Motion Deblurring
Contents

Introduction 3

1. Motion Blur Simulation 4

2. Mathematical Models 5
   2.1 2-D Convolution 5
   2.2 Image Formation 7
   2.3 Point Spread Function 8

3. Deconvolution 10
   3.1 Inverse Filtering 10
   3.2 Wiener filter 11
   3.3 Iterative Image Restoration 12
   3.4 Richardson-Lucy Algorithm 13

4. Blind Deconvolution 16

5. Hybrid Imaging Approach to Motion Deblurring 17
   5.1 Hybrid Imaging Systems 17
   5.2 Computing Motion 18
   5.3 PSF Estimation 20
   5.4 Deblurring of Moving Objects 22

Conclusion 23

References 24
Introduction

Motion blur is the result of the relative motion between the camera and the original scene during the integration time of the image. It can be clearly seen in the pictures that were taken with long exposure times and the pictures of fast moving objects. The motion blur effect is often used in computer graphics to make synthetic images and animations look more realistic and to add additional information about the art and the direction of the motion (we will touch the subject of motion blur simulation in section 1). On the other hand, the real-world images often suffer from very strong motion blurs. The typical reasons are the camera motion and the motion of an object on the scene.

In this paper we will discuss the basics of restoration of motion-blurred digital images. The process of motion deblurring can be divided into two parts: the estimation of the function that caused the blur, and applying a restoration algorithm. Since the motion path can be arbitrary, the first problem can be hard to solve.

Section 2 gives an overview of basic mathematical concepts. In sections 3 and 4 we will discuss several general techniques for image restoration. Section 5 describes a hybrid imaging approach to motion deblurring, proposed by M. Ben-Ezra and S. K. Nayar [2]. It will be shown how to construct a hybrid camera, which allows estimating the blurring function during the image integration time and effectively solve the motion deblurring problem.
1. Motion Blur Simulation

Motion blur is not always a negative effect. It makes images and animations look more realistic in certain situations. Motion blur can be observed in almost every movie and TV program. This effect is simulated in many computer games. Motion blurred images contain additional information about the direction of the movement and are more pleasing to our eyes (see Fig. 1).

How can we simulate motion blur in synthetic images? To simulate camera motion we just render the image and apply a convolution operator to it (see section 2.1). A more complex problem is simulating the movement of an object on the scene. One of the techniques is called temporal anti-aliasing. The idea is to render several images with the “moving” object in different positions and then to take the average of the images (see Fig. 2). Temporal anti-aliasing can also be used to simulate motion blur in animations.

Another “method”, which is widely used in computer games, is blending the next frame with the previous ones. This is conceptually not a motion blur and the results are very poor (Fig. 3).

Very often, motion blur is simply an undesired effect. It can significantly decrease the quality of the pictures taken with longer exposure times. In the following sections we will discuss the methods of restoration of motion-degraded images.
2. Mathematical Models

In order to solve the image restoration problem, mathematical models are required for the real world processes involved in image generation, formation and recording. In the following subsections we will learn the basic concepts of convolution, point spread function and spatial invariance. Afterwards, we will derive the equation for the image formation process.

2.1 2-D Convolution

Convolution is both a mathematical concept and an important tool in data processing, in particular in digital signal and image processing. The 1-D convolution of two vectors of polynomial coefficients is equivalent to multiplying the corresponding polynomials. We will define a convolution only for 2-dimensional case, which is relevant to 2-D image blurring and restoration.

Let us assume that we have two continuous 2-dimensional images \( f(y, x) \) and \( h(y, x) \), where the coordinates \( y \) and \( x \) may be any real numbers. Their convolved (or folded) sum is the image \( g(y, x) \) defined by the convolution integrals:

\[
g(y, x) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} h(y-r, x-c) \cdot f(r, c) \cdot dr \cdot dc = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} h(r, c) \cdot f(y-r, x-c) \cdot dr \cdot dc
\]

In case of discrete images the integrals should be replaced with sums:

\[
g(y, x) = \sum_{r=-\infty}^{\infty} \sum_{c=-\infty}^{\infty} h(y-r, x-c) \cdot f(r, c) = \sum_{r=-\infty}^{\infty} \sum_{c=-\infty}^{\infty} h(r, c) \cdot f(y-r, x-c)
\]

In digital image processing the (discrete) images \( f \) and \( h \) have a limited size and are represented by real-valued matrices. The image \( f \) is usually restricted to take only non-negative intensity values and has a larger size than \( h \). The dimensions of the matrix of \( h \) are often called a convolution window.

\( h \) is called the convolution kernel. Its elements can be any real values. The element of the kernel at position \((0, 0)\) is called the kernel key element. Note that the elements are possible at both sides of the key element, in both directions. In other words, the negative indices are allowed. As most programming languages do not directly support negative indices, these are simulated by specifying the coordinates of the kernel key element in the PSF matrix explicitly.

If \( h \) is a \( M_h \times N_h \) – matrix with a key element at position \((y_0, x_0)\), and the negative indices are not allowed, then the convolution is written:

\[
(*) \quad g(y, x) = \sum_{r=0}^{M_h-1} \sum_{c=0}^{N_h-1} h(r, c) \cdot f(y+y_0-r, x+x_0-c)
\]
(*) is often abbreviated to
\[ g(y, x) = h(y, x) \otimes f(y, x) \quad \text{or} \quad g = h \otimes f \]

If the sum of all elements of a convolution kernel is 1 (one), the kernel is called normalized.

If we look at the equations above more precisely, we can see that values of \( f \) can be required that are outside the image boundaries. The question arises: what values should we assign to \( f(y, x) \) for \( y < 0, \ y \geq \text{Height}(f), \ x < 0 \) or \( x \geq \text{Width}(f) \)? This problem is known as edge handling. The standard alternatives are:

(a) extend the image with a constant (possibly zero),
(b) extend the image periodically (circular),
(c) extend the image by mirroring it at its boundaries (reflexive or symmetric), or
(d) extend the values at the boundaries indefinitely (replicate).

Convolutions are used in many filter operations (e.g. blurring, deblurring, sharpening, offset). Some examples are shown in Fig. 4.

Note that the average intensity value of an image does not change after applying a normalized convolution kernel.

Convolution can be alternatively implemented through the use of the Frequency domain (this way is faster if the kernel is a large matrix). This aspect will be discussed below in the section about inverse filtering.

For more information on convolution see [10], [12].
2.2 Image Formation

The image formation process can be illustrated as follows:

Before being recorded, the ideal image is blurred by some function $h$, which is called impulse response function, blur function or point spread function (PSF). This degradation can be caused by relative motion between the camera and the original scene (motion blur), by optical system, which is out of focus, by atmospheric turbulence etc.

In addition to these blurring effects, the recorded image is also corrupted by noises $n$. These may be introduced by the transmission medium (e.g. a noisy channel), the recording medium (e.g. film grain noise), measurement errors due to the limited accuracy of the recording system, quantization of the data for digital storage or any combination of these.

The image formation process can be described by the following equation:

$$ g(y,x) = n \left( \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} h(y,x ; r,c) \cdot f(r,c) \cdot dr \cdot dc ; y,x \right), $$

where $f$ is the original (ideal) image, $g$ is the observed image, $h$ is the PSF and $n$ is the noise function. The PSF represents a real-valued intensity distribution and takes non-negative values only.

This model, though very exact, is not very useful for practical purposes. First, the complexity implied by the possibility of having a different PSF $h(y, x; r, c)$ at each coordinate $(y, x)$ of the image is unacceptable from a computational viewpoint. Second, it seems to be impossible to estimate a different PSF for every location in the image.

Therefore we assume that the PSF is spatial invariant over the image, i.e. is for every coordinate $(y, x)$ the same (every point is blurred equally):

$$ \forall y, x, r, c : \ h(y, x ; r,c) = h(y-r, x-c) $$

Note that this assumption may be unacceptable in certain situations (see Fig. 1). In such cases, the entire image region is to be divided in sub-regions so that the PSF is (at least
approximately) spatial invariant in every separate region. Each sub-region should then be processed separately.

For most problems we can accept the additive zero-mean noise, which is statistically uncorrelated with the image. This is a simplification since some noises (such as film grain noise) are not uncorrelated with the input and even be non-additive. This simplification nonetheless leads to restoration methods, which can be applied to a wide class of problems. Thus, the (simplified) image formation process is:

\[ g(y,x) = \int \int h(y-r, x-c) \cdot f(r,c) \cdot dr \cdot dc + n(y,x) \]

or in case of discrete images:

\[ g(y,x) = \sum_{r=-\infty}^{\infty} \sum_{c=-\infty}^{\infty} h(y-r, x-c) \cdot f(r,c) + n(y,x) \]

\[ = h(y,x) \otimes f(y,x) + n(y,x) \]

Note that both the original image and the PSF have unlimited size. In practice, however, they are represented by some limited matrices.

### 2.3 Point Spread Function

We have discovered that the PSF can be represented by a convolution kernel. It is worth saying that not every convolution kernel is a valid PSF. We will now define the constraints that a PSF must satisfy.

Any PSF is an energy distribution function

\[ h: \mathbb{R} \times \mathbb{R} \to \mathbb{R}_+, \quad (y,x) \mapsto e, \]

where \((y,x)\) is a location and \(e\) is the energy level at that location. So, every convolution kernel that contains negative values is not a valid PSF. The kernel \(h\) must also satisfy the following **energy conservation constraint** to be a valid blurring function:

\[ \int \int h(r,c) \cdot dr \cdot dc = 1 \quad \left( \sum_{r} \sum_{c} h(r,c) = 1, \quad \text{in case of a discrete kernel}, \right) \]

which states that the energy is neither lost nor gained by the blurring operation. In other words, \(h\) must be a normalized convolution kernel. An example of a valid PSF is the horizontal motion blur kernel shown in Fig. 4.

We now consider a subset of all possible PSFs: the motion blur PSFs. These can be seen as curves on the plane. The points that belong to the curve have non-zero entries at corresponding positions in the PSF matrix; those that do not belong to the curve have zero entries.
In order to define additional constraints, we use a time parameterization of the motion blur PSF with a path function $p$ and an energy function $e$. We assume that the image is integrated over the time interval $[0,1]:$

$$p : [0,1] \rightarrow \mathbb{R} \times \mathbb{R} \quad t \mapsto (y,x)$$

$$e : [0,1] \rightarrow \mathbb{R}_+ \quad t \mapsto e(t)$$

The key element of the parameterized PSF is located at $p(0)$. Due to physical speed and acceleration constraints $p(t)$ must be continuous and at least twice differentiable. If the scene radiance does not change during image integration, then the amount of energy which is integrated at any time interval is equal to the length of the interval, i.e.:

$$\int_{t_1}^{t_2} e(t) \cdot dt = t_2 - t_1, \quad 0 \leq t_1 \leq t_2 \leq 1$$

The constraints we have defined will later be used in the continuous motion blur PSF estimation (see section 5.3).

Before finishing this section, we will give an example of a motion blur PSF. First we define the curve in parametric form:

$$p(t) = \begin{pmatrix} p_y(t) \\ p_x(t) \end{pmatrix} := \begin{pmatrix} 3 + 3 \cdot \sin 6t \\ 10 \cdot t \end{pmatrix}; \quad e(t) := 1$$

These functions define a valid motion blur PSF. To convert this parametric PSF to a matrix, we use the following algorithm:

```plaintext
psf_key_elem_y = py(0) // the coordinates
psf_key_elem_x = px(0) // of the key element

for t = 0 to 1 step 0.01 // set a smaller step for more precision begin
    y = round( py(t) )
    x = round( px(t) )
    psf_matrix[y,x] = psf_matrix[y,x] + e(t)
end

psf_matrix = normalize( psf_matrix )
```

The function `normalize()` multiplies the matrix with a normalization factor, so that the sum of its elements is 1. Having applied the algorithm to our curve, we obtain the following matrix (the key element is marked bold italics):

$$\begin{array}{cccccccc}
0 & 0 & 8 & 10 & 1 & 0 & 0 & 0 \\
0 & 6 & 2 & 0 & 8 & 0 & 0 & 0 \\
2 & 4 & 0 & 0 & 1 & 5 & 0 & 0 \\
3 & 0 & 0 & 0 & 5 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 6 & 0 & 0 \\
0 & 0 & 0 & 0 & 3 & 4 & 0 & 6 \\
0 & 0 & 0 & 0 & 0 & 0 & 6 & 10 & 4 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}$$

The matrix we have obtained is called the *impulse response matrix*; we can convert it to a convolution kernel by rotating it by 180 degrees around the key element. The image on the right is another possible representation of the matrix.
3. Deconvolution

Knowing the PSF and the nature of noise, it may be possible to (at least partially) restore the ideal image. The degree to which this can be done depends on the actual sampling of the image.

The process of reconstruction of the original image from the blurred one with the same blurring function is called deconvolution. Several different techniques are used depending on the data. The methods introduced here are the inverse filtering, the Wiener filter and the Richardson-Lucy method.

3.1 Inverse Filtering

The simplest approach to deconvolution is the inverse filtering. The key idea of this method is that convolutions in spatial domain become pixel-by-pixel multiplications in frequency space:

$$ g = h \otimes f \quad \Leftrightarrow \quad G = H \cdot F, $$

where \( G \) and \( F \) are Fourier transforms of the images \( g \) and \( f \), respectively; \( H \) is the Fourier transform of the convolution kernel.

Because the Fourier transform is periodic in its nature, the edge handling must be circular when performing convolution. This is referred to as circular convolution.

Knowing the convolution kernel \( h \) and the degraded image \( g \), we can restore the original image \( f \) by the following operation:

$$ f = \mathcal{F}^{-1}\left(\frac{\mathcal{F}(g)}{\mathcal{F}(h)}\right), $$

where \( \mathcal{F} \) is the discrete Fourier transform; \( \mathcal{F}^{-1} \) the inverse discrete Fourier transform and \( \mathcal{F}(g)/\mathcal{F}(h) \) denotes the (complex) pixel-by-pixel division. The result of division by zero is defined as the value of dividend, i.e. \( x/0 := x \).

This method seems to be very simple and effective: its complexity is the complexity of the Fast Fourier Transform (FFT). However, it works only under certain conditions. First, the equivalence (**) is only valid if the edge handling is circular (which is not often the case). Otherwise strong deconvolution artifacts may appear at the edges of the restored image, so one may want to copy edge pixels from the blurred image to the restored image (or blend them). Second, the inverse filtering produces unsatisfactory results on (additive-)noised images. The reason for that is explained below.
Let us suppose that some additive noise \( n \) is present:
\[
g = h \otimes f + n \quad \Leftrightarrow \quad G = H \cdot F + N,
\]
where the capitals denote Fourier transforms.

The inverse filtering delivers the following approximation of \( f \):
\[
\tilde{f} = \mathcal{F}^{-1} \left( \frac{H \cdot F + N}{H} \right) = \mathcal{F}^{-1} \left( F + \frac{N}{H} \right)
\]

If \( H \) is a matrix with relatively small absolute values (which is almost always the case if \( H \) is a valid PSF) then the noise \( N \) will be amplified dramatically by the division. This makes the inverse filtering unusable, due to the fact that a small amount of noise is always present.

### 3.2 Wiener filter

The Wiener filter tries to minimize the difference between the blurred image and the restored image. We will learn a simplified version of the Wiener filter.

The formula can be derived from the equation for inverse filtering. Trying to reduce noise amplification, we introduce a multiplicative factor with a constant \( k \) in the denominator and receive:
\[
\tilde{f} = \mathcal{F}^{-1} \left( \frac{H^2}{H^2 + k} \cdot \frac{G}{H} \right) = \mathcal{F}^{-1} \left( \frac{H^2}{H^2 + k} \cdot \frac{H \cdot F + N}{H} \right)
\]

Note that \( k \) is not an image, but a non-negative (real) constant that is added to every pixel of the squared transformed convolution kernel. The greater the value of \( k \), the more noise is suppressed; on the other hand, the restoration process suffers.

The optimal value of \( k \) should be determined manually. General rule is: the value of \( k \) is proportional to the noise-to-signal ratio of the blurred image. The inverse filtering is a special case of the Wiener filter (for \( k = 0 \)). Note that the inverse filtering is the optimal solution if no noise is present. The results of restoration with different values of \( k \) are shown in Fig. 6.

The advantage of the Wiener filter is that it is able to (partially) restore images convolved with most kernels, not only those that are valid PSFs (of course, there are situations when the image cannot be restored at all; consider, for example, the convolution with a zero-filled matrix). However, the kernels that are not valid PSFs are irrelevant to the motion deblurring problem.
The Wiener filter involves calculating Fourier transforms of an image and of a point spread function, so we will define what the Fourier transform of a PSF is.

The Fourier transforms of the (blurred) image and of the PSF must be the same size (in order to be multiplied/divided later). Thus, the PSF must be padded with zeros to the size of the image so that the key element is located in the center of the PSF matrix. If the extended PSF is a \( M \times N \) matrix with the key element at position \((y_0, x_0)\) then its Fourier transform is defined as:

\[
H(u, v) = \sum_{r=0}^{M} \sum_{c=0}^{N} h(r, c) \cdot e^{-i2\pi \left( \frac{u(r-y_0)}{M} + \frac{v(c-x_0)}{N} \right)}
\]

Note that there is no normalization factor \( / (M \cdot N) \) in front of the sums. The Fourier transform of the PSF is called the optical transfer function (OTF). We will not define the inverse transform because we do not need it.

**3.3 Iterative Image Restoration**

Inverse filtering and Wiener filter count to direct (non-iterative) deconvolution algorithms. The advantage of direct methods is the (relative) simplicity of their implementation, while the obtained results are suitable for many purposes.
Aside from direct methods there are a number of iterative methods of deconvolution. Instead of computing the desired (deblurred) image directly, these compute a sequence of images, which converges to the desired image. One of the most popular iterative methods is the Richardson-Lucy algorithm.

### 3.4 Richardson-Lucy Algorithm

The Richardson-Lucy method, developed independently by Richardson [6] and Lucy [4], is the technique most widely used for restoring images. This is an iterative method that maximizes a Poisson statistics image model likelihood function. The algorithm works only with images blurred by a valid point spread function; it will not restore images degraded by a convolution kernel that is not a valid PSF.

The algorithm starts with a “first guess” (first approximation) of the deblurred image. The approximation is “refined” at each iteration using a correction factor based on the ratio between the blurred image and the approximation. The algorithm is described below in detail:

- A first approximation \( \tilde{f}_0 \) must be specified to start iterations. It is recommended to use a constant image with a constant value equal to the average of the blurred image. However, the algorithm works well with any positive constant value, so we can spare ourselves calculating the average. The algorithm will not work with a constant-zero-image as the first guess.

- The first step of the iteration performs a convolution of the current approximation with the PSF:

  \[ \phi_n = h \otimes \tilde{f}_n \]

- The second step computes a correction factor based on the result of the last operation and the original (degraded) image:

  \[ \phi_n = h \otimes \frac{g}{\phi_n} \]

  where “\( g/\phi_n \)” denotes pixel-by-pixel division (do not confuse with matrix division). If the value of some pixel of \( \phi_n \) is zero, the result of the division is defined as zero, i.e., \( x/0 := 0 \). The quotient image is convolved with the PSF, rotated by 180 degrees around its key element.

- Finally, a new approximation is the product of the current one and the correction-factor:

  \[ \tilde{f}_{n+1} = \tilde{f}_n \cdot \phi_n \]

  where “\( \cdot \)” is the pixel-by-pixel multiplication.
To make it short, the Richardson-Lucy (R-L) algorithm is defined recursively in the following way:

\[
\forall y, x : \tilde{f}_0(y, x) := c, \quad c \in \mathbb{R}_+ \setminus \{0\}
\]

(R-L iteration)

\[
\tilde{f}_{n+1} = \tilde{f}_n \cdot \left( h \otimes \frac{g}{h \otimes \tilde{f}_n} \right), \quad n > 0
\]

The R-L iteration should be repeated until the corrections are sufficiently small. The question of where to stop is a difficult one. The number of iterations required to obtain a good solution depends on the size and complexity of the PSF matrix. The algorithm usually reaches a stable solution very quickly (e.g. 3-8 steps) with a 3×3 PSF matrix, and a 5×5 matrix requires at least 20 steps. If one stops after a very few iterations then the image is still very smooth. On the other hand, increasing the number of iterations not only slows down the computational process, but also amplifies noise and introduces the so called ringing effect (see Fig. 7). For best results, it is preferably to manually find the optimal number of iterations for every image and every PSF.

**Fig. 7. Image restoration: the Richardson-Lucy algorithm.**

(a) Ideal image (pixel dimensions: 415 × 289).
(b) The PSF.
(c) Blurred and noised image (1.5% Gaussian noise).
(d) Restored image: 20 iterations.
(e) Restored image: 100 iterations. The amplified noise is clearly visible. The ringing effect can be seen.
(f) The image restored from the blurred one without noise: 200 iterations. The ringing effect is clearly visible.
The R-L algorithm has a number of important characteristics:

- The R-L iteration converges to the maximum likelihood solution for Poisson statistics in the data.

- The R-L method forces the restored image to be non-negative. Given that the degraded image and the first guess are everywhere non-negative, none of further approximations can be negative.

- The R-L method conserves energy both globally and locally at each iteration. Thus, neither the restored image nor any (significant) part of it will seem much brighter or darker than the blurred image (or its corresponding part).

- The restored images are robust against small errors in the point spread function, which is very important when the PSF is not known exactly and only an estimation is available (see section 5.3).

- Typical R-L restorations require a manageable amount of computer time.

Despite its advantages, the R-L method has some serious shortcomings. In particular, noise amplification and ringing effects can be problematic. This is a generic problem for all maximum likelihood techniques, which attempt to fit the data as closely as possible. The usual practical approach to limiting these undesired effects is simply to stop the iteration when the restored image appears to become too noisy or the “rings” are clearly visible. Another way is to apply noise- or ringing reduction algorithms to the resulting image. The third possibility is using modifications of the R-L algorithm, such as the damped Richardson-Lucy method (see [7]), which were developed to reduce the undesired effects.

The practice shows that a large amount of noise is introduced at the edges of the restored images, so sometimes it makes sense to copy (or blend) some rows of edge pixels from the blurred image to the restored one.

There are many other methods of iterative image restoration e.g.: POCS – for restoration after a linear degradation, Tikhonov-Miller regularization, the method of triterations etc. It is sometimes worthwhile to combine two or even more of these methods to achieve better results.
4. **Blind Deconvolution**

The deconvolution methods described above require the knowledge of the point spread function. As the PSF is not usually known, a large amount of research has been dedicated to the estimation of the PSF from the blurred image itself. This is done using the methods of *blind deconvolution*.

There are certain situations in which one may assume that the PSF that caused the blur can be parameterized by a specific and very simple model (e.g. a constant velocity linear movement or a circular blur through defocus). Although such PSFs require only one or two parameters, finding the correct values of the parameters may be very difficult. Besides, this approach is inapplicable to the real-world images with more complex PSFs.

Several algorithms assume that an approximation of the non-blurred image is available. The PSF can then be estimated from the blurred image and this approximation. Such algorithms can rarely be used as it is usually nothing known about the ideal image.

Ayers and Dainty [1] proposed to extend the Richardson-Lucy method to the unknown PSF by iterating on each of the unknowns. There are several iteration schemes for this algorithm. The advantage of this method is that it does not require any information about the ideal image or the PSF.

All the methods of blind deconvolution do not work particularly well for real-world images, so they cannot be effectively applied to the motion deblurring problem in case of a complex PSF (the results of the R-L blind deconvolution are shown in Fig. 8: the restored images are degraded by strong deconvolution artifacts). However, many techniques work well in specific situations (e.g. a symmetric PSF).
5. Hybrid Imaging Approach to Motion Deblurring

We now address the problem of motion blur due to a camera motion. We assume that the motion is shift-invariant, i.e. the camera is only moving in a plane parallel to the scene and does not rotate.

Two hardware approaches to the motion blur problem have been recently put forward. The first approach uses optically stabilized lenses for camera shake compensation. These lenses have an adaptive optical element, which is controlled by inertial sensors, that compensates for camera motion. This method is effective only for relatively small exposures (less than $1/15$ of a second). The second approach uses special sensors which prevent motion blur by selectively stopping the image integration in areas where motion is detected. It does not, however, solve the problem of motion blur due to camera motion.

Another approach was proposed by Ben-Ezra and Nayar [2]. This method estimates the PSF that caused the blur from sparse real motion measurements that are taken during the integration time of the image. The estimated PSF is then used to deblur the image.

5.1 Hybrid Imaging Systems

First we need to understand the relationship between two main characteristics of a digital camera: the spatial resolution (number of pixels) and the temporal resolution (number of images per second). Image detector of a digital camera consists of a number of pixel detectors which receive light energy during the integration time of the image. The total light energy received by a pixel during integration must be above a minimum level for the light to be detected (this minimum level is determined by the signal-to-noise characteristics of the detector). Therefore the exposure time required to ensure detection of the incident light is inversely proportional to the area of the pixel. In other words, the temporal resolution is inversely proportional to the spatial resolution.

This relationship is illustrated by the solid line in Fig. 9. The points on the line represent cameras with different spatio-temporal characteristics. If we use two very different operating points on the line, we can simultaneously obtain high spatial resolution with low temporal resolution (primary detector) and high temporal resolution with low spatial resolution (secondary detector). Such a scheme is called a hybrid imaging system.

The high-resolution image suffers from the motion blur due to a camera motion. Because low-resolution images, provided by the secondary detector, were taken with very short exposure times, we can assume that they do not suffer from the motion blur. We use this low-resolution image sequence to estimate the PSF that caused the blur and to apply a deconvolution algorithm to the high-resolution image.
Ben-Ezra and Nayar [2] have proposed three conceptual designs for the hybrid imaging system. The simplest design uses a rigid rig of two cameras: a high-resolution still camera as the primary detector and a low-resolution video camera as the secondary detector (Fig. 10-a). It is advantageous to make the secondary detector black and white since such a detector can have a higher temporal resolution.

The second design uses the same lens for both detectors by splitting the image with a beam splitter (Fig. 10-b). This model requires less calibration than the previous one.

The third design (Fig. 10-c) includes the primary and the secondary detectors on the same chip, which has a high-resolution central area (primary detector) and a low-resolution periphery (secondary detector). This chip can be implemented using binning technology, which combines the charge of a group of adjacent pixels before digitization. This allows switching between a normal full-resolution mode (when binning is off) and a hybrid primary-secondary detector mode (when binning is activated).

### 5.2 Computing Motion

The secondary detector provides a sequence of low-resolution images (frames) that are taken at fixed intervals during the exposure time. These frames are then used to compute the motion path of the camera.

First we compute the motion between successive frames. This problem is known as the image registration problem. As we have assumed that the camera motion is shift invariant, the registration problem is simplified to the translational registration problem.
The 2-D translational image registration problem (Fig. 11) can be characterized as follows: given two images $f_1(y, x)$ and $f_2(y, x)$, we have to find a disparity vector $d = (y_d, x_d)$ which minimizes some measure of difference between $f_1(y+y_d, x+x_d)$ and $f_2(y, x)$. Typical measures of difference are:

- $L_p$ norm: $L_p = \left( \sum_{(y,x)} |f_1(y+y_d, x+x_d) - f_2(y, x)|^p \right)^{1/p}$ for $p = 1, 2$

- Negative of normalized correlation:
  $$-C = - \left( \frac{\sum_{(y,x)} f_1(y+y_d, x+x_d) \cdot f_2(y, x)}{\sqrt{\sum_{(y,x)} f_1(y+y_d, x+x_d)^2} \cdot \sqrt{\sum_{(y,x)} f_2(y, x)^2}} \right)$$

The obvious technique for registering two images is the exhaustiv search: a measure of difference is calculated for all possible values of the disparity vector. This technique is very time consuming: if the frames produced by the secondary detector are $M \times N$ – images and the region of possible values of $d$ is limited to $d \in \{-M/4 \ldots M/4\} \times \{-N/4 \ldots N/4\}$, then the method requires $O(M^2 N^2 / 4)$ operations.

If the measure of difference is a cumulative function, such as $L_1$ or $L_2$, then the algorithm could be sped up by stopping accumulating the difference for the current $d$ when it becomes apparent that the current $d$ is not likely to give the best match. In other words, when the accumulated difference exceeds some threshold, one goes on to the next $d$. 
The hill-climbing technique is a kind of iterative algorithm. It begins with an initial approximation \( d_0 \) of the disparity (often \( d_0 = (0, 0) \)). To obtain the next approximation \( d_{k+1} \) from \( d_k \) one evaluates the difference function for all vectors in a small neighborhood of \( d_k \) (3×3 or 5×5); \( d_{k+1} \) is the vector that minimizes the difference. The algorithm stops after \( n \)-th iteration if \( d_n = d_{n-1} \). This method is fast, but it suffers from the problem of false peaks: the local optimum that one attains may not be the global optimum (thus, the method can fail to find the best disparity \( d \)).

To find a good initial approximation \( d_0 \) for hill-climbing, one can use the reduced exhaustive search. One selects \( k \) random pixels from the image \( f_1 \) and performs the exhaustive search, calculating the difference only for those pixels. The number of random pixels is proportional to sizes of the images and to their noise-to-signal ratios. Because the pixels are chosen randomly, this method, applied to the same images several times, can return different vectors. If \( k \) is big enough, the result is often very close to the optimal solution. The reduced search is relatively fast, so one can apply it several times and use the average of the obtained results as the initial guess \( d_0 \) for hill-climbing. This speeds up the hill-climbing up to 8 times and increases the reliability of the method. The approximation \( d_0 \) can be alternatively used to find a threshold value for normal exhaustive search: one calculates the measure of difference for \( d_0 \) (all pixels!) and uses this difference as the threshold value. This technique is much faster than the exhaustive search without a threshold, but is still very expensive.

A coarse-fine search strategy uses a pyramid of images (a set of images of the same scene at various resolutions). One of the techniques discussed above is used to find the best match at low resolution, and the low-resolution match is then used to constrain the region of possible matches examined at higher resolutions.

B. D. Lucas and T. Kanade [3] proposed an iterative image registration technique that uses a type of Newton-Raphson iteration. This method involves calculating derivatives of the images, therefore it applies best to continuous-tone images (such as real-world photographs) with only a few sharp details, because it is easy to estimate the derivatives of such images. If applied to sharp images, sophisticated techniques for estimating derivatives of the images should be used, and the method becomes less efficient.

### 5.3 PSF Estimation

Having computed motion between adjacent frames, we can add the disparity vectors successively to obtain discrete samples of the camera motion path function \( p(t) \) (see section 2.3). The next step is to interpolate these samples (which are points on the plane, Fig. 12-a) to estimate the motion path. Ben-Ezra and Nayar suggested using spline interpolation because spline curves are smooth and twice differentiable, which satisfies the speed and acceleration constraints.
In order to estimate the energy function $e(t)$, we need to find the extent of each frame along the interpolated path. We split the interpolated path $p(t)$ into segments with a 1-D Voronoi tessellation, assuming that the discrete motion samples (which we have interpolated) are centers of gravity on the interpolated curve (Fig. 12-b). The Voronoi tessellation ensures that every point on the curve is closer to the center of gravity of the segment it belongs to than to the center of gravity of any other segment. The segments we have received describe the camera motion during integration of separate frames taken by the secondary detector. Since all the frames were taken with equal exposure times and the scene radiance did not change, the equal amount of energy was integrated on every segment. Thus, the value of $e(t)$ on each segment is inversely proportional to the length of that segment:

$$\forall \text{segment } S : \forall t \in p^{-1}(S) : e(t) = \frac{1}{\text{length}(S)}$$

This is illustrated in Fig. 12-c (note that all the rectangles have equal areas). Finally, we smooth $e(t)$, for example with spline interpolation (Fig. 12-d). The functions $p(t)$ and $e(t)$, $t \in [0, 1]$ define the motion blur PSF parametrically.

**Fig. 12. The computation of the PSF from the discrete motion vectors.**
Ref: [2].

(a) Discrete motion samples.
(b) Interpolated path $p(t)$ divided into segments by Voronoi tessellation.
(c) Energy estimation for each frame.
(d) The smoothed energy function $e(t)$.

Having the motion blur PSF in parametric form, we transform it into matrix using the algorithm given in section 2.3 and apply a deconvolution algorithm to the blurred high-resolution image. Ben-Ezra and Nayar [2] suggested using the Richardson-Lucy method since it is robust against small errors in the PSF (see section 3.4).
Ben-Ezra and Nayar [2] implemented a hybrid imaging system using the design shown in Fig. 10-a. The result of applying the deblurring algorithm to an image taken at long exposure time is shown in Fig. 13. The deblurred image shows significant improvement in image quality. Some increase of noise and deconvolution artifacts are side effects of the deconvolution algorithm. Of course, it is preferable to avoid camera motion, for example by using a tripod, or to reduce motion blur by using optically stabilized lenses. However, this is not always possible. Consider, for example aerial surveillance systems, where the vehicle translation cannot be corrected by a stabilization system.

The hybrid imaging is a possible solution for consumer level digital cameras, since it can be implemented by incorporating a low-cost chip (secondary detector) into a camera (see Fig. 10-b, c).

![Fig. 13. Experimental results of hybrid imaging motion deblurring. Ref: [2].](image)

(a) Input: the motion blurred image from the primary detector and a sequence of low-resolution images from the secondary detector.

(b) The computed PSF.

(c) Restored image. Small deconvolution artifacts are visible.

(d) Ground-truth image that was captured without motion blur using a tripod.

### 5.4 Deblurring of Moving Objects

In the previous sections we considered the problem of motion blur due to a shift-invariant camera motion. In this case the blurring function is spatial invariant over the whole image. A more complex problem is the motion blur due to an object moving in front of a stationary background (here we also assume a shift-invariant motion). The PSF is not spatial invariant in this case: only the moving object is blurred. Furthermore, the blurred object blends with the rest of the scene, so it must be separated from the background before it can be deblurred.

To solve the problem of the motion blur due to a moving object, we need to know the PSF of the moving object, the mask for the shape of the nonblurred object, and a clear, nonblurred background image. In most cases these cannot be obtained from the blurred image without
any additional information. The simplest way to obtain a clear background image, which is void of a moving object, is to capture the picture of the background when the object is not present. If the moving object is always present, one can capture a sequence of images of the scene and apply a median filter to the sequence.

To obtain a PSF and a low-resolution shape mask of the moving object, we can apply a tracking algorithm (see, for example [5]) to a sequence of low-resolution images provided by the secondary detector of a hybrid camera, designed as shown in Fig. 10-a,b. The design shown in Fig. 10-c is not suitable because its secondary detector captures only a small part of the scene. The low-resolution mask is then resampled up to a higher resolution. Though some information is lost during resampling, the obtained mask still carries enough information about the form of the moving object.

Having obtained all the items we need, we restore the image by the following operation:

\[
\tilde{f} = h_{\text{obj}} \ast^{-1} \left( g - (b \cdot h_{\text{obj}} \ast m) \right) + b \cdot \bar{m},
\]

where \( \tilde{f} \) is an approximation of the ideal image; \( g \) is a blurred input image; \( b \) is a clear background image; \( h_{\text{obj}} \) and \( m \) are the PSF and the shape mask of the moving object, respectively; \( \ast \) and \( \ast^{-1} \) denote convolution and deconvolution; \( \bar{X} \) is the complement (i.e. the inverted image) of \( X \); +, – are pixel-by-pixel addition/subtraction, and \( \cdot \) is the masking: the images are multiplied pixel-by-pixel and the result is divided by the maximal possible intensity value (often 255).

**Conclusion**

We have learned the basic methods of deconvolution and image registration, and their application to the restoration of the motion-blurred images. Finally, the hybrid imaging approach to the motion deblurring was introduced. We have seen that the negative effect of the motion blur due to a camera motion can be essentially reduced by using a hybrid camera. Such a camera can be implemented by incorporating a low-resolution chip and a deblurring function into a usual digital camera. The deblurring function can be alternatively performed by a computer when downloading images from the camera. However, this slows down the downloading process and requires storing unnecessary low-resolution image sequences on the camera flash memory. The hybrid imaging can also be applied to the improvement of pictures of fast moving objects.
References


