di regolarità locale delle immagini dell'oggetto e dello sfondo. Definiscono una funzione di costo in cui l'alpha map è la sola variabile incognita, e da essa ricavano un sistema di equazioni lineari da risolvere. Per inizializzare la procedura, l'utente deve inoltre marcare l'oggetto e lo sfondo con dei tratti bianchi e neri, come ulteriormente chiarito nella Sezione 3.1.

Lo scopo principale del matting è l'estrazione di un oggetto dalla scena, per esempio allo scopo di sostituire lo sfondo di un attore. Le stesse tecniche sono utilizzabili anche con oggetti sfocati e mossi, in quanto sono in grado di gestire le zone dove il colore del soggetto si combina con lo sfondo.

La tesi si concentra principalmente attorno alla possibilità di stimare la PSF sfruttando l'informazione fornita dall'alpha map. In particolare,

questo comporta la stima della sagoma dell'oggetto applicando una soglia all'alpha map, e la deconvoluzione dell'alpha map sogliata dall'alpha map. Si ottiene così una stima grossolana della PSF, i cui parametri sono facilmente identificati, specialmente per PSF a simmetria centrale come il disco o la gaussiana.

Nota la PSF, una qualunque tecnica di deconvoluzione può essere usata per ricostruire l'immagine dell'oggetto. In questa fase, si propone anche un processo di raffinamento iterativo della PSF, finalizzato a migliorare la qualità del risultato. Infine si propongono alcuni semplici accorgimenti per filtrare l'alpha map precedentemente ricostruita, riducendone ulteriormente gli eventuali difetti. La procedura iterativa di raffinamento è ispirata al lavoro sulla deconvoluzione blind di immagini bicolore illustrato in [24].

L'approccio proposto è stato implementato in ambiente *Matlab* e sottoposto a verifica sperimentale in diversi scenari. Prove realizzate con immagini sintetiche in assenza di rumore dimostrano la validità dell'approccio nelle condizioni più favorevoli, poi si verifica l'adeguatezza delle misure adottate per neutralizzare gli effetti del rumore. Infine si propongono alcuni esperimenti con vere fotografie digitali, nelle condizioni tipiche delle applicazioni pratiche.

Sebbene i risultati siano incoraggianti, vi sono ancora molte limitazioni, tra cui la dipendenza dalla correttezza della fase di matting. Il problema del matting è intrinsecamente difficile, e anche i migliori algoritmi disponibili sono solamente in grado di fornire una soluzione visivamente gradevole, non garantendone la correttezza a meno di condizioni molto particolari. Quando l'immagine di sfondo è nota a priori, si può tuttavia ottenere un matting più accurato usando il metodo proposto in [14].

Attualmente, la degradazione dell'oggetto è descritta da una PSF parametrica. Tale scelta da un lato conferisce velocità ed efficienza all'implementazione, ma pone anche delle forti limitazioni della flessibilità complessiva

dell'approccio. Sviluppi futuri potranno riguardare l'adattamento di una delle moderne tecniche di deconvoluzione blind non parametrica con i vincoli aggiuntivi sulle caratteristiche dell'alpha map. J. Jia ha proposto già una soluzione adatta sia ad oggetti che ad immagini globalmente mosse, e ulteriori sviluppi in tale direzione sono auspicabili [18].

La presente tesi è organizzata nel modo seguente: nel Capitolo 2 si mostra lo stato dell'arte del matting e della deconvoluzione blind, con particolare riguardo ai lavori che hanno maggiore attinenza con il presente. Il Capitolo 3 illustra i concetti, le definizioni fondamentali e discute i principali contributi teorici di questo lavoro. La struttura del capitolo è suddivisa tra il matting e l'analisi dettagliata dell'approccio proposto. I contributi originali della tesi si trovano principalmente nelle seguenti sezioni: nella 3.1.1 si propone un approccio al matting basato su trasformazioni lineari nello spazio RGB. La Sezione 3.3.1 tratta la stima della PSF usando l'alpha map, mentre la 3.4.1 si concentra sul perfezionamento del risultato.

Gli aspetti implementativi sono trattati nel Capitolo 4, seguendo un approccio *top-down*: prima si fornisce una visione complessiva, poi si presenta una analisi dettagliata delle singole fasi. Il Capitolo 5 mostra i risultati sperimentali, in cui si mette alla prova l'approccio in varie condizioni. Le prove sono realizzate sia con immagini sintetiche, sia con immagini reali. Le conclusioni nel Capitolo 6 riassumono brevemente il lavoro svolto, e discutono alcune delle possibili applicazioni e sviluppi futuri. Infine, nell'Appendice A, vengono allegate la parti essenziali del codice sorgente.

Chapter 1

Introduction

The topic of this thesis is situated in two large branches of computer science: digital image processing and computer vision. In particular, natural image matting and image restoration are the most pertinent fields.

The main goal of this work is developing an approach to selectively restore blurred objects in a single image. In general, the blur degradation only affects a single, isolated object, while the rest of the image is not affected by it. The proposed approach starts from selecting a blurred object and decomposing the image into three layers: foreground, background and alpha (i.e. the transparency of the object) maps using state of art matting techniques. The blur degradation point spread function (PSF) is estimated taking advantage of the information provided by the matte. Then, the image of the object is restored using classical image restoration approaches. Numerous test cases have been provided to validate the proposed approach, with both synthetic and camera images.

Figure 1.1(a) shows a blurred image, which is the case found in most previous works on deblurring. On the contrary, in this work, the case of blur affecting an isolated object is considered, as clarified by Figure 1.1(b). Accurate computation of the matte is of critical importance for the subsequent phases. One of the most successful approaches to this difficult task

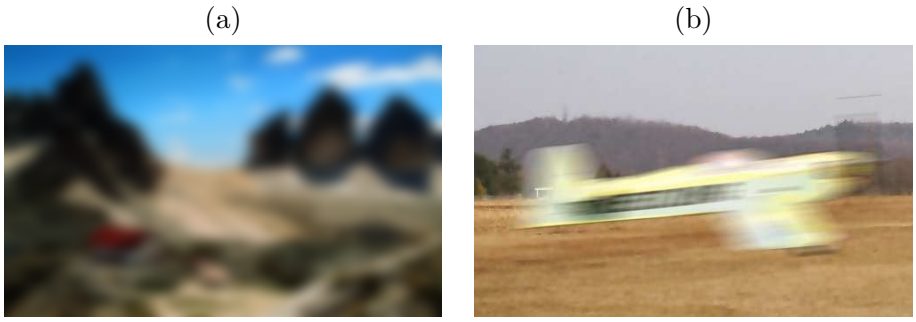

Figure 1.1: (a) A blurred image. (b) Image containing a blurred object due to motion.

was proposed by Levin, Lischinski and Weiss in [23]. Their method relies on the assumption of local smoothness of the foreground and background images. They derive a cost function only dependent on the transparency values. With the help of additional user defined constraints, in the form of manual strokes indicating foreground and background regions, a sparse system of linear equations is defined and used to solve the matting problem.

While the main purpose of matting techniques is typically to separate an object form the scene, these can also be used in the case of blurred objects, whose boundaries are mixed with the background image.

The main focus of the thesis consists in exploring the possibility to estimate the blur PSF exploiting the information provided by the matte. In particular, this is done by estimating the true silhouette of the object by applying a segmentation threshold to the alpha map. By deconvolution of such estimate from the computed matte, it is possible to roughly estimate the underlying PSF. This approach, which is very simple and efficient, works particularly well in case of rotationally symmetric filters such as Gaussian low-pass blur or disk. In some particular cases, it even works on motion blurred objects, and, in this case, an *ad hoc* PSF estimation procedure has been proposed.

Any classical image restoration technique can be used to restore the im-

age of the blurred object, once the PSF is known. In this phase it is optionally possible to start an iterative PSF refinement process aimed at improving the quality of the restored image, and finally apply further post-processing operations on the reconstructed alpha map to reduce the remaining artifacts. The iterative PSF refinement process is inspired to the research on blind deconvolution of binary images in [24].

The proposed approach has been implemented and tested to verify its behavior under various conditions. Synthetic tests with noiseless input proved the adequateness of the approach under the specified assumptions. Further tests on noisy images allow to analyze how the noise affects the estimation, and to test the measures adopted to neutralize its negative effect. Last experiments finally consider real camera images, which represent a good way to stress the approach under practical conditions.

While the results are encouraging, there are still many limitations in this approach. The first one is the intrinsic dependence on the accuracy of the matting algorithm. The matting extraction problem is intrinsically difficult, and even the best state of the art algorithms are only able to provide a nice looking solution, without guaranteeing its correctness unless very particular conditions are met. In case the background image is known in advance, the method proposed in [14] can be applied for a more accurate matting process. Moreover, since deconvolution is an ill-conditioned operation, all these errors together with noise further hinder the restoration process.

Currently, the blur is described by a simple parametric PSF. While this choice confers speed and efficiency to the implementation, it also poses a strong limitation to the flexibility of the approach. Ongoing works concern the adaptation of the existing blind deconvolution methods with the additional constraints of binary alpha maps. In [18] J. Jia already proposes a solution for motion blur, which can be applied both for blurred objects or blurred images, however more research on this particular aspect is still

desirable.

This thesis is organized as follows. In Chapter 2 are shown the state of the art techniques that are related to this work. In particular, the areas of natural image matting and blind deconvolution are taken into account. Chapter 3 illustrates basic concepts, definitions and discusses the main theoretical contributions of this work. The chapter is subdivided between the extraction of the matte, which is very important for the outcome of the following phases, and the in-depth analysis of the original approach that is proposed. In particular, the original contributions of this thesis are in the following sections: Section 3.1.1 covers a matting approach using linear transformations in RGB space. Section 3.3.1 deals with the PSF estimation from the alpha matte. Section 3.4.1 focuses on the parameter refinement procedure.

The most relevant implementation aspects are described in Chapter 4, following a top-down approach: first a general overview is given, then follows a detailed analysis of the the various phases. Chapter 5 shows some experimental results to verify how the proposed approach performs in practice. Both synthetic images and real camera images are used to prove the effectiveness of the approach. The conclusions are included in Chapter 6, that briefly summarizes the contents of the work and proposes some final considerations on the principal achievements that have been attained. Some potential applications are also discussed, before ending with a general outline of the possible developments. Finally, in Appendix A is reported the most significant part of the source code used to produce the experimental results.

Chapter 2

State of the art

The selective restoration of a blurred object given a single image requires two important steps: first the blurred object has to be isolated from its background, then its image has to be restored. The two research areas in image processing that pertain to these phases are natural image matting and image restoration. In what follows we show the most important state of art achievements that lead to this work.

2.1 Digital matting

The classical *matting problem* in image processing consists in separating a foreground image from a background image. One typical application in movies would be extracting an actor from the scene, to substitute the background image. Since this thesis mainly aims at restoring objects without altering background features, it is critical to understand and choose the most successful approaches to this challenging problem.

The observed image (I) is modeled as a convex linear combination of foreground (F) and background (B), where the coefficients represent opacity information (the alpha map or *matte*), according to the Porter-Duff com-

positing equation [29]:

$$I = \alpha F + (1 - \alpha)B \tag{2.1}$$

The problem consists in computing α, F and B from the observation I. However, there exists an infinite number of solutions, since for every pixel in color images there are three equations and seven unknowns. Therefore, the problem is severely under-constrained.

Early approaches have been used to simplify the problem by using a constant backing color, which often happened to be blue, hence these techniques are commonly referred to as *blue screen matting*. Extensive research and numerous patents exist on this subject [12, 35, 33, 27]. Even in this case, however, the human eye is still the best judge to evaluate the solution.

There are numerous times where the knowledge of the background color is not available, so the previously mentioned techniques cannot be used. In these cases there exist other approaches that try to solve the problem by making various assumptions on the input data and the unknowns. This particular research field is known as natural image matting.

Many techniques require a *trimap* as input to start the matting process. A trimap is an auxiliary image indicating which regions are certainly part of the background, which are part of the foreground and finally the mixed areas that have to be solved [7, 32, 34]. In Bayesian matting approaches assumptions are made on the local color distribution of F and B. The matte with the maximum probability is then chosen as solution [7, 32].

Another trimap-based method is Poisson matting [34]; they compute the matte starting from the image gradient, solving a Poisson equation. To obtain good results it is often necessary to undergo an iterative local refinement process of the matte demanding significant manual editing.

Disadvantages of these methods include the accurate and tedious manual painting of an adequate input trimap. While some techniques were developed to provide automatic trimap generation from few manual input strokes,

these are not suited to all situations [25].

Recently, some more successful approaches have been proposed [16, 36, 23]. These are generally characterized by a scribble based user interface, where a limited set of user input strokes indicating background and foreground is used to find the solution. One of these approaches is the iterative algorithm proposed by M. Cohen and J. Wang [36]. The matting solution is obtained by iteratively setting up a suitable *conditioned random field* problem. The estimation is initially done only on the pixels that are close to the input strokes, and further propagates after each iteration till the image is fully covered and the uncertainty of the solution is minimized. More details on this are seen in Section 3.1.

The other two approaches in [16, 23] are based on the optimization of a quadratic cost function derived from the matting equation and additional assumptions on the local smoothness of the foreground and background colors. Under the given conditions, the dependency of the cost function on F and B can be eliminated, and the alpha matte is computed as the solution to a sparse set of linear equations. In particular, the method described by Levin *et al.* in [23], which is briefly summarized in the next chapter, will be used to provide input data for the experiments with actual camera images in Chapter 5.

2.2 Image restoration

The following degradation model of the blurred objects is considered:

$$I = I_0 \circledast h + \eta \tag{2.2}$$

where I is the observed image of the object, I_0 is its true *latent* image, h is the blur kernel also called point spread function (PSF) and η is Gaussian white noise. $\circledast$ denotes the convolution operation.

The problem consists in finding the true image I_0 prior to the degradation. When the PSF is known, this is the classical image restoration problem, that can be solved using different deconvolution techniques. The approach preferred in the current implementation is the Wiener deconvolution, based on the least squares minimization of the expected mean square error [15]. Other methods however exist for this purpose, such as the Richardson-Lucy algorithm, that converges to the maximum likelihood estimate for I_0 [26, 31].

These methods alone are not much useful in practical situations, because the PSF is often unknown and very little information is available, other than the observed image itself. Therefore a very interesting research topic is *blind deconvolution*, well summarized by D. Kundur and D. Hatzinakos in [19].

There exists a vast number of blind deconvolution techniques, offering different compromises of optimality, computational complexity and convergence properties that usually have to be accurately considered according to the application requirements.

Most techniques can be classified in two main groups. The first one comprehends those that separate the model estimation process from the image restoration itself; these can be called *a priori* blind deconvolution methods. For example, in astronomical imaging it is sometimes possible to directly identify the blur filter by inspecting the image of a point light source on a constant background [1]. Other examples are those methods based on inspecting the pattern of zeroes in the Fourier domain, allowing the identification of rectilinear motion blur or out of focus blur PSF [4, 3, 5]. Once the PSF is estimated, the restoration can be performed using any of the classical restoration techniques.

The methods in the second category are characterized by the simultaneous reconstruction of both I_0 and the PSF. Numerous algorithms in this group are naturally more complex and can be generally divided in subcategories, according to [19]. Some of the most successful are briefly summa-

rized: one class of these algorithms models I_0 as an auto-regressive (AR) process and the PSF as a symmetric moving average (MA) process. Thus, identifying the ARMA parameters leads to the solution. Most techniques in this subarea typically use maximum-likelihood estimation, generalized cross-validation or even neural networks [21, 30, 6].

Other important blind deconvolution algorithms are the non-parametric techniques based on image constraints. In these cases neither the blur or the image are described by parametric models. Instead, various image constraints are assumed and used to perform an iterative reconstruction of the latent image and the PSF simultaneously.

Fergus *et al.* developed a method to blindly remove the effects of global motion blur caused by camera shake during the exposure interval, using prior knowledge on the statistical distribution of the image gradient [11].

Another blind motion deblurring method exploiting a prior on the distribution of the gradient is proposed by A. Levin in [22]. While this approach is restricted to uniform rectilinear motion, it has the advantage of segmenting the image in areas having the same blur PSF. This segmentation allows the authors to consider images depicting objects moving at different velocities in the scene.

In [24], Ta-Hsin Li and Keh-Shin Lii propose an iterative blind deconvolution method for two tone images. They iteratively alternate the estimation of the unknown base tones with that of the deblurring filter. Since the alpha map of blurred objects computed in the matting phase can be considered a degraded black and white image, this work has influenced some of the theoretical aspects discussed in Section 3.4.

Contrarily to most of the existing methods, in this thesis it is assumed that the image is degraded according to Equation (2.2) only at those pixels depicting the considered object. The same PSF affecting the object applies

to the latent alpha map α_0 as well:

$$\alpha = \alpha_0 \circledast h + \eta. \tag{2.3}$$

The object color and alpha maps therefore share the same degradation model.

Very recently J. Jia *et al.* proposed a motion deblurring algorithm based on the alpha matte, providing a unified way to estimate the motion in both cases of camera shake or object motion [18]. Their work indeed shares some of the goals with the present thesis, and it makes use of the same prior knowledge on the binary nature of α_0. However it differentiates in the estimation approach, defining a *Maximum A Posteriori* framework and solving it with iterative optimization algorithms.

Chapter 3

Theoretical approach

This chapter focuses on the theoretical aspects of the selective restoration of blurred objects in a single image.

In Section 3.1 the concern is the separation between the object and the background data. This is the well known problem of natural image matting. A new approach based on linear transformations in RGB space is analyzed; then the main state-of-art natural image matting techniques are briefly discussed. Their key concepts are analyzed to expose how to obtain better results with blurred objects.

After dealing with matte extraction, Section 3.2 introduces the image formation process, along with the assumed image degradation model. Then, the core elements of the proposed approach are analyzed under ideal conditions in Section 3.3. The restoration takes place in two phases: first an estimation of the blur kernel is derived from the opacity values computed in the matting process. Subsequently, classical deconvolution strategies are used to restore the original image. The last phase also incorporates an iterative refinement of the previously estimated parameters, seen in Section 3.4.

Since the ideal conditions are rarely met in real camera images, the factors that negatively influence the result are introduced in the last section.

Finally, strategies to counteract the effects of noise and to improve the overall quality of the composition are discussed.

3.1 Separating foreground and background data

The first step in the direction of selectively deblurring objects in an image is to extract color and opacity values from the background. This operation is equivalent to the well known problem of *extracting the matte*, or 'pulling the matte' as found sometimes in literature. The observed image is typically modeled as a convex linear combination of the foreground color F_i and the background color B_i for every pixel i.

$$I_i = \alpha_i F_i + (1 - \alpha_i) B_i \tag{3.1}$$

The α_i term represents the opacity values of the foreground image; its value ranges in the $[0, 1]$ interval, representing all possible values in the transition between 100% background to 100% foreground pixels.

The matting problem consists in estimating α, F and B for each pixel i, starting from the combined image I. The matting problem is heavily under-constrained, since the image I is often the only information that is available. As already mentioned in Chapter 2, many techniques have been studied and developed to face this problem, by adding various constraints and making assumptions on the properties of the solution. Since the quality of the matte is critical for the outcome of the following phases, the most successful approaches to date have been taken into account.

Matting algorithms are generally aimed at extracting an object from its background for later use, such as composition on a new background. Typically the objects are not blurred, however boundary pixels as well as transparent regions are a mixture of background and foreground color that have to be computed, as seen in Figure 3.1. The presence of mixed pixels is naturally more relevant in the case of blurred objects, so it is critical that the

chosen matting algorithm correctly handles this case. The matting problem

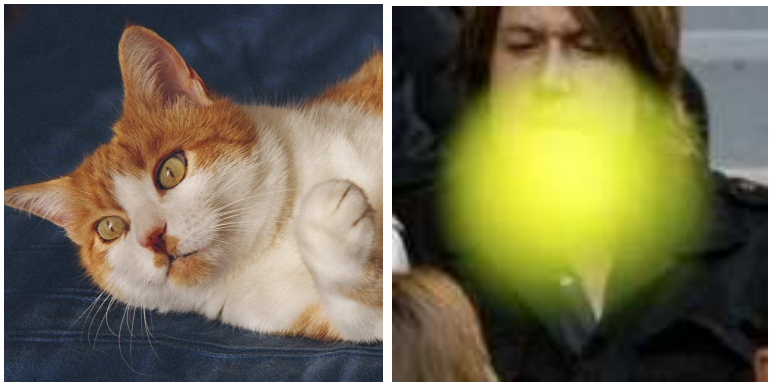

Figure 3.1: Typical matting problems. Even when the object is not blurred, there are regions where foreground and background colors are mixed. The mixed regions are more relevant in case of blurred objects, as seen in the image on the right.

can be solved in a more simple way when some of the terms in Equation (3.1) are known in advance. For example, in blue screen matting, the key idea is to take a photograph of the subject against a known, constant color background. However, the problem remains under-constrained and, in real applications, the human eye is still required to ultimately judge whether a result looks correct [33].

Recently, another method has been proposed in [14], which focuses on motion blurred objects, but still holds in the out of focus case. The main assumptions here include knowledge of the background image and the object color lying on a single plane in the RGB space. In these conditions, the matte is fully constrained and can be computed. This is very desirable for many applications that depend on an exact solution rather than one that looks good enough.

Noise and uncertainty on the alpha matte is a common problem with camera images, even when using state of the art approaches. In the following section a fast and reliable method to compute the matte in particular

conditions is discussed.

3.1.1 Matting using linear RGB transformations

A new approach to matting using linear transformation in RGB space is now derived.

Let the target be a color image, where background and foreground colors for each pixel are known and, of course, different. This constraint will be relaxed later; in this particular case, the solution to Equation (3.1) is trivial, but it is still worth to mention its meaning in the RGB color space to introduce the subsequent generalization. From the Equation (3.1), all

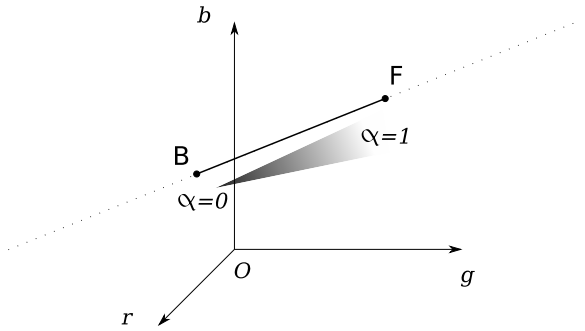

Figure 3.2: RGB space representation of the target image I. *All pixels lie on the* BF *segment: alpha ranges from zero to one when moving from* B *to* F.

pixels in the image are known to lie on the BF segment, because they are convex linear combination of the fixed points B and F. Their position on BF is directly related to their opacity value α. With this geometric interpretation in mind, a solution to the matting problem can be computed with an appropriate 3D linear transform in homogeneous coordinates. The idea is finding a new reference system such that BF coincides with the unit length vector on one axis; this way, the alpha values can be immediately retrieved.

The H matrix that follows accomplishes exactly this purpose:

$$H^{-1} = \begin{bmatrix} & & & \\ \vec{d_r'} & \vec{d_g'} & \vec{d_b'} & \vec{B} \\ & & & \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

It is easier to define H in terms of its inverse: $\vec{d_r'}, \vec{d_g'}, \vec{d_b'}$ represent the directions of the new reference axes with respect to the old one. $\vec{d_r'} = \vec{F} - \vec{B}$ to make BF coincident with the new "alpha" axis. $\vec{d_g'}$ can be any direction orthogonal to $\vec{d_r'}$. Finally, $\vec{d_b'} = \vec{d_r'} \times \vec{d_g'}$ is simply the cross product of the first two. At this point, all image pixels are converted to homogeneous coordinates and pre-multiplied by H. The transformed image will contain the matte in the alpha channel.

The very strict assumption on the background and foreground colors can be relaxed if they are allowed to lie on a RGB line passing for the origin of the coordinate system. This formulation of the problem is more interesting than the first one, since it can handle, for example, Lambertian surfaces under a single light. When these two lines are different, all pixels in the target image lie on the RGB plane defined by the three points O, B and F, as seen in Figure 3.3.

Like the previous case, a linear transform in RGB space will be derived to solve the problem. The whole H matrix can be expressed in terms of four simpler transformations, that are now discussed one at a time. The first matrix, H_1, transforms the *OBF* plane into the *Org* plane. This can also be seen as switching reference system as suggested by Figure 3.3. Moreover, F's image becomes the unit vector on the transformed axis r'.

$$H_1^{-1} = \begin{bmatrix} & & & 0 \\ \vec{F} & \vec{F} \times \left(\vec{F} \times \vec{B}\right) & \vec{F} \times \vec{B} & 0 \\ & & & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

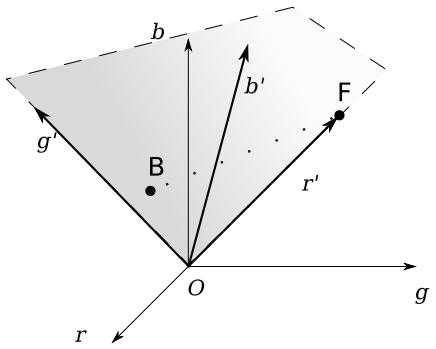

Figure 3.3: RGB space representation of the target image. All pixels lie on the plane defined by the points O, B and F; r', g' and b' are the new reference axes after transformation H_1 is applied.

Since all pixels are now expected to lie on the plane defined by the red and green axis, the blue value can be discarded, applying H_2. Points that did not lie on such plane, possibly because of image noise, are projected there.

$$H_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

At this time, the problem becomes two-dimensional since all points are tied to the *rg* plane. The next step consists in finding a degenerate planar homography H_3 collapsing all points to the red axis. To compute such transformation, which has eight degrees of freedom, at least four points and their corresponding images have to be known: each point i allows to write two homogeneous equations of the kind $x'_i = \mathrm{H}x_i$. Figure 3.4 illustrates the situation, after transforms H_1 and H_2 have been applied. To solve for H_3, the *Direct Linear Transformation* method from Hartley and Zisserman, 2004 [17], pp. 88–93 is used.

The computation of H_3 may potentially introduce errors in the matte. In fact, for each pixel with alpha value different from either zero or one, it

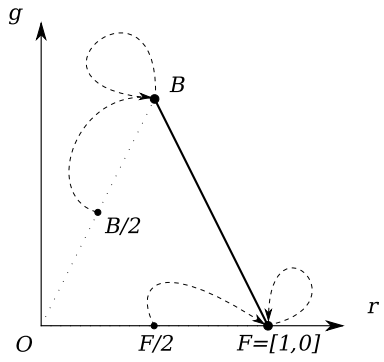

Figure 3.4: To compute H_3 the following mapping is being used: both F and F/2 transform into [1, 0], while B and B/2 transform into B.

is assumed that the linear combination of foreground and background color takes place on a line parallel to BF. This holds only when both foreground and background colors are affected by the *same* amount of shading.

Figure 3.5 better illustrates the underlying ambiguity: every image pixel X is a convex linear combination of the true unknown colors F_t and B_t; H_3 projects each pixel on BF along the line that connects the pixel itself to O[1]. The alpha value is computed on BF, while the real value lies on B_tF_t: this implies that real and computed values coincide when the triangles $\triangle OBF$ and $\triangle OB_tF_t$ are similar. If this is not the case, then the profile of the alpha map will not be correct. A practical example supporting this fact is proposed in Figure 3.6.

Finally, the coordinate system has to be transformed so that BF, where all points now lie, becomes a unit vector on the red axis, using the following 3×3 matrix.

$$H_4^{-1} = \begin{bmatrix} \vec{d_x} & \vec{d_y} & \vec{B} \\ 0 & 0 & 1 \end{bmatrix}$$

Where $\vec{d_x} = \vec{F} - \vec{B}$ and $\vec{d_y} \perp \vec{d_x}$ are the new orientation of the reference axes,

[1]Because the homography conserves collinearity

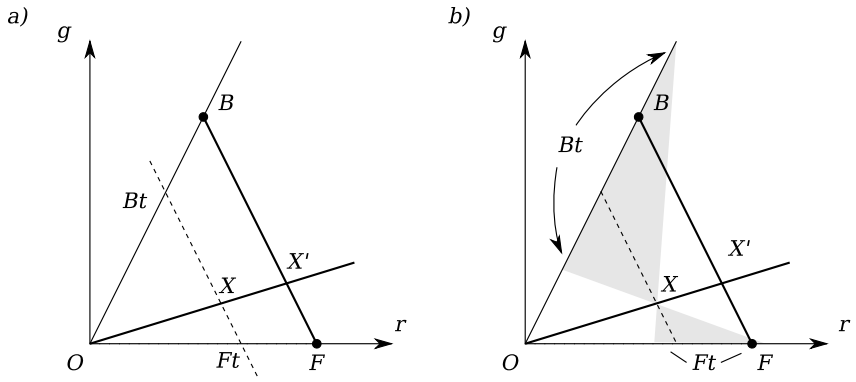

Figure 3.5: On the left side, the ideal condition for the algorithm is shown: X *is the pixel transformed by* H_2H_1. X' *is its image after applying* H_3. *The situation may actually differ since segment* BtFt *may not to be parallel to* BF*; it just needs to belong to the star of lines passing through X. Therefore, the ratio between* BX' *and* BF *may not correspond to the real alpha value of the pixel.*

scaled to normalize BF's length to 1.

Naturally, all steps can be applied in one single matrix multiplication:

$$H = H_4H_3H_2H_1.$$

As already discussed, strong shading differences between foreground and background will definitely lead to unexpected results. This method was tested on two synthetic images[2] shown in Figure 3.6: (a) and (b) display a successful matte, while (d) and (e) show a failure. The white artifacts in the lower left sector in 3.6(e) are caused by numerical instability around black pixels. The phenomenon arises because H transforms the black color into the homogeneous 2D point $[0, 0, 0]^T$, which is undefined. Moreover, the different shading between foreground and background causes the profile of the computed alpha map (e) to differ from the ground truth data (c). This method has also been tested with a real photograph. The subject is a red rough paper circle affected by a strong defocus blur, in the leftmost image

[2]The most relevant part of the source code can be found in Appendix A.1.

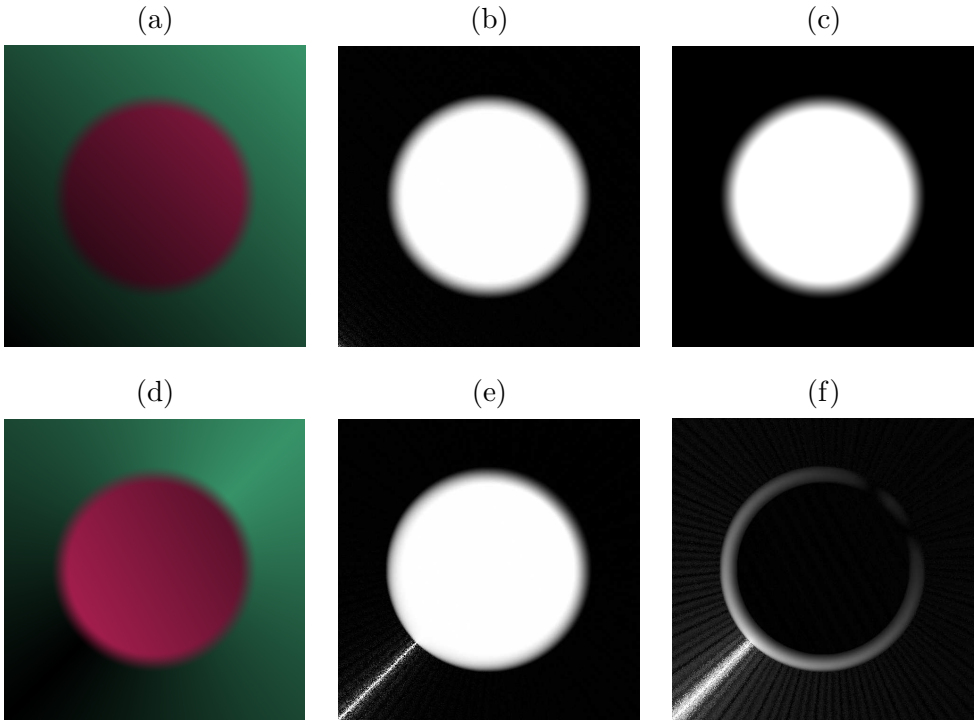

Figure 3.6: (a) show a synthetically generated test image meeting the stated preconditions. The result (b) of the matte extraction coincides with the ground truth alpha map provided in (c). Figure (d) stresses the pitfalls of the algorithm: the gradient in the background was modified to include a line of black, ill-posed pixels and the foreground gradient is different. (f) visualizes the error (e) - (c), slightly enhanced for clarity.

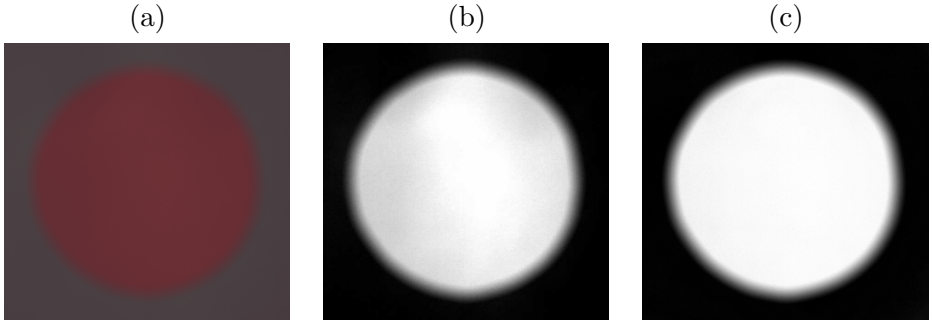

Figure 3.7: Testing done with an actual photograph. (a) The target image. (b) Matte obtained with the trivial version of the algorithm. (c) Matte obtained with the complete version of the algorithm.

of Figure 3.7.

The photograph was taken in a fairly uniform lighting condition, and the rough paper presents no specular highlights, but some shading variations still affect it. Foreground and background colors F and B to be used in the algorithm were obtained as the mean of two 50×50 sample regions. Figure 3.7(b) shows the results from the trivial version of the algorithm. It fails to compute a correct matte because foreground and background colors are not constant. The more refined method, which is able to handle small amounts of shading variations in the image, leads to a better overall quality of the matte, suitable for restoration purpose.

Even though the proposed method is quite fast and efficient, it works under heavily constrained conditions that hardly occur in real situations. The remaining part of this section deals with the more general — and difficult — case of natural image matting. This is usually the choice made in practical situations, when the information available on the starting image is limited to the image itself.

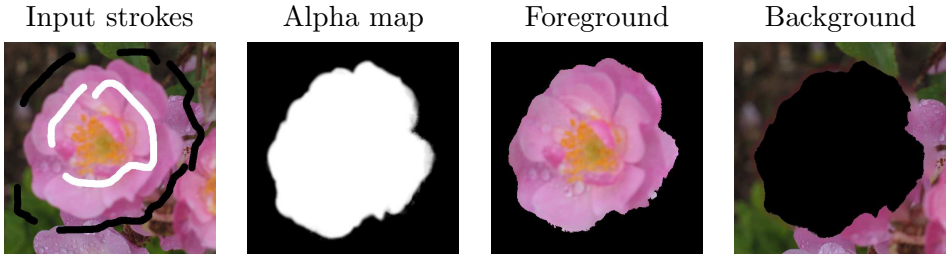

Figure 3.8: Typical matting scenario. Alpha, foreground and background maps are computed from a limited set of user defined scribbles.

3.1.2 Natural image matting methods

As introduced in the previous chapter, the most successful natural image matting approaches to date were proposed by Levin *et al.* in [23] and Cohen-Wang in [36]. While the first approach was chosen to extract the matte from real camera images in chapter 5, there are also some interesting aspects in the other one that are worth mentioning. In particular, the latter approach — which is more complex — has arguably a better control over the smoothness of the alpha map typically found in blurred objects. A brief explanation of the two methods follows to support this statement.

A closed form solution to natural image matting What follows is the natural image matting technique proposed by Levin *et al.* in [23]. To extract the matte, a cost function is derived from the compositing Equation (3.1) by assuming a specific color model for both background and foreground colors. This cost function only depends on the starting image and the unknown alpha values, and can be minimized by solving a linear system of equations. A complete *trimap* is not required, although the user must manually provide some additional information in the form of black and white strokes (see Figure 3.8). In the case of *greyscale* images, these colors are assumed to be constant in a small window around each pixel. The matting equation can

be locally rewritten as follows:

$$\alpha_i \approx aI_i + b, \quad \forall i \in w, \tag{3.2}$$

where $a = \frac{1}{F-B}, b = -\frac{B}{F-B}$ and w is a small image window. Taking into account all pixels, the following cost function is defined:

$$J(\alpha, a, b) = \sum_{j \in I} \left(\sum_{i \in w_j} (\alpha_i - a_j I_i - b_j)^2 + \varepsilon a_j^2 \right) \tag{3.3}$$

The regularization term εa_j^2 is added for numerical stability reasons, and also to favor smoother alpha maps. It is advisable to increase it when processing noisy or compressed images.

At this time, the cost function to be minimized still has $3N$ unknowns for an image with N pixels. However, a and b can be eliminated from the (3.3). In particular:

$$J(\alpha) = \min_{a,b} J(\alpha, a, b) = \alpha^T L \alpha, \tag{3.4}$$

Where L is the so called *Matting Laplacian*, a $N \times N$ matrix whose elements (i, j) are defined as follows:

$$\sum_{k|(i,j) \in w_k} \left(\delta_{ij} - \frac{1}{|w_k|} \left(1 + \frac{1}{\frac{\varepsilon}{|w_k|} + \sigma_k^2} (I_i - \mu_k)(I_j - \mu_k) \right) \right) \tag{3.5}$$

μ_k and σ_k^2 are the mean and variance of the image intensity inside the window w_k of size $|w_k|$. δ_{ij} is the Kronecker delta: $\delta_{ij} = 1$ if $i = j$, 0 otherwise.

Key elements in the proof are rewriting J in matrix notation and computing the optimal values a^* and b^* by solving a least squares problem, leaving a quadratic function in alpha. Further details can be found in [23].

At this point, the remaining cost function is minimized for α, given the initial set of input strokes S:

$$\alpha = \arg\min_{\alpha} \ \alpha^T L \alpha + \lambda (\alpha^T - v_S) D_S (\alpha - v_S), \tag{3.6}$$

where λ is a 'large' number; D_s is a diagonal matrix whose elements on the diagonal are 1 for user-constrained pixels and zero for all others; v_S is finally a vector containing the constrained alpha values, and zero where pixels are unconstrained. Differentiating and setting derivatives to zero, the minimization is achieved solving the following sparse linear system:

$$(L + \lambda D_S)\alpha = \lambda v_S \tag{3.7}$$

In case of color images, the initial constraint on foreground and background colors can be relaxed: they are allowed to locally vary on a single line in *RGB* space. This assumption is also referred to as the *color line model*; the definition of L has to be slightly modified, but the key passages are essentially the same.

Finally, it is necessary to compute the foreground and background colors. This is either done solving for a and b in (3.2) or, better, by defining a new cost function derived from the matting equation with additional smoothness constraints on F and B are included.

The algorithm has a publicly-available implementation, is efficient and robust, so it was finally chosen to compute mattes from actual camera images. A multi-resolution version is also available, to deal with large images in a matter of minutes while keeping low the memory requirements. Even though the tests in this work were conducted using this approach, the method from [36] is also worth mentioning.

Iterative optimization approach for unified segmentation and matting In 2005 Wang and Cohen proposed an iterative algorithm to extract a matte starting from a limited set of user defined strokes [36]. They use a *Belief Propagation* optimization method to estimate the unknown terms [10]. While an exhaustive description of the approach is not needed for this work, a quick overview is now presented to introduce those elements that can help in case of blurred objects.

For each pixel, their algorithm estimates foreground and background colors, alpha values and the uncertainty, ranging in the [0, 1] interval. The unknown pixels close to low uncertainty regions are estimated first; the algorithm stops when all pixels have been taken into account and the total uncertainty cannot be further reduced. This is accomplished by defining a *Conditioned Random Field* problem and solving it with available belief propagation algorithms.

The image and a set of user defined input strokes are required, as in the previous case. User-constrained regions start with an uncertainty of 0, alpha of 0 for background pixels and 1 for the foreground; the correspondent background or foreground color are set accordingly. All other pixels start with maximum uncertainty and initial $\alpha = 0.5$. To limit the size of the problem, all pixels are divided into three regions: U_c containing all pixels whose alpha values have already been estimated with zero uncertainty. $\tilde{U}_c$ contains all pixels that have been considered but still have a non-zero uncertainty. Finally, U_n contains pixels that were not touched by the procedure yet.

At the start of each iteration, a small region in U_n within 15 pixels from U_c is moved to $\tilde{U}_c$. A *Markov Random Field* (MRF) is generated defining a node for each pixel in $\tilde{U}_c$. Edges are defined according to a 4-connectivity scheme. Pixels with zero uncertainty connected to the aforementioned ones are added to the network as well. The energy function to be minimized by the process is made up by two terms:

$$V = \sum_{i \in \mathrm{MRF}} V_d(\alpha_i) + \sum_{i,j \in \mathrm{MRF}} V_s(\alpha_i, \alpha_j) \tag{3.8}$$

V_d is the *data* energy term. It reflects the likelihood that the estimated variables are correct, penalizing, for example, variable assignments that do not satisfy the alpha composition equation. The other term, V_s, is a function designed to explicitly penalize abrupt changes between neighboring alpha

values:

$$V_s(\alpha_1, \alpha_2) = 1 - e^{-(\alpha_1 - \alpha_2)^2/\sigma_s^2}, \tag{3.9}$$

where σ_s is tuned empirically. This energy term is interesting because it allows to fine-tune the smoothness of the alpha map. In case of blurred objects, the alpha map as well as the foreground color are always quite smooth, so image discontinuities are imputable to the background. In Levin's approach, instead, foreground and background are assumed smooth and discontinuities in the image are more likely to be reflected in the alpha map. This suggests that V_s can be redesigned to better fit the smooth matte of blurred objects.

At each iteration, the MRF is solved and a new one is built according to the updated values. The process is repeated until the set U_n is empty and the total uncertainty is minimized.

Future work in this area may involve implementing this method aiming specifically at blurred objects.

3.2 Blurred objects model

After the extraction of the matte of the desired object has been accomplished, the next step consists in restoring its image. Before introducing the actual blur estimation approach, the most important models for image formation and degradation have to be discussed.

Out of focus blur formation A simple optical system is now proposed, to introduce basic elements of geometric optics that are useful for the formulation of the problem. Under ideal conditions, assuming thin lenses and small viewing angles, all the light coming from a point source X converges to a single point X' after being deflected, according to the convergence law:

$$\frac{1}{d} + \frac{1}{r} = \frac{1}{f}, \tag{3.10}$$

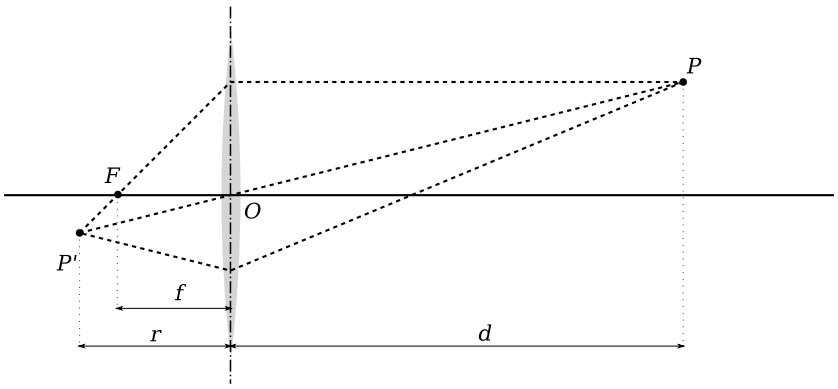

Figure 3.9: A simple optical system.

in which f is the *focal* distance of the lens system. When the image plane is placed exactly a a distance r from the lens, the point X is at focus. Instead, when the image plane does not satisfy Equation (3.10), the point is out of focus. The image of the point X is then the intersection between the image plane and the light coming from X. In particular, the image of a point source is called *Point Spread Function* (PSF). If the aperture of the lens has a circular shape, then also the PSF is a disk, called the *circle of confusion*. The image is formed by the additive contributes of all points in the scene.

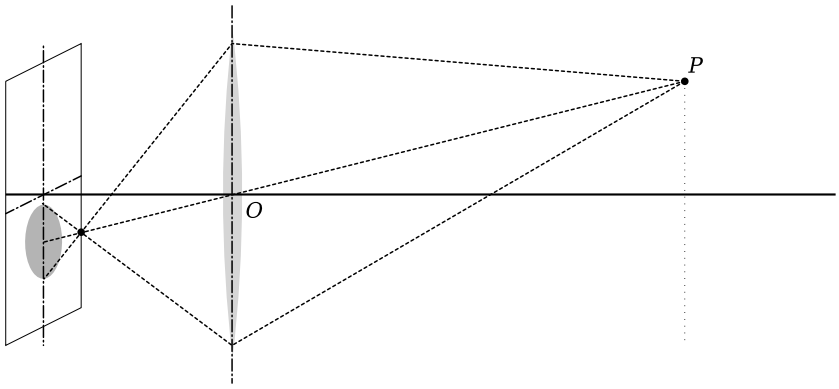

Figure 3.10: The circle of confusion

When all the points of the photographed object lie on a plane parallel to

the image plane, the PSF is also shift-invariant. In these conditions, the degraded image of an object is the output of a linear shift invariant system described by the PSF.

Motion blur formation Another source of blur in images is the relative motion of the camera and the objects during the exposure interval $[t_0,\ t_1]$.

A motion blurred image can be modeled as the integration of an infinite number of 'still' images during the exposure interval:

$$I = \frac{\int_{t_0}^{t_1} I_t \mathrm{d}t}{t_1 - t_0} \tag{3.11}$$

According to this, the alpha map represents the fraction of exposure time during which the object was projected to each pixel. Then a strong relation between motion blurred objects and alpha matting is in place.

When the object is moving on a plane parallel to the image plane, and perspective effects are negligible, its image I_t in Equation (3.11) is translating on the apparent trajectory $x(t)$ during the exposure interval, so it can be written as:

$$I_t = I_0(x(t_0) - x(t)) \tag{3.12}$$

By substituting (3.12) in (3.11), the observed image corresponds to the convolution of the object with its apparent trajectory during the exposure interval:

$$I = I_{t_0} \circledast \frac{x(t)}{t_0 - t_1} \tag{3.13}$$

Where $\circledast$ denotes the convolution operator. This work mainly focuses at rectilinear constant speed motion blurred objects.

To summarize, in both out of focus and motion blur, the observed object image $I(x, y)$ is modeled as the two-dimensional convolution of the true, object image $I_0(x, y)$ with the PSF $h(x, y)$, possibly adding Gaussian white

noise $\eta \sim N(0, \sigma^2)$.

$$\begin{aligned} I(x,y) &= \sum_{j,k=-\infty}^{+\infty} h(j,k) I_0(x-j, y-k) + \eta(x,y) = \\ &= I_0(x,y) \circledast h(x,y) + \eta, \end{aligned} \tag{3.14}$$

In order to keep constant the mean intensity of the image, the PSF also has to sum to 1:

$$\sum_{j=-\infty}^{+\infty} \sum_{k=-\infty}^{+\infty} h(j,k) = 1. \tag{3.15}$$

Note that the same blur model describes the alpha map of the blurred object. Therefore, the PSF is unchanged when considering the alpha map:

$$\alpha(x,y) = \alpha_0(x,y) \circledast h(x,y) + \eta(x,y). \tag{3.16}$$

Note that the PSF is here assumed to be shift-invariant: the same degradation model applies to all pixels.

To recover the latent image, *deconvolution* of the PSF from the degraded image is used. In the classical image restoration problem, the filter is known and there exist various techniques to recover the latent image: for example, these include the trivial inverse filtering, the more robust Wiener deconvolution and Lucy-Richardson iterative algorithm. Before proceeding to the estimation of the PSF, a brief overview of the most relevant blur models is proposed.

3.2.1 Defocus blur PSF

To generate out of focus blur for synthetic tests, two PSF models are taken into account: Gaussian lowpass filtering and disk blur. The Gaussian blur kernel of standard deviation σ (centered on (0, 0)) is defined as follows:

$$g(x,y) = \frac{1}{2\pi\sigma^2} \cdot e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{3.17}$$

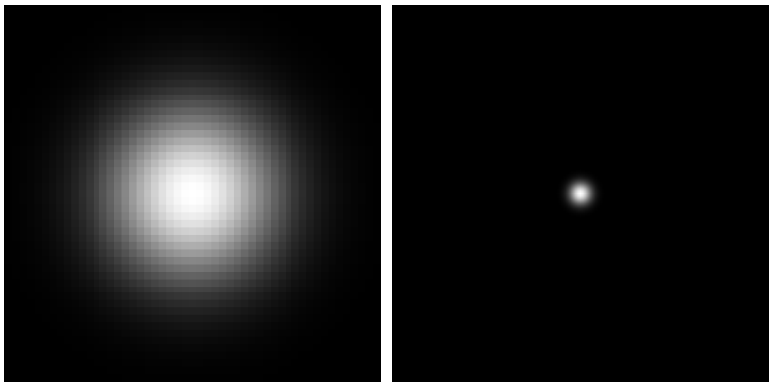

Figure 3.11: Gaussian blur PSF; absolute value of its Fourier transform is still a (different) Gaussian.

while a disk blur kernel of radius r is:

$$d(x,y) = \begin{cases} \frac{1}{\pi r^2} & \text{if } x^2 + y^2 < r \\ 0 & \text{otherwise} \end{cases} \tag{3.18}$$

Disk blur resembles the one usually found in camera images; its Fourier

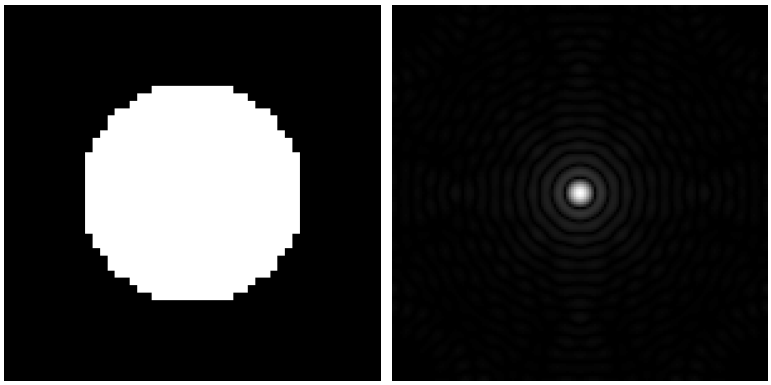

Figure 3.12: Disk blur PSF. The circular patterns formed by the zeros in the Fourier domain can be observed.

domain representation is characterized by zeros approximately lying on concentric circumferences. This property has been exploited in the past to estimate the disk radius looking for a regular pattern of zeroes in the frequency

plot of the original image [13].

A Gaussian blur PSF in Fourier domain is still Gaussian. This can be an advantage for synthetic tests since it does not have zeroes that cause instability when performing deconvolution. Quantization error alone does in fact prevent the cancellation of zeros in numerator and denominator when performing a simple inverse filter deconvolution. Moreover, disk blur tends to create hard edges from highlighted points and lines, as Figure 3.13 shows, while Gaussian blur gracefully smooths all features in the image.

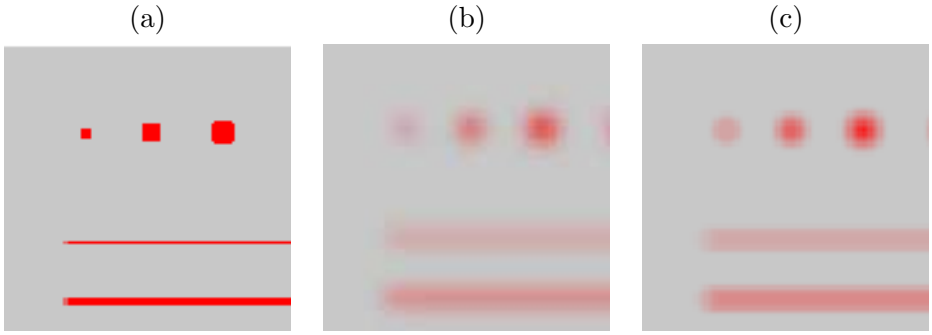

Figure 3.13: (a) starting test image. (b) Gaussian blur, sigma = 3 px. (c) Disk blur, radius = 4 px. Note how disk blur preserved sharper edges in correspondence of small points and thin lines.

3.2.2 Motion blur PSF

This work focuses at the blur produced by objects moving in front of a still camera, rather than the blur caused by camera movements in the exposure interval. The most recent works on global shift-invariant motion blur appeared in [11, 18].

In general, the appearence of motion blurred objects cannot be modeled in a simple way. The trajectory of the motion may be very complicated and perspective deformation may invalidate the shift-invariance assumption. For these reasons, only the case of objects moving at uniform speed in front of

the camera has been considered in this work. According to this, the PSF is modeled by two parameters (θ, l): these represent the direction and the extension of the apparent motion of the object on the image plane. Next

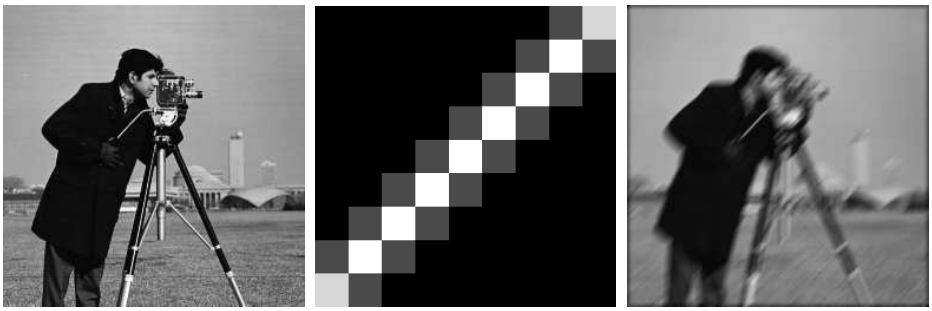

Figure 3.14: Uniform motion blur: 12 pixel at a $45°$ direction

section deals with the new approach to parameter estimation that exploits the information provided by the alpha matte.

3.3 Estimating the PSF from opacity values

One of the most relevant original contribution in this thesis is presented in this section.

As previously discussed, it is assumed that both the object and its opacity values are degraded by a linear shift-invariant system, which corresponds to the convolution with the PSF in spatial domain (3.15). The convolution operation in Fourier domain is equivalent to a multiplication, so that the following relation holds:

$$\alpha = \alpha_0 \circledast h + \eta \iff A = A_0 \cdot H + N \tag{3.19}$$

where $A = \mathcal{F}(\alpha)$, $A_0 = \mathcal{F}(\alpha_0)$, $H = \mathcal{F}(h)$ and $N = \mathcal{F}(\eta)$. From now on, upper-case letters denote the 2D Discrete Fourier Transform of the corresponding term, unless otherwise specified.

Neglecting noise, Equation (3.19) can be rewritten as:

$$H = \frac{A}{A_0}. \tag{3.20}$$

This indeed suggests that the PSF can be estimated as deconvolution of the latent alpha map from the blurred alpha map, when a good estimation of α_0 is available.

3.3.1 Alpha map segmentation

One of the key concepts explored in this work is the estimation of α_0 by segmentation of the blurred alpha map.

The latent alpha map of the object, before any blur filter is applied, is a binary image, made of either black (background) or white (object) pixels. This assumption implies that semi-transparent objects are not taken into account, and that pixels are back-projected to ideal lines. A rough yet simple way to estimate α_0 is using a threshold:

$$\alpha_0 \approx \hat{\alpha}_0 = \alpha > t$$

Without any *a priori* information on the shape of the object, the threshold level is set to $t = 0.5$. In what follows the errors introduced by the assumption of a fixed threshold level are computed. First, a distinction has to be made according to the nature of the degradation affecting the image: defocus blur and motion blur are treated separately. The case of both these occuring simultaneously is not taken into account in the scope of this work.

Defocus blur PSF paramater estimation In case of out of focus objects, the most relevant fact is the relation between the curvature of the object's apparent contour and the estimation error. Straight edges are correctly estimated: when the (rotationally symmetric) blur kernel is centered on such edges, it covers both object and background pixels in a 1:1 ratio; the output of the filter at the edge pixels is then 0.5, equal to the threshold

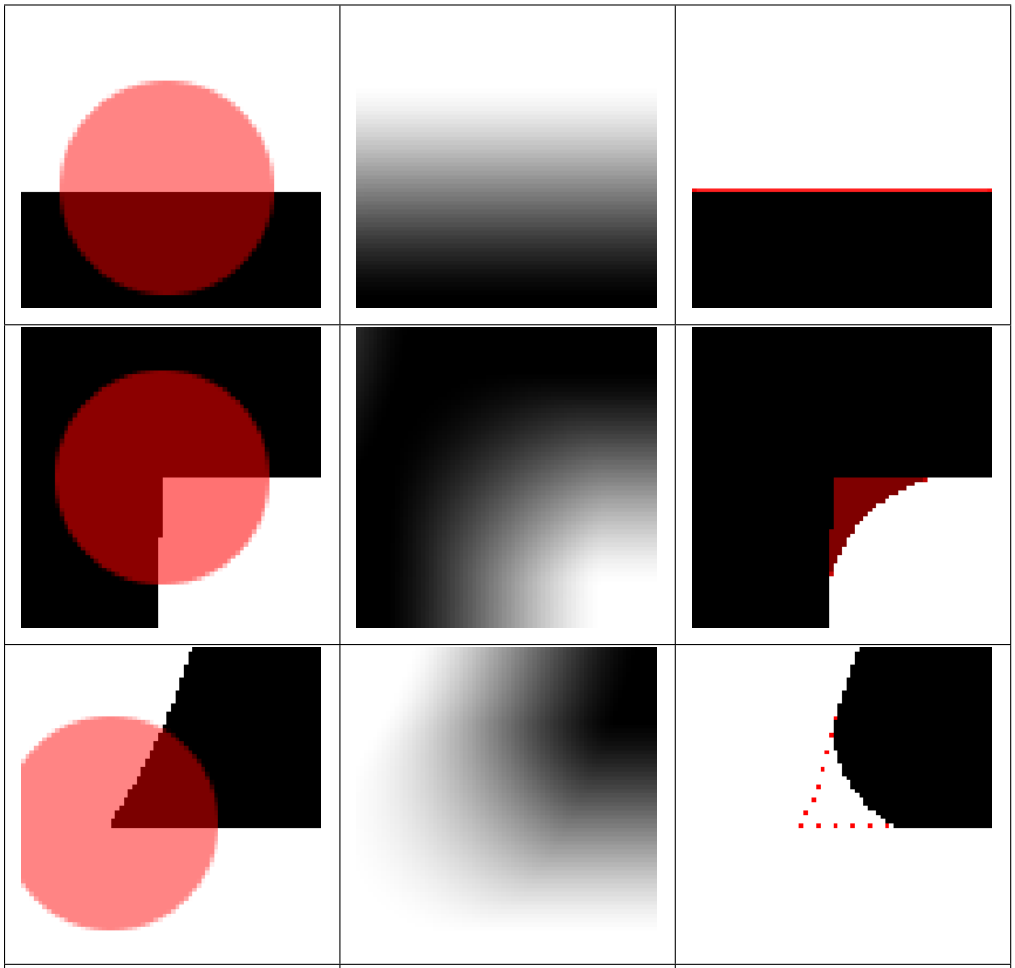

Figure 3.15: Estimation of the unknown two-tone alpha map using a segmentation threshold of 0.5. Blur was generated by a disk kernel of r=25 px. The first row shows how straight edges are correctly recognized. The second row shows the error introduced where the object has a convex contour. Finally the third row shows what happens near concave corners.

level. Therefore this threshold correctly estimates α_0 in areas containing straight edges.

This is not the case when the object contours are curved lines, or in the proximity of corner points. When the object contour is convex, the correct threshold value is smaller than 0.5, and the opposite holds with respect to concave object boundaries. Figure 3.15 further clarifies this by comparing α_0 and $\hat{\alpha}_0$ in three different situations.

As an example, the relation between the optimal threshold value t^* and the curvature radius of the edge is now derived in the case of uniform disk blur (see Figure 3.16). When the blur kernel crosses the contour of the

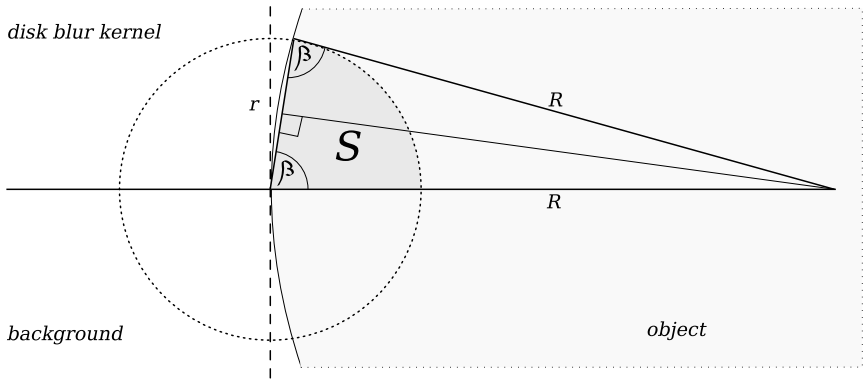

Figure 3.16: In the case of uniform disk blur, the optimal threshold value can be approximated as twice the sector S over the whole kernel area

object, the output of the deconvolution is the area of the intersection between the kernel and the object over the whole area of the disk. Let r be the radius of the blur filter and $R > 0$ the curvature radius of the edge. The area of the intersection is approximated as a circular sector of the disk PSF, corresponding to $2S$ in Figure 3.16. According to this:

$$R\cos\beta = \frac{r}{2}$$

so

$$\beta = \arccos \frac{r}{2R}.$$

Therefore, an estimate for the correct threshold is given by t^*:

$$t^* \approx \frac{2S}{\pi r^2} = \frac{\beta}{\pi}$$

This result agrees with the previous statement on straight line edges, that can be seen as circumferences having $R \longrightarrow \infty$ so that $t^* \longrightarrow \frac{1}{2}$.

Summarizing:

$$\begin{cases} t^* < \frac{1}{2} & \text{if convex} \\ t^* = \frac{1}{2} & \text{if straight} \\ t^* > \frac{1}{2} & \text{if concave} \end{cases}$$

It is now important to verify that, despite the estimation errors on $\hat{\alpha}_0$, it is still possible to identify the PSF parameter (i.e. the σ or the disk radius) with reasonable accuracy. The first estimation of the PSF is performed by Wiener deconvolution. In Fourier domain, this corresponds to:

$$\widehat{H} = A \cdot \frac{\hat{A}_0^*}{|\hat{A}_0|^2 + k},$$

where k is a regularization parameter to improve numerical stability. More details on the Wiener filter are provided in Section 3.4, dealing with image restoration. Figure 3.17 shows a fairly complex image containing all kinds of features that cause errors in the estimation of α_0. The segmented alpha map is definitely far from perfect, and the PSF estimated directly is useless for image restoration. However, when assuming a parametric PSF, its parameter can be identified by minimizing the L^2 norm of the error:

$$r = \arg\min_r \left\| \text{disk\_psf}(r) - \widehat{\text{psf}} \right\|_2 . \tag{3.21}$$

The minimization can be performed using any optimization technique. A procedure to further refine the parameter value is discussed later, in Section 3.4.

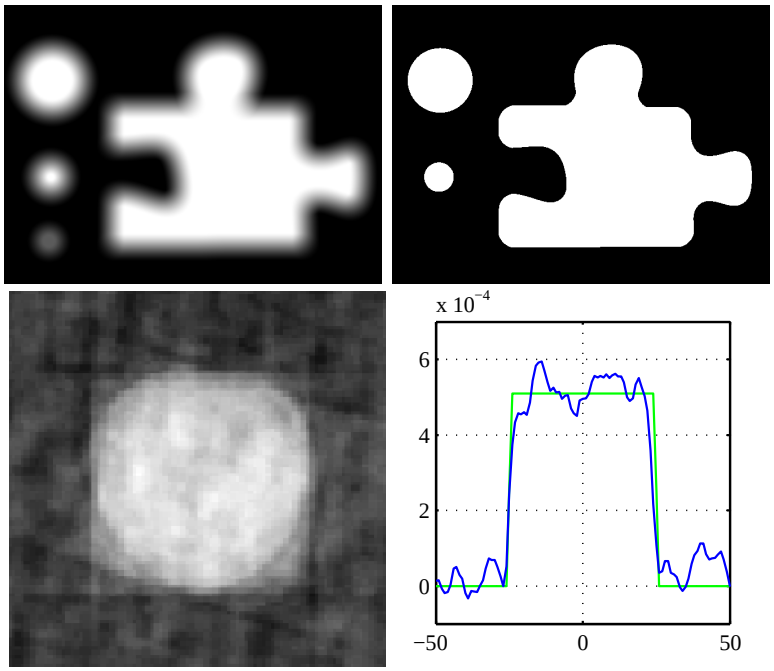

Figure 3.17: PSF estimation from the alpha map. The estimated PSF can be used to identify, in this case, the disk radius.

Motion blur PSF parameter estimation The streak of a moving object in motion blurred images has some interesting properties as shown in [2]. The alpha map can be subdivided into regions according to the number of times that each pixel is crossed by the apparent contour of the moving object. Fully background or foreground regions are named R0 since they are never touched by the moving contour. Regions that are traversed once are called R1, and so on. The boundary between these regions is made visible by filtering the alpha map with a Laplacian kernel, as seen in Figure 3.18. Note that some of the edges in the Laplacian correspond to the contour of

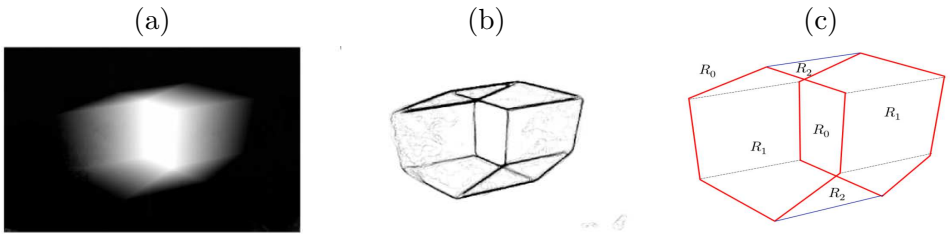

Figure 3.18: (a) Alpha map of a motion blurred object. (b) Laplacian of (a). (c) Regions classification. Images from [2].

the object at the extremes of the exposure interval. The other edges represent the envelope of the moving contour. Another important property is related to the R1 regions. Here is an interesting relation between the apparent contour of the object and the *iso-alphas*, curves formed by pixels having the same alpha value. Note that at alpha values of 0.5, the iso-alpha curves represent the apparent contour at the middle of the exposure interval. More details on this matter are in [2].

In other words, iso-alphas at level 0.5 approximate the the object's silhouette 'frozen' in the position occupied at the middle of the exposure interval. However, outside R1 regions, the iso-alpha curves do not correspond to the actual shape of the object [2]. Then, this segmentation-based approach

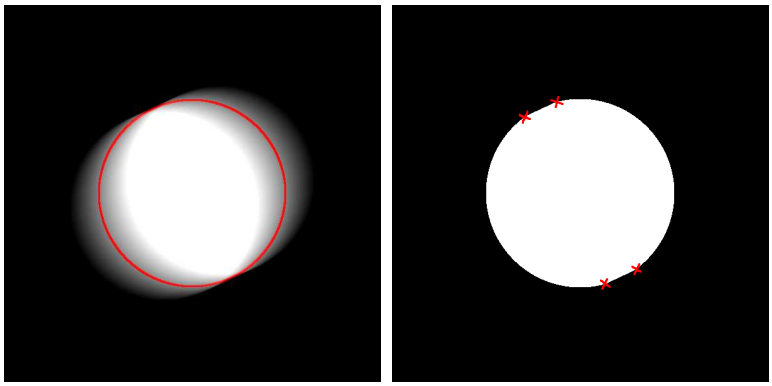

Figure 3.19: Alpha map of a motion blurred circle. The iso-alpha 0.5 lines inside R1 regions represent the apparent contour at half of the exposure interval. On the right, the result of the segmentation of alpha with 0.5 as threshold. Only a small portion of the contour does not correspond to the latent two-tone alpha map.

could only work when the R1 regions contain most of the object's contour at the middle of the exposure interval. This drawback prevents to consistently use this approach for motion blur parameters estimation. As an example, Figure 3.20 shows a blurred square, and the resulting segmentation strongly differing from the real shape of the object. Future work may involve extrapolating the edges in the 'safe' R1 regions to consistently obtain a usable estimate of α_0.

In what follows, a different approach for estimating the motion blur PSF parameters is introduced. The *Radon transform* of the alpha map Laplacian is used to estimate the blur direction.

Let $(s,\ \phi)$ be the parameters of a 2D line, respectively distance from the origin and angle of the normal vector. The Radon transform of a 2D function $f(x, y)$ is then defined as:

$$R(s,\phi)\,[f(x,y)] = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} f(x,y)\ \delta(r - x\cos\phi - y\sin\phi)\ dxdy \quad (3.22)$$

This corresponds to the projection of the function (i.e. the image) along a line. Such line is tied to pass through a reference point (for example, the

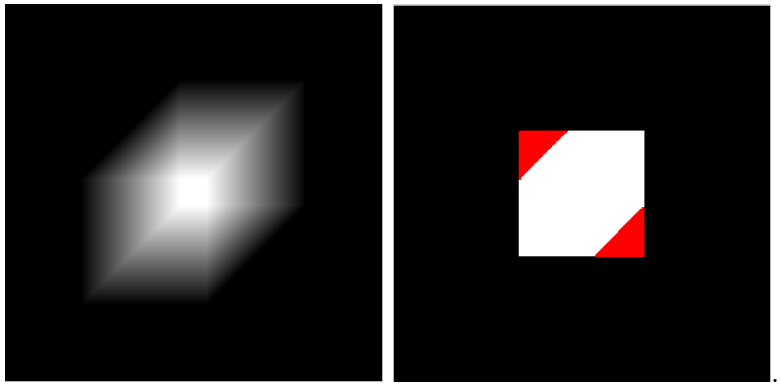

Figure 3.20: Alpha map of a motion blurred square. In this case, the two shaded corners are lost when applying a threshold to the alpha map. In fact, these missing parts belong to a R2 region

central pixel) and its orientation varies in the interval $\phi \in [0, \pi]$. For any fixed value ϕ, the image intensity is projected on the respective line.

The Laplacian of the alpha map, $\nabla^2\alpha$, provides some useful information about the motion of the contour. Recalling again the results in [2], the $|\nabla^2\alpha|$ of motion blurred objects contains edges that represent either the apparent contour at the extremes of the exposure interval, or the envelope of such contour during the motion. When the object motion is rectilinear, the envelope of the apparent contour is a line parallel to the direction of the motion. This suggests that the Radon transform of $|\nabla^2\alpha\ |$ has a maximum when ϕ is orthogonal to the motion direction θ. An issue with this approach arises when the apparent contour contains many straight edges itself, causing multiple peaks in the Radon transform; in this case, all potential directions have to be tested at a later stage to find the best fit. The estimated blur direction θ can be used to estimate l. Once θ is available, it is in fact possible to define a target function as the correlation between the $|\nabla^2\alpha(x, y) = L(x, y)|$ and

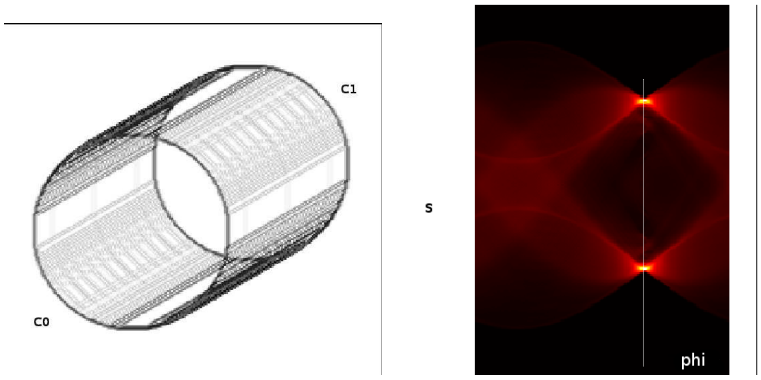

Figure 3.21: Radon transform of the Laplacian of a motion blurred circle. The horizontal axis represents angles, the vertical is the offset from the chosen reference point. Two peaks appear at an angle orthogonal to the motion direction.

itself after a variable non-null translation along such direction:

$$J(l) = \sum_{x,y=-\infty}^{+\infty} L(x,y)\ L(x + l\cos(\theta), y + l\sin\theta). \tag{3.23}$$

The correlation has a peak when the two apparent contours (c_0 and c_1 in Figure 3.22) overlap after the translation [37].

However, test results showed that estimating the blur extent using this approach is quite demanding in terms of processing resources. It is also likely to fail in absence of a sophisticated pre-processing of the contours: it would be much more robust if c_0 and c_1 could be isolated from the other edges prior to the optimization of the cost function (3.23).

In Jia, 2007 [18] a method is proposed to identify at least an upper bound to the size of the PSF. The topmost pixels with $\alpha = 0$ and $\alpha = 1$ are considered. Then the height of PSF cannot exceed their distance M_y along the vertical direction. The same applies to the rightmost pixels with $\alpha = 0$ and $\alpha = 1$. Their distance along the horizontal direction, M_x, provides an upper bound to the maximum width of the PSF. In [18] this procedure is just needed to find an upper bound to the support of the PSF. In this work,

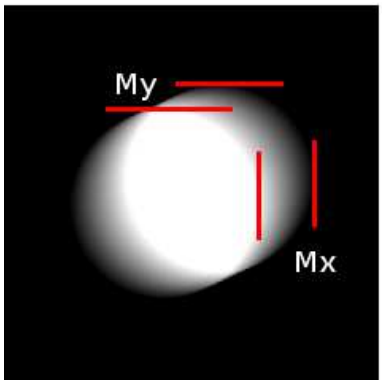

Figure 3.22: Finding an upper bound to the size of the PSF

since a rectilinear motion blur PSF is assumed, it is also possible to derive the extent of the blur by using the Pythagorean theorem:

$$l \leq \sqrt{M_x^2 + M_y^2}$$

The estimated parameters θ and l are now used to define the PSF and restore the image, as seen in the next section.

3.4 Image restoration and further refinement

Once the PSF has been computed, the image can be restored using classical deconvolution techniques, that will now be briefly introduced. In Fourier domain, image I is the result of filtering the latent image I_0 with the PSF, adding noise V. V and I_0 are considered independent.

$$I = I_0 \cdot H + V \tag{3.24}$$

The most simple deconvolution approach is inverse filtering. It is implemented as a simple element-wise division of the degraded image over the PSF, in Fourier domain.

$$\hat{I_0} = \frac{I}{H}$$

In practice, this approach is not reliable at all since it tends to dramatically amplify noise and PSF parameter errors. Even when using synthetically generated alpha maps and exact PSF models, the quantization error alone becomes critical for anything less than double precision maps.

This obviously leads to adopt more robust approaches that offer a better control over the error. Wiener deconvolution is based on least squares optimization of the expectation of the residual, under the assumption that signal and noise are independent [15].

Restoration is performed in Fourier domain using a suitable filter W:

$$\hat{I}_0 = I \cdot W \tag{3.25}$$

where W is computed in order to minimize the expectation of the error

$$E\left[|I_0 - \hat{I}_0|^2\right].$$

The solution of the least square problem is given by

$$W = \frac{H^* S}{|H|^2 S + N} \tag{3.26}$$

where $S = E\,|I_0|^2$ and $N = E\,|V|^2$ are the mean power spectral densities of the latent signal I_0 and the noise. In most cases, these quantities are not known, however Equation (3.26) can be rewritten in terms of their ratio:

$$W = \frac{H^*}{|H|^2 + k} \tag{3.27}$$

Where k is N/S, the inverse of the *signal to noise* ratio. The Wiener deconvolution, in the sense of Equation (3.27), was chosen because of its good performance and simplicity. Another well known technique is Richardson-Lucy iterative algorithm, which is based on maximum likelihood estimation [31, 26].

Since the PSF parameters are available, it is now possible to reconstruct both the alpha map and the color map of the blurred object. The choice

of the regularization parameter has to be empirically made. Typical values range from 10^{-4} to 10^{-2}, depending on factors including the amount of noise, and errors in the PSF estimation. More details concerning issues with real camera images are found in Section 3.5.

3.4.1 Iterative parameter refinement

In what follows we illustrate an optimization procedure that has been devised to improve the PSF estimation. It is likely that the reconstructed alpha map is *not* a two-tone black and white image as expected. This prior knowledge on α_0 can be exploited in an iterative procedure to refine the estimated parameters. This procedure has been inspired by a more general and complex restoration method for two-tone images introduced by Li and Lii, 2002 [24]. In their work, the latent image z that has to be restored is composed by two unknown grey levels, β_1 and β_2. The observed image x is modeled as:

$$x = z \circledast h + \eta, \tag{3.28}$$

where h denotes again the unknown PSF and η is Gaussian white noise. The main goal is finding a filter f such that

$$\hat{z} = f \circledast x$$

minimizing the following cost function:

$$J = \sum_{\forall i} (\hat{z}_i - \beta_1)(\hat{z}_i - \beta_2) \tag{3.29}$$

Their algorithm alternates the estimation of β_1 and β_2 with that of f. Finally $\hat{z}$ is segmented in two regions according to the computed levels.

To refine the estimated blur parameters, we derived a simplified algorithm from the aforementioned approach. For alpha maps the two tones of α_0 are already known to be either 0 or 1. The cost function is defined as

follows, in terms of the parameter vector $\vec{\Theta}$:

$$\begin{aligned} \hat{\alpha}_0 &= \alpha \oslash h(\vec{\Theta}) \\ J(\vec{\Theta}) &= \sum_{\forall i} min\{|\hat{\alpha}_{0_i}|, |\hat{\alpha}_{0_i} - 1|\} \end{aligned} \tag{3.30}$$

Where $\vec{\Theta}$ is a vector containing the estimation of the PSF parameters, and $\oslash$ denotes deconvolution. The cost function $J(\vec{\Theta})$, which only depends on

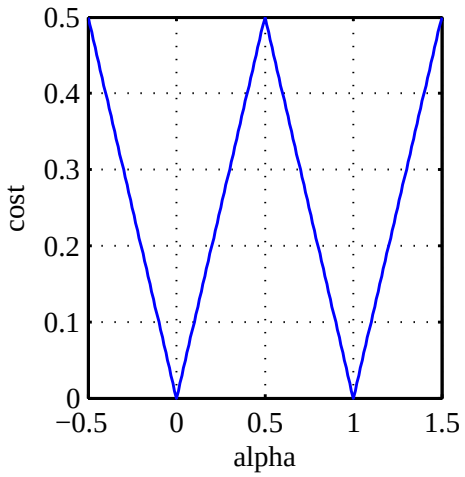

Figure 3.23: The cost function is shaped like a 'W' to penalize pixels that differ from either zero or one

the PSF parameter and the regularization term for the Wiener deconvolution, is minimized using available non-linear optimization algorithms. The starting values for the unknown parameters are naturally the ones estimated as described in Section 3.3.

We also tried the minimization of the same cost function on a non-parametric PSF. The advantage of using a non-parametric PSF is the ability to handle more complicated blur cases. However this requires the definition of a variable for each pixel in the blur kernel: this brute force approach did not perform well in tests, even with a PSF as small as 15×15.

3.5 Dealing with real images

In Section 3.2 a simplified optical system has been proposed to clarify the image formation process. In real camera images, things eventually differ from this ideal situation, because of a wide variety of factors. These cause errors and uncertainty in the PSF, influencing the overall result of the restoration phase. This section covers some of these factors and what can be done to limit their impact.

First, real images are always affected by a certain amount of noise. Its source widely varies: for example, noise originates in electronic circuitry, by quantization of the intensity levels and data compression. In practice, the impact of the noise on the proposed approach is twofold. First, noise may prevent matting techniques from computing a reliable matte; secondly, the restoration phase will need higher regularization coefficients for Wiener deconvolution. This keeps artifacts such as ringing to a minimum, at the cost of a reduced deblurring effect. Section 3.5.1 illustrates a general strategy to counter noise in the alpha map.

Another factor that impacts the shape of the PSF is the gamma correction that imaging devices use to convert the acquired image to the standard color space. Power-law transformation of the image changes the profile of the alpha map which in turns affects the estimation of the PSF. When raw images are available, it is possible to work directly on the data read from the sensors. Otherwise, there exist approaches to identify and reverse such issues [9, 8].

Other issues arise because modeling the blur as a linear shift-invariant system is a fairly strict assumption. For example, the size of the blurring circle depends on the distance Z from the object focal plane. When such distance is not the same for all points of an object, then the PSF is not shift-invariant.

Shape of the PSF Another assumption was made on the PSF being a uniform disk. Cameras are equipped with a diaphragm to adjust the

Figure 3.24: Photograph of an out of focus ballpoint pen. On the left at full aperture, the spot of light on the tip appears as a circle. On the right, aperture is reduced and the six blades of the diaphragm are visible in the PSF.

aperture of the lenses. This may change the shape of the PSF, therefore invalidating the disk assumption. Diaphragm shape varies according to the number of blades and aperture settings: at full aperture, the PSF can be considered circular. As the lenses are increasingly 'stopped down', the PSF takes on the shape of a polygon, according to the number of blades forming the diaphragm. In [20] the image restoration problem is solved using a triangular PSF to mimic the shape of a three-bladed diaphragm.

Vignetting is another effect that can influence the PSF. Vignetting appears as image intensity dropping toward the peripheral region. There are different causes of vignetting. *Mechanical vignetting* is due to physical obstruction of the lenses by hoods or other parts. This also affects the shape of the PSF, especially at full diaphragm aperture. Figure 3.25 shows how mechanical vignetting can be directly seen in the blurred part of the background. Small pointy highlights in the foliage have a round shape in the middle of the image, which becomes oval at the peripheral. Loosely speaking

this is also known as the *cat's eye* effect. Naturally this also invalidates the shift-invariant PSF assumption, especially at large viewing angles. Another

Figure 3.25: Note the oval shape of the spots on the background.

kind of vignetting is *natural vignetting*, due to the angle θ at which the light hits the photosensitive medium. The intensity falls proportionally to $\cos^4 \theta$.

Optical aberrations Even well made and carefully assembled lenses may possess certain inherent aberrations. As an example, *spherical aberration* causes focal variation as the distance from the optical axis increases. Strong aberrations prevent from focusing details in the image, since the PSF is never smaller than the *circle of least confusion*. In out of focus images, spherical aberration results in a non-uniform PSF distribution. This issue may be corrected combining different lenses that cancel out the effect. Certain photographic equipment even exploit a low amount of spherical aberration for artistic purposes, like smoothing either background or foreground blur, or producing a *soft focus* effect.

There are also cases where even the most advanced matting techniques fail to provide a reliable matting, because of color ambiguity, numerical issues, or noise. This is naturally a strong limitation that must be consid-

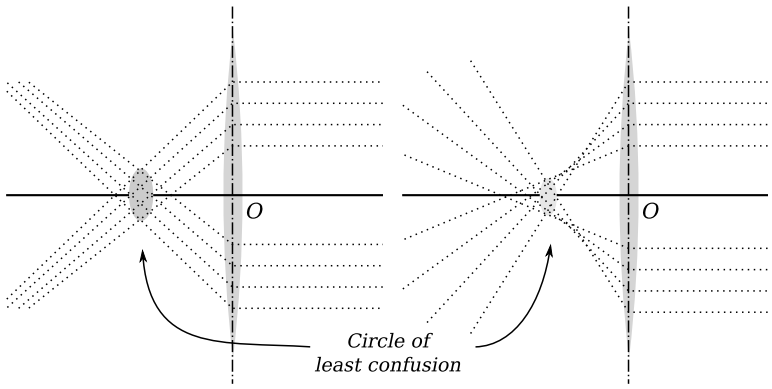

Figure 3.26: Two opposite cases of spherical aberration

ered when making assumptions on the alpha and color maps in real camera images.

Compensating for all these issues is out of the scope of this work, since most of them depend on a specific combination of camera model, optics and shooting conditions. Nevertheless, there are some useful tips to deal with generic situations of model uncertainty and noise, covered in the remaining part of this chapter.

3.5.1 Alpha map and noise

When the alpha map is affected by noise, it is likely that PSF parameters cannot be reasonably identified. This is a common problem in deconvolution-based restoration since blur inversion typically amplifies the noise.

In Section 3.3 a disk or Gaussian blur PSF was estimated as follows:

$$H = A \cdot \frac{\hat{A}_0^*}{|\hat{A}_0|^2 + k} \tag{3.31}$$

A low-pass Gaussian filter G can be applied to both terms:

$$G \cdot PSF = G \cdot A \cdot \frac{\hat{A}_0^*}{|\hat{A}_0|^2 + k} \tag{3.32}$$

Pre-filtering the alpha map results in smoothing the estimation of the PSF as well. This low-pass filtering cuts off the high frequency components of

the noise in the alpha map. These are source of many numerical pitfalls when performing deconvolution, since they tend to be amplified until they completely suppress the signal. It should be noted from Equation (3.32) that the estimated PSF is now convoluted with the same filter G. The choice of G has to take into account both the noise power and the size of the PSF: it has to be strong enough to neutralize the noise but not too much, otherwise the estimation would still be useless to identify any PSF parameter at all.

To clarify the effect of alpha map pre-filtering, a simple example is now proposed. A synthetic alpha matte was generated convoluting the image of a white circle with a 30 pixels radius PSF. Then some Gaussian white noise $\sim N(0, 10^{-3})$ was added to it. The regularization term k used in the Wiener deconvolution was empirically set to 10^{-3} as well. Figure 3.27 shows the results of the first PSF estimation phase in three different pre-filtering scenarios. Testing shows that pre-filtering the alpha map dramatically increases the quality of the PSF estimation, reducing the negative effects of the noise. However, the filter parameters are tuned empirically at the moment. Future work may involve automatic dimensioning of G, according to the amount of noise detected in the available data.

3.5.2 Alpha map post-filtering

Once all parameters are set, the restored alpha and color images may still contain artifacts. Especially in real images, inaccuracy in the matte extraction, uncertainty of the blur model, noise and machine precision do not allow a perfect restoration (see Section 3.5).

Consider the image in Figure 3.28, showing a blurred petal and its alpha map. One of the most annoying artifacts found in deconvolution is the so called *ringing*. It is characterized by wave-like ripples propagating from hard edges, as seen in Figure 3.29. Often, and more frequently in presence of noise, ringing effects appear in the restored images. Another cause of ringing is the

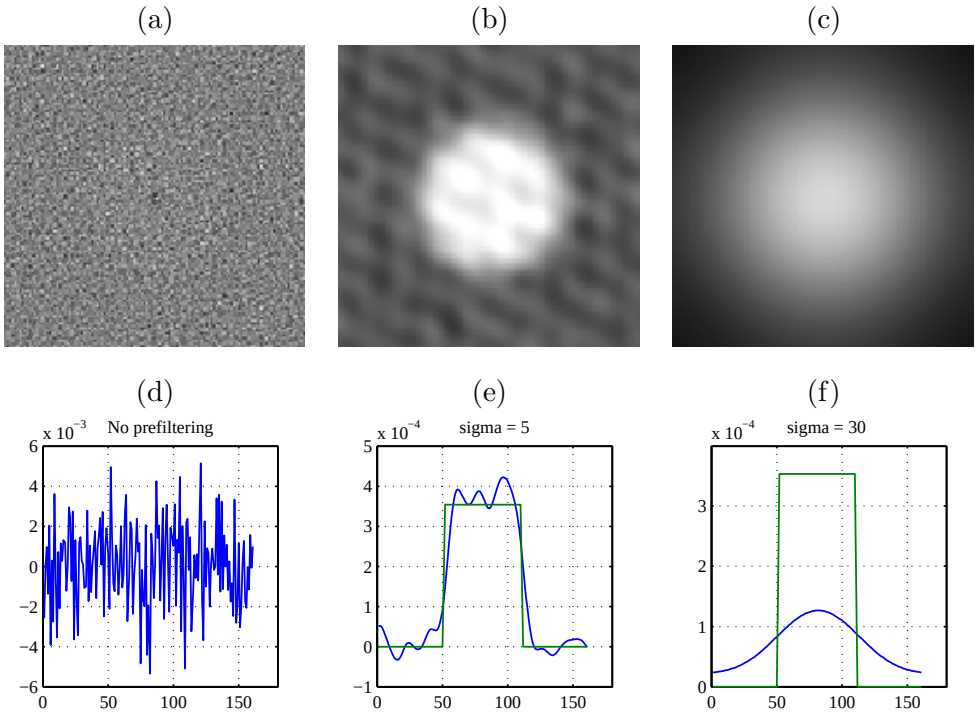

Figure 3.27: Examples of PSF estimation from a synthetic alpha map with Additive Gaussian White Noise, 0 mean and variance .001. (a) PSF from unfiltered alpha map. (b) PSF from filtered alpha map. (c) Here the filter was so strong that the PSF is unrecognizable. (d), (e) and (f) show the estimated PSF profile along the central row. In (d), the noise completely hides the underlying signal. In (e), the PSF correctly appears as a disk, with the edges smoothed according to Equation (3.32)*; the green profile stands for ground truth PSF data. Finally (f) shows what happens when the chosen pre-filtering blur is too strong: here the PSF cannot be recognized anymore.*

Figure 3.28: Detail of a blurred flower petal and its corresponding alpha map.

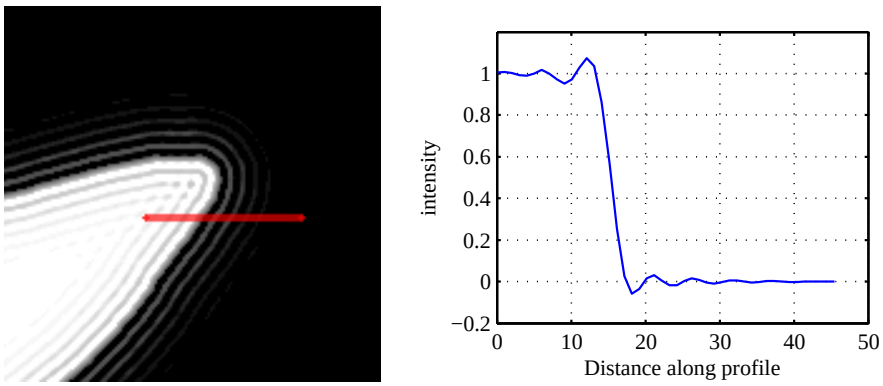

Figure 3.29: An example of ringing artifacts in the reconstructed alpha map. The image intensity levels were modified to make even the smallest ripples visible. The plot on the right shows the actual, unmodified, intensity values along the indicated section.

Gibbs' phenomenon, because the edges of the reconstructed image present a jump discontinuity. This occurs when trying to approximate a *discontinuous* function with a finite sum of continuous sine and cosine waves; this causes the reconstructed edges to overshoot, producing artifacts in the results.

Naturally, these unwanted ripples strongly affect the quality of the final composite image. To improve the overall result, some post-processing on the alpha map can be applied. The following post-filtering procedure can be applied to the reconstructed alpha map.

Since a two tone alpha map is expected as a result from the restoration phase, it is reasonable to apply again a segmentation threshold to it, in order to eliminate ringing artifacts. They will still affect the color map, but will not spill outside the object's boundary anymore. Especially at

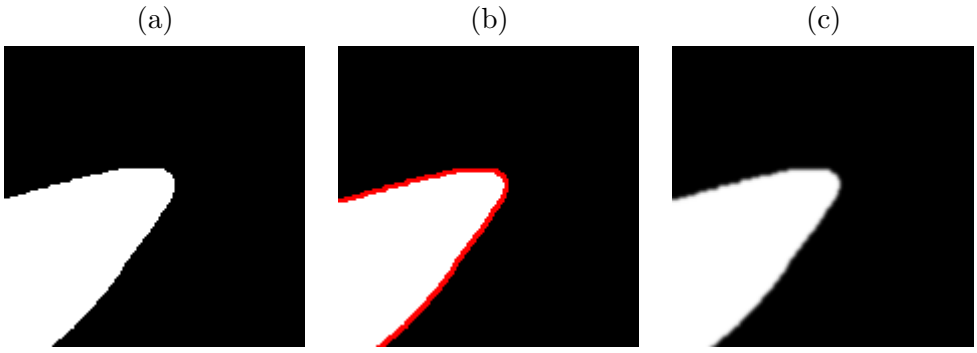

Figure 3.30: (a) Segmentation with 0.5 as threshold. (b) Erosion affecting the highlighted pixels. (c) Smoothing the edges.

low resolutions, using a 'hard' separation between object and background pixels yields poor results because the under-sampling of edges is unavoidable. To blend realistically the object with the background, the alpha map can smoothed by adding some blur. A one-pixel erosion of the alpha is applied prior to this *feathering* phase to avoid picking up black pixels from the reconstructed color map. Figure 3.31 compares the composition on a smooth

background before and after the post-processing operations.

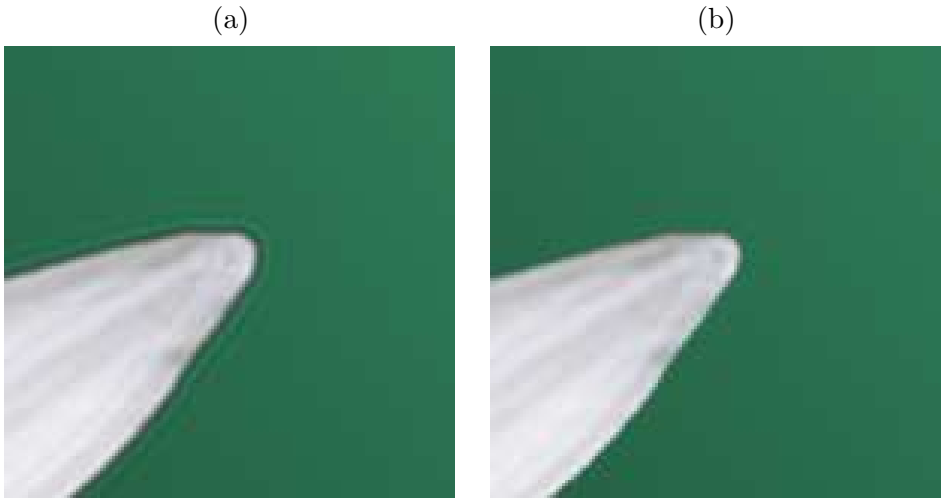

Figure 3.31: (a) Composite image before post-processing is performed. (b) Composite image after segmentation, erosion and feathering of the matte. Ringing artifacts that ruin the background in (a) are cleaned up.

Chapter 4

Implementation aspects

This chapter focuses on the implementation details of the proposed approach. Large but less important source code samples are included in Appendix A, for completeness.

All the code is written and run within *The Mathworks* MATLAB environment.

As in Chapter 3, the contents are subdivided into three main sections: extraction of the matte, PSF estimation and restoration of the image of the object. In the next sections, each phase is discussed in depth, uncovering all the relevant details of the implementation.

Adopting a top-down approach, a flowchart representing an overview of the system is given in Figure 4.1.

4.1 Matting setup

The source code of the natural image matting approach in [23] is publicly available for research purpose at the authors' personal page. Being entirely implemented in Matlab, it is immediately usable to extract alpha, foreground and background maps needed by the following phases.

As already introduced in Section 3.1, computing the matte using the

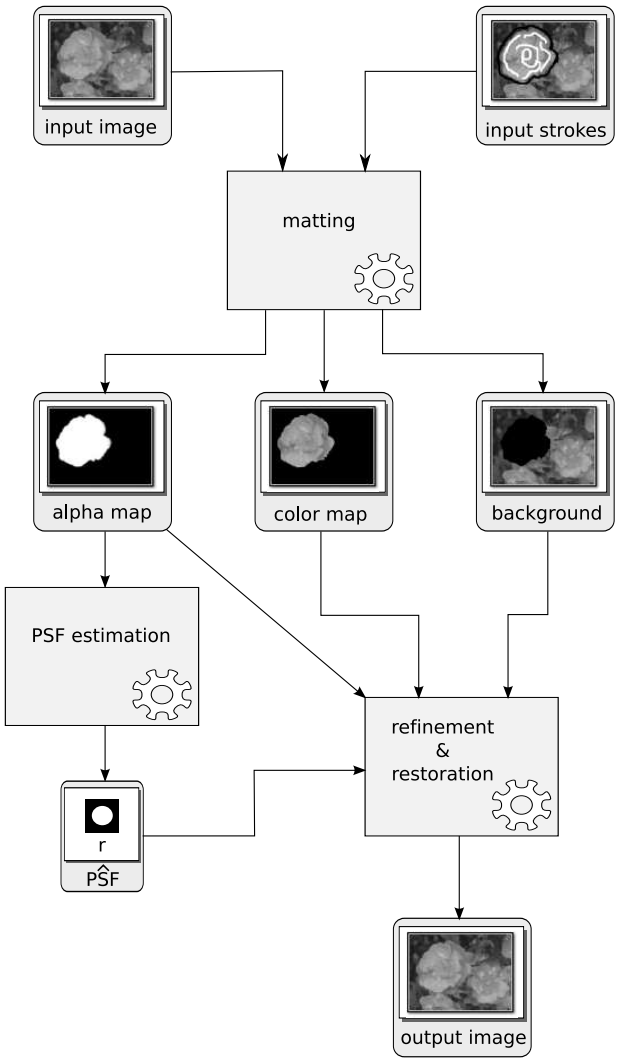

Figure 4.1: Overview of the system

method from [23] requires a starting image and a set of user-defined strokes. White strokes are used to mark foreground regions, black strokes indicate background. Brush feathering must be deactivated to avoid blending the border of the scribbles with the image.

There are also a few optional parameters that can be set to improve the result:

epsilon: the regularization weight defined in Equation (3.3). Typical values range between 10^{-7} and 10^{-5}, depending on the amount of noise and compression in the image. Higher values tend to bias the algorithm toward smoother alpha maps [23].

win_size: size of the window where the *color-line model* is assumed. Default value is 3×3.

The matte is computed using the 'backslash' Matlab operator, on the linear system defined in Equation (3.7). With large images, this can fail because of memory limitations. A multigrid approach is however available to simplify the system and overcome these events. First the image and the scribbles are down-sampled to a lower resolution, where the problem can be solved more quickly. Then, the results are interpolated at finer resolution and the values close enough to zero or one are clamped and considered as a constraint for the upper level. When passing to a finer level, the constrained pixels can be eliminated from the *matting Laplacian*. Moreover, only the windows w_k that contain at least one unconstrained pixels have to be computed, further reducing the required computations. The multiresolution approach can be activated and tuned by setting the following variables:

levels_num: the total number of levels to be used. Default value is 1, meaning multiresolution is not used.

active_levels_num: must be less than or equal to levels_num. If active_levels_num is less than levels_num, then in the finer resolutions alpha is not computed explicitly. Instead the values of the coarser resolution are interpolated.

thr_alpha: threshold on alpha used to determine which pixels will be constrained to either zero or one, when moving to the next resolution level.

When the process is complete, the results are stored in the variables alpha, F and B. These can be immediately used for restoration or stored in an image for later. However, this operation may introduce additional quantization error, due to the conversion from double precision arrays to a specific image format.

4.2 PSF Estimation

As seen in Section 3.5.1, it is useful to pre-filter the alpha map prior to the estimation of the blur filter. This is accomplished by a first median filtering passage to remove possible outliers that are sometimes produced in the matting phase, when the background is not smooth enough. After median filtering with a 5×5 window, Gaussian filtering is finally applied. The blur σ to be used in this phase is defined in the variable pre_filter . A good value to start with is pre_filter $= 1$.

After filtering the alpha map, the PSF estimation phase takes a different path according to the blur type.

4.2.1 Defocus blur parameter estimation

In case of defocus blur, the alpha map is segmented applying a threshold. As discussed in Section 3.3.1, the threshold level used to estimate the latent two-tone alpha map is set to 0.5.

Once the estimate of $\hat{\alpha}_0$ is available, the PSF is roughly computed by

using classical deconvolution techniques. The Wiener filter method is used, implemented by the deblur function.

```
psf_hat = deblur(alphamap, alpha_zero_hat, .01);
```

The implementation is reported in Appendix A.2. Empirical tests show that a regularization term of 10^{-2} is a good guess to start with, since noise suppression is more critical than accuracy in this preliminary phase. For synthetic images, when accurate and noiseless alpha maps are available, this can be lowered by one order of magnitude or two. However, there is no reason to finely tune this parameter now, since it will be automatically done in the following iterative refinement phase.

Periodicity of the image and the Discrete Fourier Transform One fact to recall when moving to frequency domain by DFT is the assumption that the image is periodic. When objects intersect the image borders, some unexpected results will occur when estimating the PSF. This happens because of the jump discontinuity that takes place at the borders when the image is periodically tiled. Moreover, when restoring the image under such conditions, a strong *ringing* effect propagates from the image borders. Since the out of focus blur PSF is assumed to be shift-invariant and rotationally symmetric, the image can be mirrored along both the vertical and the horizontal directions without altering it. The resulting image is four times larger, but the opposite borders now can be seamlessly tiled, eliminating the unwanted artifacts. Figure 4.2 shows a practical example where the estimation is performed before and after the addition of the mirrored replicas. Naturally, this rather inelegant expedient does not work with motion blur, since the PSF is not invariant to flipping and mirroring. After the deconvolution, the estimation psf_hat matches the size of the image. However, the information needed to estimate a parametric PSF is concentrated in a small central region. To isolate the meaningful part of the estimation, the follow-

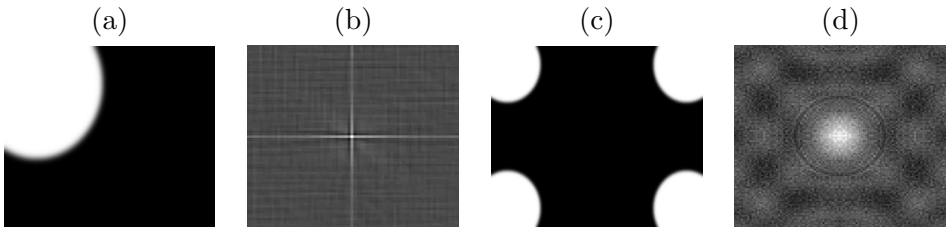

Figure 4.2: Adding mirrored replicas help dealing with the situation when the object intersects the border of the image. (a) and (b): alpha map and the incorrect PSF estimation due to edge jumps. (c) the same alpha map with its mirrored clones. (d) the Gaussian PSF is now identifiable.

ing heuristic rule is proposed: let C be the number of PSF samples in psf_hat that are greater than half the maximum. C approximately corresponds to the area of a circle having a radius of $\rho \approx \sqrt{C/\pi}$. The support size of the PSF is then set as the minimum between 50 and 6ρ; reducing the size of the PSF support has the benefits of discarding regions containing irrelevant data, and reducing the time required to estimate the parameter. The value ρ obtained in this phase is also used as starting value for the parameter identification.

To identify the PSF parameter, the built-in fminsearch Matlab function is used. It performs the unconstrained non-linear minimization of a function of multiple variables. It is based on the *downhill simplex method* by Nelder and Mead [28]. In Matlab, it can be invoked as follows:

```
[x, fval] = fminsearch(@(x) func(x, ...), x0)
```

The objective function takes the independent variable vector x and possibly a number of fixed parameters. Additional options can be provided to fine tune tolerance, maximum iterations and maximum function evaluations.

To estimate the PSF parameter, the following objective function is defined, assuming the blur is Gaussian:

```
function [sigma] = getSigma(psf, start_sigma, norm)
 (...)
 support = size(psf,1);
 % trying different norms for the error function.
 errorfun_square = ...
   @(s,l,p) sum(sum( ((p - fspecial('gaussian', l, s) )).^2));
 errorfun_minmax = ...
   @(s,l,p) max(max(p - fspecial('gaussian', l, s) ));

 sigma_square = ...
    fminsearch(@(s)errorfun_square(s, support, psf), start_sigma);
 sigma_minmax = ...
    fminsearch(@(s)errorfun_minmax(s, support, psf), start_sigma);
 (...)
```

Note that is is possible to choose to minimize either the L_2 or the L_∞ of the error.

In case of disk blur, the objective function, which is very similar, is reported in Appendix A.

4.2.2 Motion blur parameter estimation

Up to now, only out of focus blur has been covered. In case of motion blur, the approach slightly differs. First the direction θ is found by exploiting the Radon transform, as seen in Section 3.3.1. The Laplacian of the alpha map is numerically estimated, then its Radon transform for a given direction ϕ is given by the Matlab command

```
r = radon(l_alpha, phi)
```

Maximizing the Radon transform for phi is just matter of defining a cost function for fminsearch:

```
radon_objfunc = @(theta, alpha) -max(radon(alpha, theta));
```

Note that the blur direction is orthogonal to the result, therefore $\theta = 90 + \phi$.

To estimate the motion blur extent, the method proposed in [18] is used. The following code finds the upper bound to the vertical PSF size, looking for the two topmost pixels with $\alpha > 0$ and $\alpha = 1$ respectively:

```
function [y] = upper_bound(alpha)
sx = size(alpha,1);
t = .01;
for i = 1:sx
    if max(alpha(i,:)) > t
        break;
    end
end
for j = i:sx
     if max(alpha(j,:)) > 1-t
          break;
     end
end
y = j-i;
```

The threshold is required to neutralize the effect of noise. The same procedure is repeated for the horizontal direction. Since, differently from [18], only rectilinear motion is considered, the support size of the PSF, $(M_x,\ M_y)$ allows to estimate the motion length:

```
l = sqrt(Mx^2 + My^2)
```

This concludes the estimation of the parameters from the alpha map, in case of either out of focus or rectilinear uniform motion blur. Next phase consists in the iterative process of restoration and further refinement of the estimated parameter, with additional alpha map post-processing to improve the overall result.

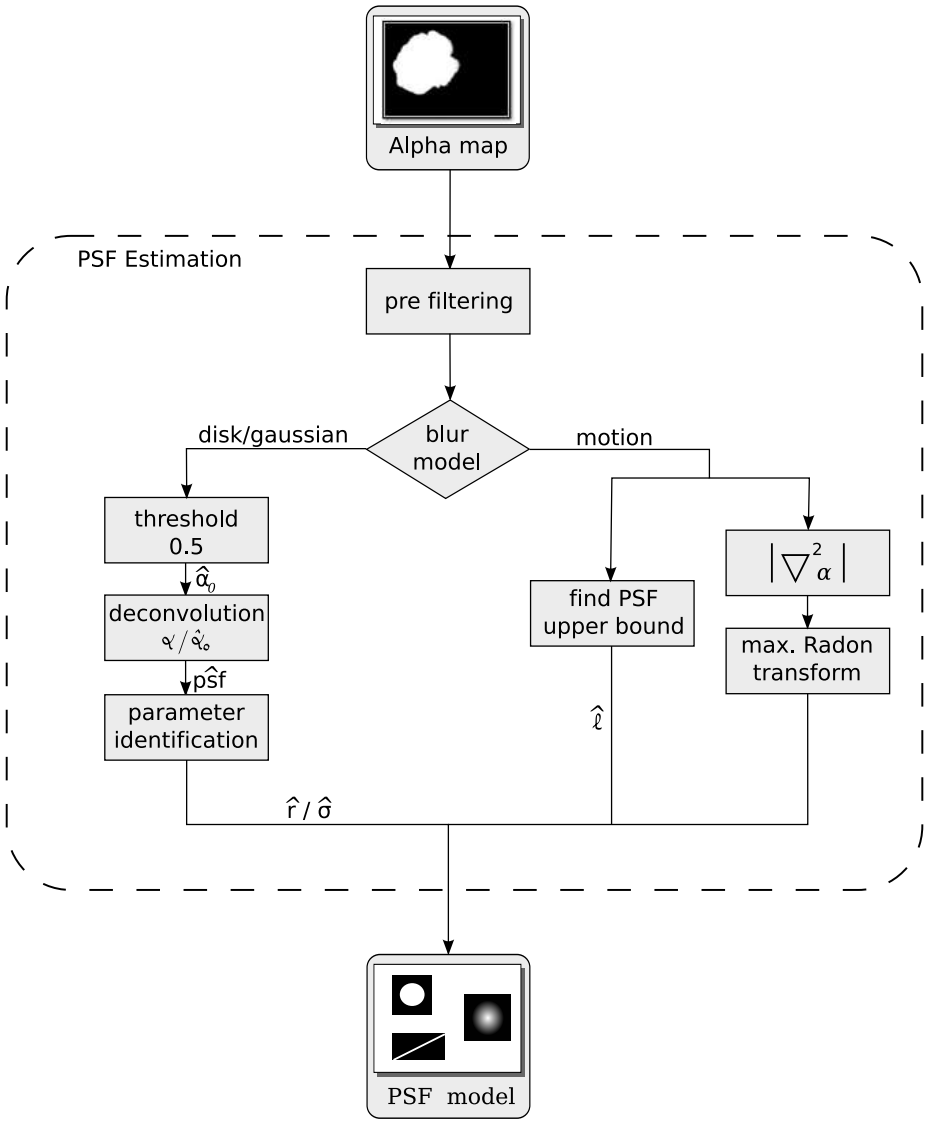

Figure 4.3: Detailed virew of the PSF estimation block

4.3 PSF refinement and restoration

The core issue in this phase consists in iteratively refining the PSF parameter and the deconvolution regularization term. To assert the optimality of the solution a cost function is defined according to the selected blur type. Here follows the cost function for disk blur:

```
% used by fminsearch to refine the parameters
function [J] = disk_func(param, k, alpha)
  alpha_post = deblur(alpha, fspecial('disk',(param(1))),k);
  J = sum(sum( min(abs(alpha_post),abs(alpha_post -1))));
```

The other two cost functions, gaussian_func and motion_func are defined in a similar way, and are reported in Appendix A. The cost functions are designed to penalize pixels that are not zero or one. This way the optimization procedure is lead towards the best compromise between deblurring effectiveness and reduction of ringing artifacts.

The optimization is an iterative procedure where the refinement of the parameters alternates with the refinement of the regularization term k. The same cost function defined above is used in both cases, just by changing the independent variable.

Once the parameters are set, the restored alpha map can be post-processed to improve the look of the final composition, as discussed in Section 3.5.2. The structuring element to be used with the built-in function imerode is

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

to reduce the reconstructed silhouette of the object by one pixel. Then a mild Gaussian blur is added. Note that this post-processing phase affects the alpha map only, while the color map is left untouched.

Finally, the composition can be done either on the original background or on a new one.

```
for i = 1:3
  out = rest_alpha .* rest_color(:,:,i) + ...
   (1 - rest_alpha) .* background(:,:,i);
end
```

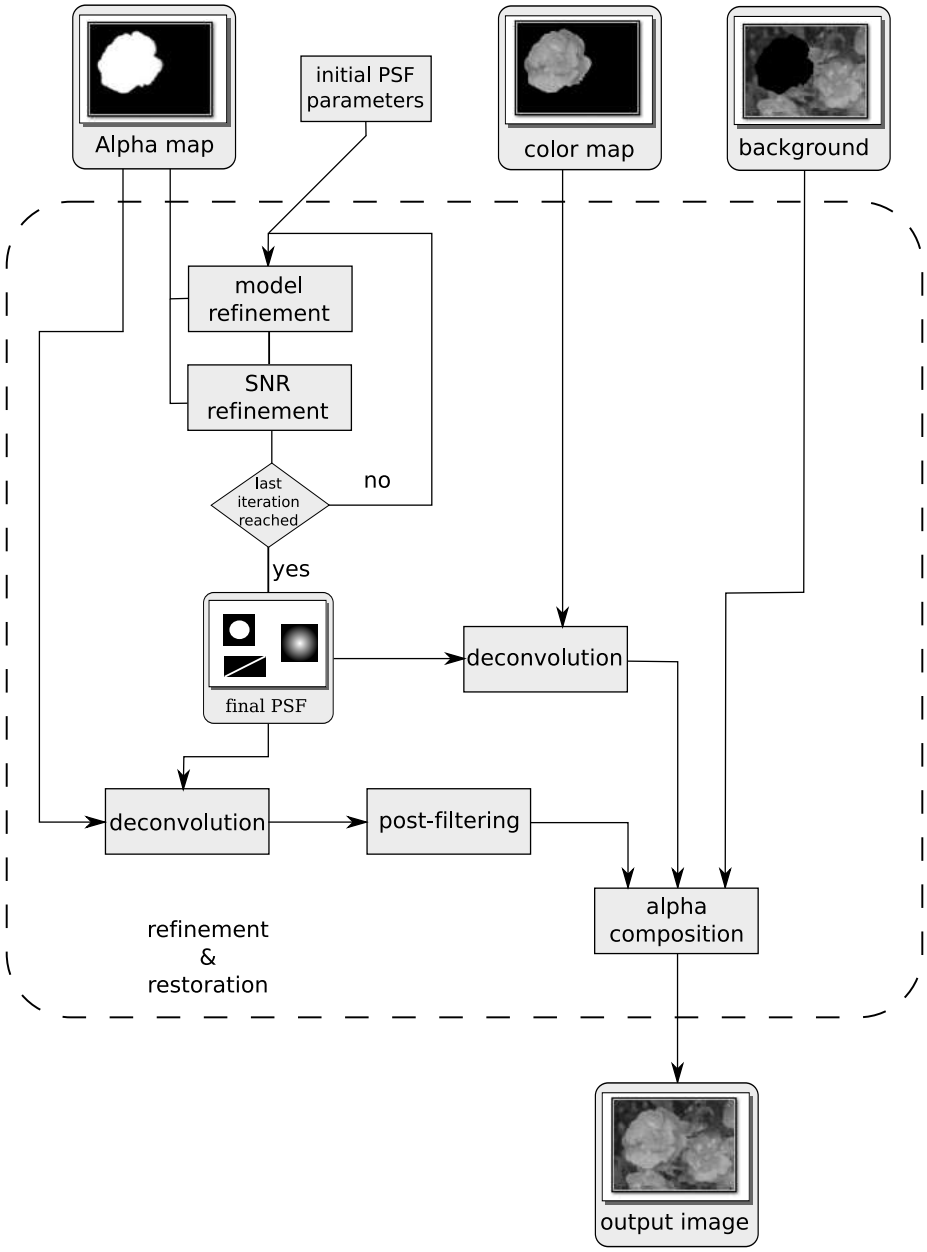

Figure 4.4: Flowchart of the restoration phase

Chapter 5

Experimental results

This chapter is aimed at showing how the proposed approach performs in practical situations. The first set of experiments takes place under ideal conditions, where the blur PSF, alpha map and color map are perfectly known. Each type of blur is tested with a certain range of parameters and results are presented and commented. Some tests are also presented with additive noise, to show its tolerance to different noise levels.

Section 5.2 finally covers experiments with camera images, to stress the algorithm under practical conditions.

5.1 Tests with synthetic images

Tests in this section are conducted under strictly controlled and known conditions. Before dealing with the actual experiments, the procedure used to generate synthetic test cases is shown in what follows.

The starting data of each experiment are the latent black and white alpha map, and the sharp color image of the object. These also represent ground truth for later comparison. Then a PSF is chosen according to blur type and parameter, and the starting images are convoluted with it. Therefore the theoretical preconditions of the algorithm are fully met.

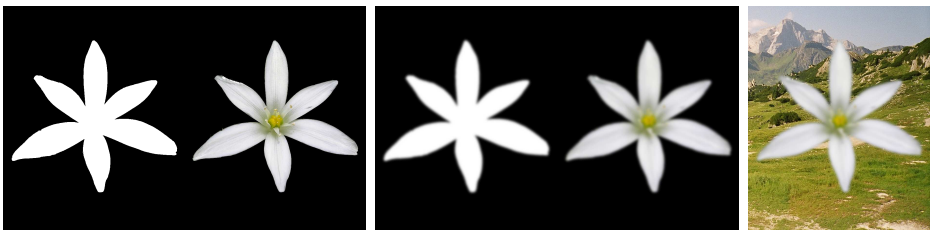

Figure 5.1: Synthetic test with Gaussian blur. From left to right: sharp alpha and color maps, blurred alpha and color maps, combined image.

Note that one cannot directly compose the blurred color map on a background, since its color values are already combined with black background pixels, according to the correspondent alpha value. To obtain a correct composition, the actual color values are preprocessed as follows:

```
color_map = color_map./(alpha_map + epsilon)
```

where epsilon is a small number to avoid division by zero issues. This is not needed when the color map is computed with matting techniques. However, in synthetic tests, the matting phase is skipped to ensure that no unforeseeable errors or biases are introduced in the starting data, because of the intrinsic difficulty of the matting problem.

5.1.1 Gaussian and disk blur tests

The first synthetic test image is generated with a Gaussian blur kernel of parameter $\sigma = 3$, without the addition of noise. No pre-filtering of the alpha map is performed, since it is not essential for noiseless images. The standard threshold level of 0.5 is used, and the starting value of the regularization parameter (the k in Wiener deconvolution, Equation (3.27)) is set to $5 \cdot 10^{-3}$. Figure 5.2 shows the outcome of the initial estimation of the PSF, where the typical Gaussian bell shape is perfectly recognizable, observing its profile along the central row. The parameter estimation obtained by minimizing

the L_2 norm of the error is $\hat{\sigma} = 2.9977$.

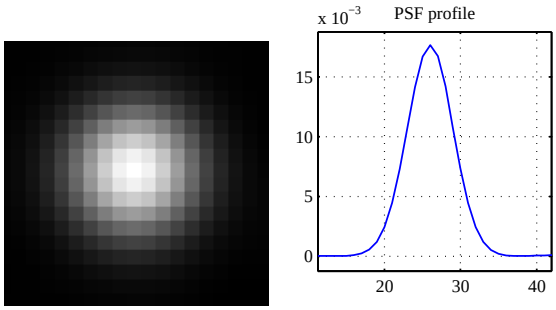

Figure 5.2: Estimated PSF on the left. The plot on the right shows the profile along the central row.

This is indeed close to the true value of $\sigma = 3$. Figure 5.3 compares the profile of the alpha map before and after deconvolution using the current estimation of the PSF and the predefined value of k.

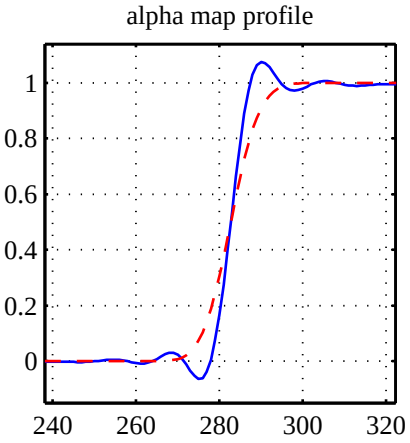

Figure 5.3: Profile of the reconstructed alpha map using the initial estimation parameters. The dashed line represents the alpha map before restoration.

Note that ringing artifacts are in place, even if the parameter is indeed very close to the ground truth data. This cannot be avoided most of the times because of the bad conditioning of the deconvolution; this fact also

has to be kept in mind to interpret the results of the following refinement phase, shown in Figure 5.4.

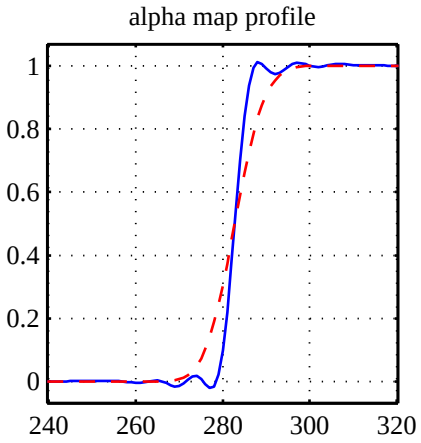

Figure 5.4: Profile of the reconstructed alpha map using the refined parameters. The dashed line represents the alpha map before restoration.

The refinement phase was limited to 3 iterations, setting the maximum number of function evaluation for fminsearch to 20. The resulting parameters are $\hat{\sigma} = 2.8019$ and $k = 2.58 \cdot 10^{-6}$. This estimation appears to be worse than the initial value, with a relative error raising to about 6.7%, however the profile of the reconstructed alpha map does contain less annoying artifacts (compare Figure 5.3 and 5.4).

Finally Figure 5.5 proposes a qualitative view of the results, showing a couple of relevant details in the image.

The proposed test has been repeated varying the blur amount and model. Results are summarized in the following tables.

In these tests conducted under ideal conditions, without any noise added to the data, the initial parameter estimation from the alpha map is very accurate. This suggests that the errors introduced by blindly choosing 0.5 as global threshold level are hardly significant, at least in this case where the silhouette of the object contains a variety of convex and concave edges.

Figure 5.5: Detailed view of results from the proposed test. First column shows the starting image, central column contains the result from our approach, and the last one presents ground truth data.

Real σ	First $\hat{\sigma}$	Refined $\hat{\sigma}$	Refined NSR
1.5	1.5050	1.3931	6.25e-7
3.0	2.9977	2.8019	2.58e-6
6.0	5.9690	5.4746	9.37e-7
9.0	9.0951	8.2368	7.54e-8

Table 5.1: Results with different Gaussian blur parameters.

It is interesting to note what happens in the following refinement phase: the blur parameter here consistently converges to a smaller value. This is caused by the objective function, that penalizes ringing artifacts in the restoration. Further testing shows that artifacts almost completely disappear only when a perfect PSF is used in combination with an extremely low value of k, close to the machine epsilon. This very particular configuration is never reached by the optimization algorithm, unless the optimal solution is already given as starting value: instead, fminsearch finds an under-corrected PSF, reducing the artifacts at the cost of a weaker deblurring effect.

The same procedure has been used with a disk blur PSF, with slightly different results.

Real r	First $\hat{r}$	Refined $\hat{r}$	Refined NSR
2.0	1.9654	1.9993	5.16e-6
4.0	4.0028	3.9993	9.37e-7
6.0	6.0092	5.9995	6.25e-7
8.0	8.0069	7.9990	6.25e-7

Table 5.2: Results with different disk blur parameters.

In this case, the refinement of the parameter consistently improves the value found in the first phase. This behavior is quite different from the Gaussian blur case, and it can be understood reasoning on the frequency domain representation of the disk PSF, seen in Section 3.2.1. The DFT of a Gaussian is still Gaussian, so it is smooth and has no zeros, while a disk kernel has a specific pattern of zeros along concentric circumferences. Therefore, if the parameter is not accurately identified, zeros do not cancel each other when performing deconvolution, causing very strong artifacts in the restored images. This makes the accuracy of the disk radius much more critical than that of the Gaussian blur parameter.

5.1.2 Motion blur test

Here follows an example with motion blur. In Chapter 3, the strong limitations of this approach with motion blurred objects have been discussed. Therefore, experiments with motion blur are reduced to very simple shapes that do not contain straight edges, as in Figure 5.6, representing a moving circle.

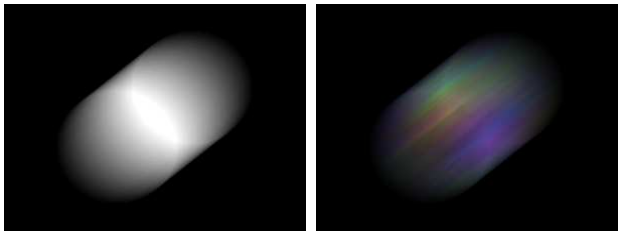

Figure 5.6: Alpha and color map for a motion blur test case having parameters $l = 100$ *px,* $\theta = 40°$.

The blur parameters are $l = 100$ px and $\theta = 40°$; again, noise is not present. The algorithm was initialized, as in the previous cases, without alpha map pre-filtering, and a starting k of 0.005. They were initially identified as $\hat{l} = 100.6500$ and $\hat{\theta} = 40.0189°$, and were respectively refined to 100.0668 and 39.9581°.

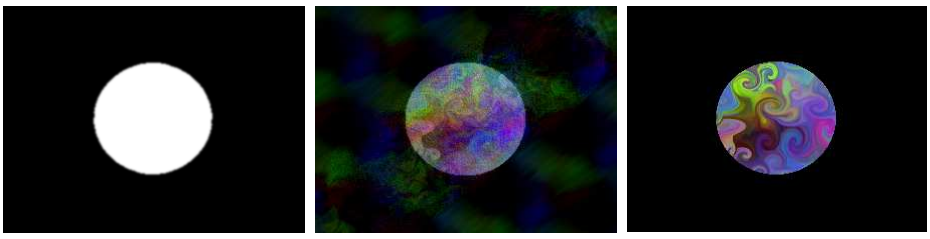

Figure 5.7: Restored maps and ground truth data as a comparison on the far right.

5.1.3 Noise tolerance tests

It is now shown how the algorithm performs in slightly unfavorable conditions: some zero mean Gaussian white noise is now added to alpha and color maps.

Tests are generated with a fixed disk blur of parameter $r = 4$, adding a varying amount of noise. Pre-filtering is now necessary for a proper estimation of the parameter, so it is empirically set to a Gaussian blur with $\sigma = 0.75$. Table 5.3 summarizes the results, while figure 5.8 gives a more qualitative idea of the outcome.

Noise variance	First $\hat{r}$	Refined $\hat{r}$	Refined NSR
0	4.0028	3.9993	9.37e-7
1e-5	4.1805	3.9478	4.72e-3
1e-4	4.1686	3.9265	1.12e-2
1e-3	4.0611	4.0041	2.32e-2
1e-2	4.0347	4.2311	4.39e-2

Table 5.3: Test results with a fixed disk blur and variable amount of noise.

Observe the initial over-estimation of the parameter, which is a side effect of the alpha map pre-filtering phase. The regularization term is automatically raised according to the noise level found in the starting images.

5.1.4 Performance tests

It is also interesting to analyze the processing time required to perform the various phases of the process, in function of the input dimensions.

Test settings were defined according to the following table:

blur type	radius	starting NSR	iterations
disk	4	0.005	3

$\sigma^2 = 10^{-5}$ $\sigma^2 = 10^{-4}$ $\sigma^2 = 10^{-3}$ $\sigma^2 = 10^{-2}$

Figure 5.8: Qualitative view of the restoration with increasing noise variance.

Four $N \times N$ size images are generated and processed 5 times for each value of N. The average values are plotted in Figure 5.9. Testing platform is a Pentium IV 2.4 GHz, with Linux as operating system.

The implementation heavily uses the FFT algorithm, which is optimized for power-of-two sized images; it is strongly advisable to either resize the input image or cut an adequate bounding box around the object.

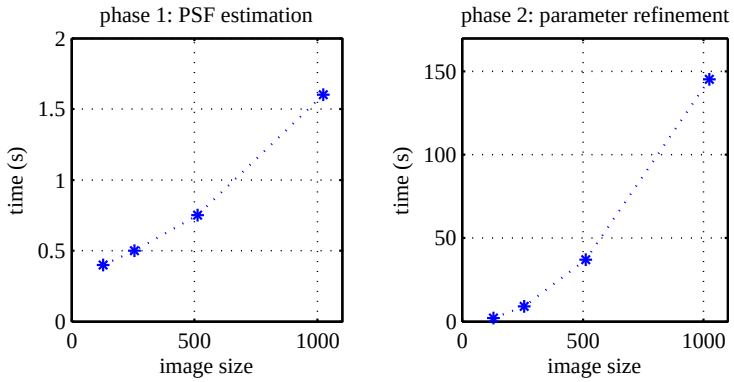

Figure 5.9: Average processing time for $N \times N$ power-of-two sized images.

Also note that the most time consuming part is the refinement of the parameters. This phase can be either simplified or skipped by reducing the number of iterations, making the proposed approach a practical tool for image editing.

5.2 Tests with camera images

Up to now the algorithm has been tested only in controlled conditions, fully meeting the assumptions made on the image degradation model. In this section, the focus is posed on tests with real camera images. As already discussed in Section 3.5, there are many factors that may influence the PSF and are specific to every particular device. These are often difficult to foresee and indeed may raise doubts on the validity of this approach with actual photographs, since the simplified shift-invariant PSF assumption is just an approximation of the real image formation process.

Another element that is difficult to account for is the intrinsic difficulty of the matte extraction process. Since there is no ground truth data available with real images, it is only possible to qualitatively judge the computed matte and see if a PSF can be reasonably identified. However despite all difficulties, results are encouraging.

The first example presented here is the image of a blurred measuring tape. A pre-filtering step of median filtering and Gaussian blur $\sigma = 0.5$ is used to clean up the alpha map. Median filtering was also applied to the alpha map because of a few artifacts, caused by the dark spots on the background being misinterpreted as parts of the object itself. The starting k is set to 0.01. Since the object intersects the border, the image also has to be doubled and filled with mirrored replicas (by setting b_mirror=true). The disk blur model is chosen because it better reflects the image formation process in camera images. Figure 5.10 summarizes the results providing the image before and after the process and a profile of the alpha map before and after the restoration.

Some ringing artifacts propagate from the stronger edges, however the overall quality is still satisfying. Also note how the blurred light circle on the left edge of the metal clip becomes a well definite white dot in the restored version (see Figure 5.10(d)).

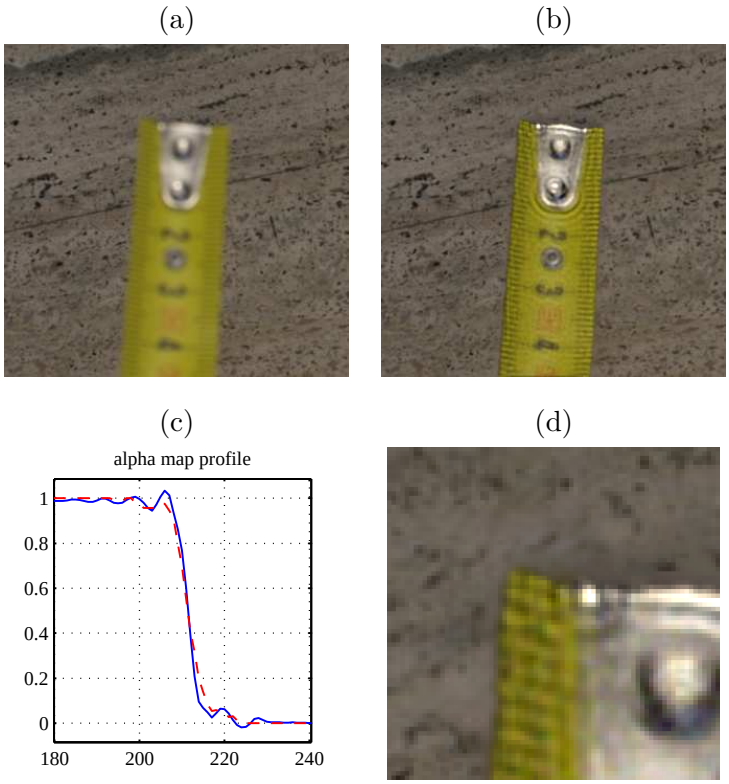

Figure 5.10: Restoration of an out of focus measuring tape. (a) Starting image. (b) Restored image. (c) Comparison of the alpha map profile before and after the restoration, along the central row of the image. (d) Magnified detail showing the restoration of the small highlight on the metallic tip.

Next examples show a blurred shell, in Figure 5.11 and a flower image in Figure 5.12. The first one appears convincing. The starting image is quite dark since it directly comes from the *raw* format of the camera, before intensity levels and color ratios are translated to the standard RGB color space. Also the matting process gave very high quality results, with only slight defects in the shadow below the object.

In the pink flower image there are some artifacts on the right edge of the object. Here the object and its background share the same color, misleading the matting algorithm. Artifacts in the starting maps are further worsened by the deconvolution.

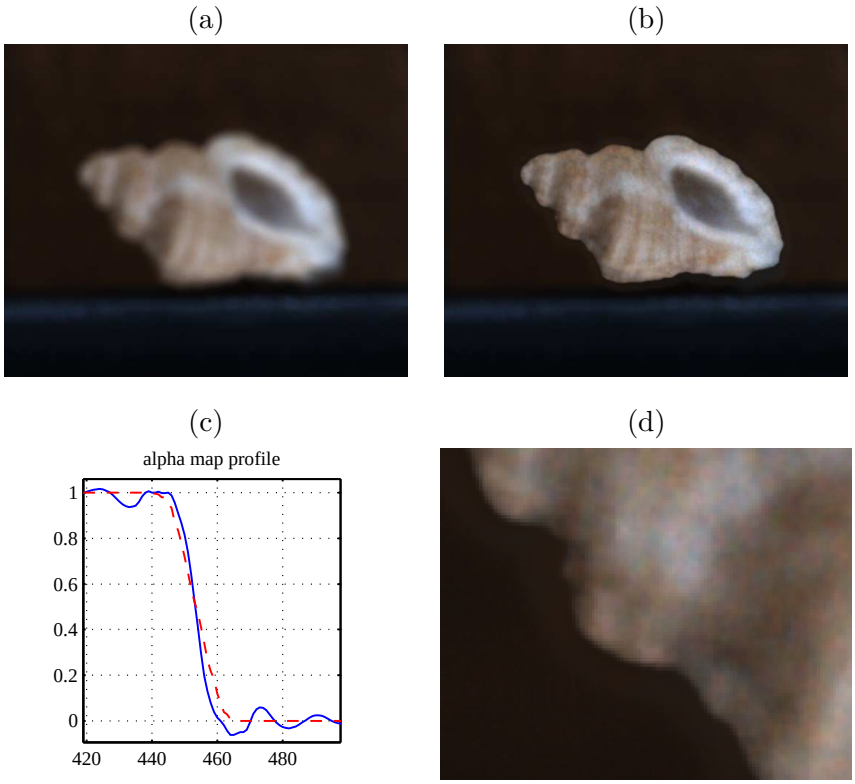

Figure 5.11: (a) Starting image. (b) Restored image. (c) Comparison of the alpha map profile before and after the restoration, along the central row of the image. (d) A detail of the restored image.

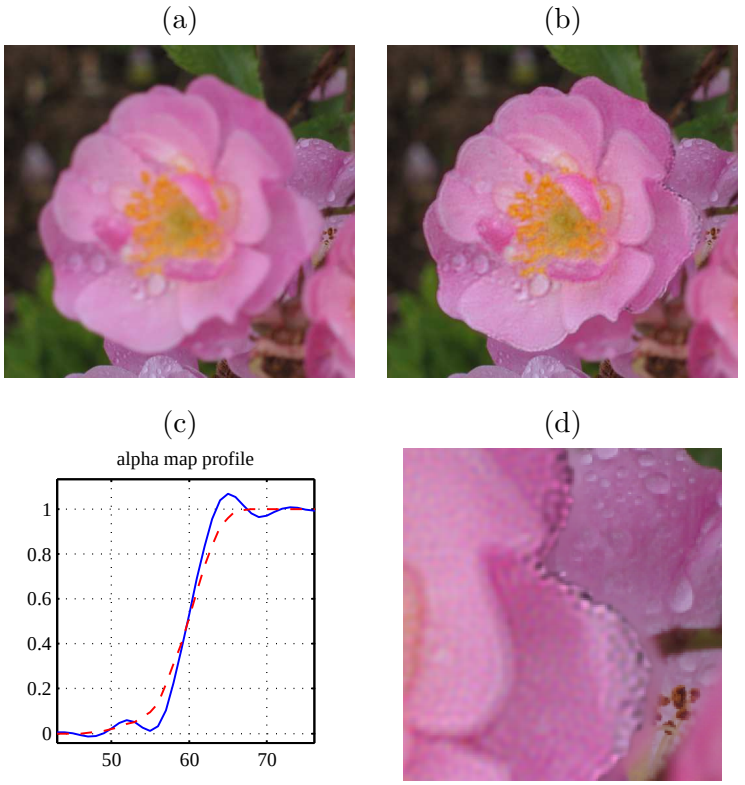

Figure 5.12: (a) Starting image. (b) Restored image. (c) comparison of the alpha map profile before and after the restoration, along the central row of the image. (d) Artifacts in the matte due to color ambiguity are critically amplified in the restoration process.

Chapter 6

Conclusions

This final chapter deals with the conclusions and other remarks on the work done on the restoration of blurred objects. The main goal is deblurring an object without altering the background scene. Most existing techniques do not easily allow this without a considerable amount of manual work, since they are aimed at the restoration of the whole image.

An advantage of the proposed approach lies in the use of the data coming from the matte extraction phase to identify the PSF and restore the image of the object without affecting the background. The whole process requires minimal user interaction and can be performed fast enough for image editing applications, after some optimizations. The idea originated from research on the characteristics of the alpha map of motion blurred objects in [2]. While using a threshold on the alpha map to estimate the unknown silhouette was quickly found to be more adequate to restore out of focus objects, the application to motion blur is still open to future developments.

Experimental results confirm that the behavior under controlled conditions is convincing. The PSF parameter is identified correctly and the restored images look good. A simple pre-filtering step on the alpha map confers tolerance to noisy data, up to reasonable noise levels. The iterative refinement phase helps reducing the typical ringing artifacts that often re-

sult from deconvolution. Another interesting feature is the automatic alpha map post-filtering that helps to improve the look of the results. Computation time is mainly dominated by the 2D FFT, extensively used in both estimation and refinement phases.

There are several drawbacks and disadvantages with this approach. One strong limitation is due to the inherent difficulty of the matting problem: even the most advanced techniques to date do not guarantee correct results, albeit visually pleasing. Naturally this is a major issue since the alpha map is the most critical input.

Another issue with camera images lies in the simplified PSF models: Gaussian, disk or rectilinear motion blur. This is definitely a weakness with respect to the modern and more sophisticated non-parametric blind deconvolution techniques.

There are also pathological cases where the blur is so strong that all pixels in the alpha map are below the threshold level, preventing the estimation of the PSF.

In what follows are now shown potential applications and future developments of the proposed approach.

6.1 Potential applications

The first application one can imagine for the selective deblurring of objects is fixing bad pictures with little manual effort, for all those cases where the blur degradation only affects a well defined object in an image.

Moreover, once an object has been restored, it is possible to modify the amount of blur affecting either the object or even the background image. This way it is possible to create smooth animations where the object is gradually put at focus while the background becomes blurry.

Another application would be producing an artificial *panning* effect from a single long exposure image of a moving object. Panning is a particular

photographic technique typically used with fast moving objects such as cars or planes. It consists in moving the camera during the shot, to achieve a sharp image of the object on a motion blurred background. It takes practice to master, but the results are impressive.

The proposed algorithm could be used to achieve a similar effect. First the blurred object is restored; the identified blur function is applied to the background, and the whole image is recombined. Figure 6.1 shows how artificial panning would look. Naturally there are many limitations when working with motion blur at the current time, not to mention the necessity to compute the matte in a very accurate way, that is unlikely in case of real images. However this is still open to future developments.

Figure 6.1: Artificial panning on a synthetic test image.

6.2 Future developments

There are aspects that deserve a deeper analysis in ongoing works. The first interesting area is natural image matting; since the accuracy of the matte is critical to the outcome of the PSF estimation, it is worth researching how to best adapt existent approaches to either out of focus or motion blurred images; in particular, the Wang-Cohen iterative algorithm could be implemented and tuned specifically for blurred images, exploiting the additional knowledge on the smoothness of the alpha map. Meanwhile, a

shortcut to limit the impact of bad mattes could be manually masking out low quality regions in the computation of the cost function used to refine the model. Also, the brief detour in Section 3.1 on alpha matting with a known background scene could be developed to allow for a more accurate matte in controlled environments.

Another weak aspect is the dependence from a parametric PSF that is often inadequate to represent the complexity of the real image formation process; also, being restricted to rectilinear motion blur is a strong limitation. The natural evolution would consist in developing a non-parametric PSF estimation that takes advantage of the results of this work.

Last, the proposed approach can be implemented in the form of a plug-in to an existing image editing software, aiming at performance optimization.

Bibliography

[1] R. H. T. Bates. Astronomical speckle imaging. *Phys. Rep.*, 90(4):203–97, Oct 1982.

[2] Vincenzo Caglioti and Alessandro Giusti. On the apparent transparency of a motion blurred object. In *Proceedings of ICCV workshop on Photometric Analysis in Computer Vision (PACV)*, 2007.

[3] M. Cannon. Blind deconvolution of spatially invariant image blurs with phase. *IEEE Trans Acoust, Speech, Signal Processing*, 24(1):58–63, Oct 1991.

[4] B. Chalmond. Psf estimation for image deblurring. *CVGIP: Graphical Models and Image Processing*, 53(4):364–372, Jul 1991.

[5] M. M. Chang, A. M. Tekalp, and A. T. Erdem. Blur identification using the bispectrum. *IEEE Trans Signal Processing*, 39(10):2323–2325, Oct 1991.

[6] C. M. Cho and H. S. Don. Blur identification and image restoration using a multilayer neural network. *IEEE Int. Joint Conf. on Neural Networks*, 3:2558–2563, 1991.

[7] Yung-Yu Chuang, Brian Curless, David H. Salesin, and Richard Szeliski. A bayesian approach to digital matting. In *Proceedings of*

IEEE CVPR 2001, volume 2, pages 264–271. IEEE Computer Society, December 2001.

[8] H. Farid. Blind inverse gamma correction. *IEEE Trans. on Image Processing*.

[9] Hany Farid and Alin Popescu. Blind removal of image Non-Linearities. In *Proc. ICCV*, volume 1, pages 76–81, 2001.

[10] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Efficient belief propagation for early vision. *cvpr*, 01:261–268, 2004.

[11] R. Fergus, B. Singh, A. Hertzmann, S. T. Roweis, and W.T. Freeman. Removing camera shake from a single photograph. *ACM Transactions on Graphics, SIGGRAPH 2006 Conference Proceedings, Boston, MA*, 25:787–794, 2006.

[12] Raymond Fielding. *Techniques of Special Effects of Cinematography (Library of Communication Techniques, Film)*. Focal Press, October 1985.

[13] D.B. Gennery. Determination of optical transfer function by inspection of frequency-domain plot. *JOSA*, 63(12):1571–1577, December 1973.

[14] Alessandro Giusti and Vincenzo Caglioti. Isolating motion and color in a motion blurred image. In *proc. of British Machine Vision Conference (BMVC) 2007*, 2007.

[15] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2001.

[16] Leo Grady, Thomas Schiwietz, Shmuel Aharon, and Rüdiger Westermann. Random walks for interactive alpha-matting. In J. J. Villanueva, editor, *Proceedings of the Fifth IASTED International Confer-*

ence on Visualization, Imaging and Image Processing, pages 423–429, Benidorm, Spain, Sept. 2005. ACTA Press.

[17] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[18] Jiaya Jia. Single image motion deblurring using transparency. In *CVPR*, 2007.

[19] D. Kundur and D. Hatzinakos. Blind image deconvolution. *Signal Processing Magazine, IEEE*, 13(3):43–64, 1996.

[20] K. Kurosawa, K. Kuroki, and N. Saitoh. Influence of diaphragm of camera on deblurring problem. In A. G. Tescher, editor, *Proc. SPIE Vol. 3164, p. 544-554, Applications of Digital Image Processing XX, Andrew G. Tescher; Ed.*, volume 3164 of *Presented at the Society of Photo-Optical Instrumentation Engineers (SPIE) Conference*, pages 544–554, October 1997.

[21] R. L. Lagendijk, A. M. Tekalp, and J. Biemond. Maximum likelihood image and blur identification: a unifying approach. *Optical Engineering*, 29(5):422–435, May 1990.

[22] Anat Levin. *Blind Motion Deblurring Using Image Statistics*, pages 841–848. MIT Press, Cambridge, MA, 2007.

[23] Anat Levin, Dani Lischinski, and Yair Weiss. A closed form solution to natural image matting. In *CVPR '06: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 61–68, Washington, DC, USA, 2006. IEEE Computer Society.

[24] Ta-Hsin Li and Keh-Shin Lii. A joint estimation approach for two-tone image deblurring by blind deconvolution. *IEEE Transactions on Image Processing*, 11(8):847–858, 2002.

[25] Yin Li, Jian Sun, Chi-Keung Tang, and Heung-Yeung Shum. Lazy snapping. *ACM Trans. Graph.*, 23(3):303–308, 2004.

[26] L. B. Lucy. An iterative technique for the rectification of observed distributions. *Astronomical Journal*, 79:745–+, June 1974.

[27] Y. Mishima. Soft edge chroma-key generation based upon hexoctahedral color space. *U.S. Patent 5,355,174*, 1993.

[28] J. A. Nelder and R. Mead. A simplex method for function minimization. *The Computer Journal*, 7:308–313.

[29] Thomas Porter and Tom Duff. Compositing digital images. In *SIGGRAPH '84: Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, pages 253–259, New York, NY, USA, 1984. ACM.

[30] S. J. Reeves and R. M. Mersereau. Blur identification by the method of generalized cross-validation. *IEEE Trans Image Processing*, 1(3):301–311, Jul 1992.

[31] William Hadley Richardson. Bayesian-based iterative method of image restoration. *J. Opt. Soc. Am.*, 62(1):55, 1972.

[32] M. Ruzon and C. Tomasi. Alpha estimation in natural images. In *In Proceedings IEEE Conference on Computer Vision and Pattern Recognition*, 2000.

[33] Alvy Ray Smith and James F. Blinn. Blue screen matting. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer*

graphics and interactive techniques, pages 259–268, New York, NY, USA, 1996. ACM.

[34] Jian Sun, Jiaya Jia, Chi-Keung Tang, and Heung-Yeung Shum. Poisson matting. *ACM Transactions on Graphics*, 23(3), 2004.

[35] P. Vlahos and B Taylor. *Traveling Matte Composite Photography*, pages 430–445. American Society of Cinematographers, 1993.

[36] Jue Wang and Michael F. Cohen. An iterative optimization approach for unified image segmentation and matting. In *ICCV '05: Proceedings of the Tenth IEEE International Conference on Computer Vision*, pages 936–943, Washington, DC, USA, 2005. IEEE Computer Society.

[37] Y. Yitzhaky, I. Mor, A. Lantzman, and N. S. Kopeika. Direct method for restoration of motion blurred images. *JOSA-A*, 1998.

Appendix A

Source code

Important Matlab scripts and functions will be included in this appendix.

A.1 Matting code

exactmatte.m

This function implements the simplified matting approach analyzed in section 3.1. F and B are assumed to be constant over the whole image

```
% [alpha] = exactmatte(I, F, B)
% I = input RGB image.
% F = foreground color in RGB space, values must be in [0, 1]
% B = background color in RGB space, values must be in [0, 1]
% alpha = computed matte

function [alpha] = exactmatte(I, F, B)
%% transform F-B into the red axis
v = F - B;

% transform F-B vector into the new red axis
v1 = v;
% any direction orthogonal to v1.
```

```
v2 = [v(2); -v(1); 0];
% v3 is orthogonal to both v1 and v2
v3 = cross(v1,v2);

% H rotates the reference system such that BF vector and
% translates the origin to B.
H = inv([v1,v2,v3,B; ...
         0, 0, 0, 1])

for i = 1:size(I, 1)
  for j = 1:size(I, 2)
    % convert each pixel to homogeneous coordinates and
    % transform it by H
    p = [I(i,j,1); I(i,j,2); I(i,j,3); 1];
    J(i,j,:) = H*p;
  end
end

% recover the matte from the red channel
alpha = J(:,:,1) ./ J(:,:,4);
```

unshadingmatte.m

The more sophisticated matting technique presented in section 3.1.

```
%% -- Script arguments --
% F and B are the input fore/background colors as RGB column vectors
% I contains the starting image
% J will contain the output matte

%% H1 transforms FOB plane into the rOg plane
H1 = inv([F, cross(cross(F,B),F), cross(F,B), zeros(size(F)); 0,0,0,1]);

%% H2 ignores the b component, projecting everything on the rg plane
H2 = [1 0 0 0;...
      0 1 0 0;...
      0 0 0 1];

%% H3 is the degenerate planar homography collapsing all image points
% on the red axis.
% H3, which has 8 dof, is found by fixing the mapping between 4 points
% and their respective images.

% The mapping is defined as follows: F and F/2 transform into [1, 0];
% B and B/2 transform into B. These points are referred to the
% reference system transformed by H1 and H2, hence they were named
% F_12# and B_12#
% H3 is found imposing this mapping in the equation H3*x = x'

F_121= H2*H1*[F; 1];
F_122 = F_121*0.5; F_122(3) = 1;
B_121 = H2*H1*[B; 1];
B_122 = B_121*0.5; B_122(3) = 1;

% The problem can be solved defining a matrix A, whose (right) null
```

```
% space is a vector containing H3's unknown coefficients.

A=[F_121', 0,0,0, -F_121';...
   0,0,0,  F_121', 0,0,0;...
   F_122', 0,0,0, -F_122';...
   0,0,0,  F_122', 0,0,0;...
   B_121', 0,0,0, -B_121'*B_121(1);...
   0,0,0,  B_121',-B_121'*B_121(2);...
   B_122', 0,0,0, -B_122'*B_121(1);...
   0,0,0,  B_122',-B_122'*B_121(2) ];

h3_v = null(A,'r');

% build the matrix from its coefficients
H3 = zeros(3);
for i = 1:3
  for j= 1:3
    H3(i,j) = h3_v(3*(i-1)+j);
  end
end

newx = F_121(1:2)/F_121(3) - B_121(1:2)/B_121(3);
newy = [newx(2), -newx(1)]'; % orthogonal to newx

H4 = inv([newx, newy, B_121(1:2)/B_121(3);...
          0,    0,    1          ]);

%% apply the computed transformation to the image
J = zeros(size(I,1), size(I,2));
for i = 1:size(I,1)
  for j = 1:size(I,2)
    P = ones(4,1);
    for c = 1:3
```

```
      P(c) = I(i,j,c);
    end
    P1 = H * P;
    % just store the red channel in the output J
    J(i,j) = P1(1)/P1(3);
  end
end
```

A.2 Linear filtering

This section contains the code used to add and remove blur given the PSF.

blur.m

Filter input image `y` using `v` as blur kernel.

```
function [y_blur] = blur (y, v)
 [yN, xN] = size(y);
 [ghy, ghx] = size(v);

 % pads PSF with zeros to whole image domain, and centers it.
 big_v = zeros(yN, xN);
 big_v(1:ghy, 1:ghx) = v;
 big_v = circshift(big_v, -round([(ghy-1)/2 (ghx-1)/2]));

 % Frequency response of the PSF
 V = fft2(big_v);
 % performs filtering (convolution is obtained by
 % product in frequency domain)
 y_blur=real(ifft2(V.*fft2(y)));
```

deblur.m

Wiener deconvolution, in case only the inverse signal to noise ratio is given

```
function [out] = deblur(y_blur, v, k)

 [yN, xN] = size(y_blur);
 [ghy, ghx] = size(v);

 Y = fft2(y_blur);

  big_v = zeros(yN, xN);
  big_v(1:ghy, 1:ghx) = v;
  big_v = circshift(big_v, -round([(ghy-1)/2 (ghx-1)/2]));

 % Frequency response of the PSF
 V = fft2(big_v);
 % compute and apply the Wiener filter
  out = real(ifft2((Y.*conj(V))./(conj(V).*V + k)));
```

A.3 Deblurring

This section includes source code for all the components of the main deblurring algorithm.

deblurring.m

Here follows a list of the parameters needed by the algorithm.

I, alphamap, **colormap**, background contain respectively the starting image, alpha, color and background.

model = 'gaussian'|'disk'|'motion' selects the type of blur that needs to be removed.

epsilon is the starting value for the regularization term. Empiric tests show that a good value to start with is 0.01.

b_mirror = true if object intersects the image borders, otherwise false. Does not work with motion blur.

```
function [composition] = ...
  deblurring(I, alphamap, colormap, background, ...
             model, epsilon, b_mirror)

  threshold = .5; % unless a more accurate value is known.
  pre_filter = 1; % increase with caution, if necessary.
  norm = 'square'; % use 'inf' to minmax error on param. estimation.
                   % instead of the L2 norm
  maxiters = 3; % iterations of the final parameter refinement.

  if b_mirror
    alphamap = quadmirror(alphamap);
    colormap = quadmirror(colormap);
  end

  %% pre-filtering
```

```
alphamap_ = alphamap;
if pre_filter > 0
  alpha_filter = ...
    fspecial('gaussian',ceil(max(10,5*pre_filter)),pre_filter);
  alphamap_ = medfilt2(alphamap, [5,5]);
  alphamap_ = blur(alphamap_, alpha_filter);
end

%% defocus blur: estimate the latent alphamap and the psf
if ~strcmp (model, 'motion')
  alpha_zero_hat = double(alphamap_ > threshold);
  psf_hat = deblur(alphamap_, alpha_zero_hat, epsilon);

  ypsf = size(psf_hat, 1);
  xpsf = size(psf_hat, 2);

  [support, start]= getPSFSupport(psf_hat)

% crop a square in the middle of the estimated psf
% with size equal to the estimated support
if mod(support,2) == 0
  psf_hat = ...          % even support
   psf_hat(ceil(ypsf/2-(support/2-1)):ceil(ypsf/2+(support/2-1)),...
           ceil(xpsf/2-(support/2-1)):(xpsf/2+(support/2-1)));
else
  psf_hat = ...          % odd support
   psf_hat(ceil(ypsf/2-(support/2-1)):ceil(ypsf/2+(support/2)), ...
           ceil(xpsf/2-(support/2-1)):ceil(xpsf/2+(support/2)));
end

%% identify the parameter according to the given model
if strcmp (model, 'disk')
  diskradius = getDiskRadius(psf_hat, start, norm);
```

```
    % objective function for refinement:
    objfunc = @disk_func;
    % starting value for refinement
    x = diskradius
  elseif strcmp (model, 'gaussian')
    sigma = getSigma(psf_hat, start, norm);
    % objective function for refinement:
    objfunc = @gaussian_func;
    % starting value for refinement
    x = sigma
  end
else % (motion blur case)
    % estimating direction
    l_alpha = computeLaplacian(alphamap_);
    radon_objfunc = @(theta,alpha) -max(radon(alpha, theta));
    % play with the initial value if unsuccessful
    [theta, foo] = fminsearch(@(x) radon_objfunc(x, l_alpha), 0);
    % estimating length
    ly = upper_bound(alphamap_);
    lx = upper_bound(alphamap_');
    l = sqrt(lx^2 + ly^2);
    objfunc = @motion_func;
    x = [l, theta + 90]
    % or [l, atan2(ly, lx)*180/pi]
    % psf_hat = fspecial('motion', x(1), x(2));
  end

  %% iterative parameter refinement
  options = optimset;
  % Modify options setting
  options = optimset(options,'Display','iter');
  options = optimset(options,'MaxFunEvals',20);
```

```
for i = 1:maxiters
  [x, fval] = ...
    fminsearch(@(x) objfunc(x,epsilon,alphamap_), x, options)

  [epsilon, fval] = ...
    fminsearch(@(k) objfunc(x,k,alphamap_), epsilon, options)
end

% finally build the correct PSF
if strcmp(model, 'disk')
  final_psf = fspecial( 'disk',x(1));
elseif strcmp(model, 'gaussian')
  final_psf = fspecial('gaussian', max(10, ceil(5*x(1))), x(1));
else
  final_psf = fspecial('motion', x(1), x(2));
end

%% restoration via wiener deconvolution
rst_alphamap = deblur(alphamap_, final_psf, epsilon);
for i=1:3
  colormap(:,:,i) = colormap(:,:,i).*alphamap;
  rst_colormap(:,:,i)=deblur(colormap(:,:,i),final_psf,epsilon);
end

% downsize if mirrored
if b_mirror
  s = size(rst_alphamap);
  rst_alphamap = rst_alphamap(1:s(1)/2,1:s(2)/2);
  rst_colormap = rst_colormap(1:s(1)/2,1:s(2)/2,:);
  alphamap = alphamap(1:s(1)/2, 1:s(2)/2);
end

%% alphamap post-filtering
```

```
% rethreshold, shrinking it a bit by 1 pixel erosion,
% then smooth.
rst_alphamap = double(rst_alphamap > threshold);
rst_alphamap = imerode(rst_alphamap, ones(3));
rst_alphamap = blur(rst_alphamap, fspecial('gaussian',5,1));

%% final output
for i = 1:3
  composition(:,:,i)= rst_alphamap.*rst_colormap(:,:,i)+...
                      (1-rst_alphamap).*background(:,:,i);
end

figure;
  subplot(1,2,1)
    imshow(I);
    title('Starting image');
  subplot(1,2,2)
    imshow(composition);
    title('Final composition');
```

quadmirror.m

Creates an image tiling mirrored and flipped replicas of the input.

```
% An image is mirrored in both H and V directions
% (doubling output size)
function [out] = quadmirror(in)
  rows = size(in, 1);
  cols = size(in, 2);

  ul = in;
  c = size(in, 3);
  for i = 1:c
    ur(:,:,i) = fliplr(ul(:,:,i));
```

```
    ll(:,:,i) = flipud(ul(:,:,i));
    lr(:,:,i) = flipud(ur(:,:,i));

    out(1:rows, 1:cols, i) = ul(:,:,i);
    out(1:rows, cols+1:2*cols, i) = ur(:,:,i);
    out(rows+1:2*rows, 1:cols, i) = ll(:,:,i);
    out(rows+1:2*rows, cols+1:2*cols, i) = lr(:,:,i);
  end
```

getPSFSupport.m

Roughly estimates the support of the estimated PSF, and gives a starting value for the parameter. Not used in motion blur estimation.

```
function [l, r] = getPSFSupport(psf)

  % sum all pixels greater than half the maximum
  psf_thresh = psf > 0.5*max(psf(:));
  area = sum(psf_thresh(:));
  r  = ceil(sqrt(area/pi));

  % make it an odd number, easier to center
  l = max(1 + 6*r, 51);
```

getSigma.m

Estimates the parameter sigma of a Gaussian blur filter

```
function [sigma] = getSigma(psf, start_sigma, norm)

  % check if psf is square
  if size(psf,1) ~= size(psf,2)
    disp('warning: psf must be square ... resizing');
    x = min(size(psf,1),size(psf,2));
    psf = psf(1:x, 1:x);
```

```
  end

  support = size(psf,1);
  % trying different norms for the error function.
  errorfun_square = ...
    @(s,l,p) sum(sum( ((p - fspecial('gaussian', l, s) )).^2));
  errorfun_minmax = ...
    @(s,l,p) max(max(p - fspecial('gaussian', l, s) ));

  sigma_square = ...
    fminsearch(@(s)errorfun_square(s,support,psf), start_sigma);
  sigma_minmax = ...
    fminsearch(@(s)errorfun_minmax(s,support,psf), start_sigma);

  if strcmp(norm, 'inf')
    sigma = sigma_minmax;
  else
    sigma = sigma_square;
end
```

getDiskRadius.m

Estimates the radius of a disk blur filter

```
function [r] = getDiskRadius(psf, r_start, norm)

  % check if psf is square
  if size(psf,1) ~= size(psf,2)
    disp('warning: psf must be square ... resizing');
    x = min(size(psf,1),size(psf,2));
    psf = psf(1:x, 1:x);
  end

  support = size(psf, 1);
```

```
  errorfun_square = @(r,l,p) sum(sum( ((p - disk_psf(l, r) )).^2));
  errorfun_minmax = @(r,l,p) max(max(p - disk_psf(l, r) ));

  r_square = ...
    fminsearch(@(r) errorfun_square(r, support, psf), r_start);
  % start higher to avoid local minima
  r_minmax = ...
    fminsearch(@(r) errorfun_minmax(r, support, psf), r_start+5);

  if strcmp(norm,'inf')
    r = r_minmax;
  else
    r = r_square;
  end

function [psf] = disk_psf(support, radius)
  psf = zeros(support);
  center = ceil(support/2);

   % subtract 0.5 to make it interchangeable
   % with fspecial('disk', radius).
   % (also make sure radius is positive)
  radius=max(eps,radius-.5);

  for r = 1:support
    for c = 1:support
      d = sqrt((r-center)^2 + (c-center)^2);
      if d <= radius
        psf(r,c) = 1;
      elseif d >= radius+1
        psf(r,c) = 0;
      else
```

```
            psf(r,c) = radius+1-d;
        end
    end
  end
  % normalize it
psf = psf ./ sum(sum(psf));
```

computeLaplacian.m

Used to approximates the Laplacian of the alpha matte for motion blur direction estimation.

```
function [L] = computeLaplacian(I)

  [gx, gy] = gradient(I);

  gabs = sqrt(gx.^2 + gy.^2);
  gabsl = log(gabs + 1);

  [ggx, ggy] = gradient(gabsl);
  ggabsl = sqrt(ggx.^2 + ggy.^2);

  L = ggabsl;
```

upper_bound.m

Finds the topmost pixel with $\alpha > 0$ and the topmost pixel with $\alpha = 1$. Then returns the difference along the vertical direction, providing an upper bound to the PSF height. Transpose the input alpha map to compute the upper bound for the width.

```
function [y] = upper_bound(alpha)
  sx = size(alpha,1);
  t = .01;
```

```
for i = 1:sx
 if max(alpha(i,:)) > t
   break;
 end
end
for j = i:sx
  if max(alpha(j,:)) > 1-t
    break;
  end
end
y = j-i;
```

disk_func.m

```
% Objective function used by fminsearch
% to refine disk blur parameters
function [J] = disk_func(param, k, alpha)

  alpha_post = deblur(alpha, fspecial('disk',(param(1))),k);
  J = sum(sum( min(abs(alpha_post),abs(alpha_post-1))));
```

gaussian_func.m

```
% Objective function used by fminsearch
% to refine Gaussian blur parameters
function [J] = gaussian_func(param, k, alpha)

  alpha_post = deblur(alpha, ...
          fspecial('gaussian', max(9, ceil(5*param)), param), k);
  J = sum(sum( min(abs(alpha_post),abs(alpha_post-1))));
```

motion_func.m

```
% Objective function used by fminsearch
% to refine motion blur parameters
```

```
function [J] = motion_func(param, k, alpha)

  alpha_post = ...
    deblur(alpha, fspecial('motion', param(1), param(2)), k);
  J = sum(sum( min(abs(alpha_post),abs(alpha_post-1)) ));
```