Image Analysis

2. Image Restoration / b. Deblurring

Lars Schmidt-Thieme

Information Systems and Machine Learning Lab (ISMLL)
Institute for Business Economics and Information Systems
& Institute for Computer Science
University of Hildesheim
http://www.ismll.uni-hildesheim.de

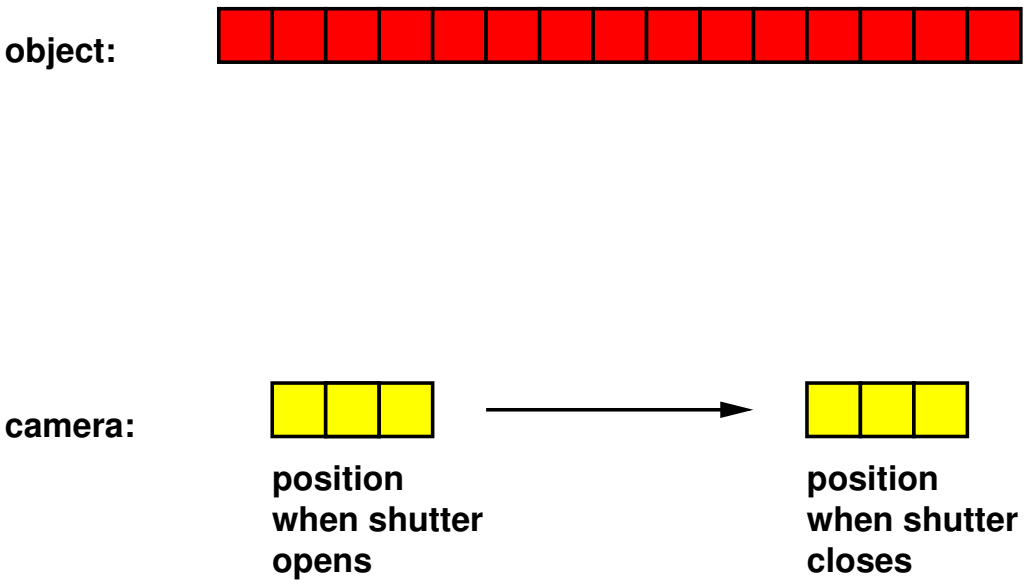
What is Motion Blur?

When camera and object move fast relative to each other,
the object can appear blurred as a rectangular area of the object contributes to an image pixel.

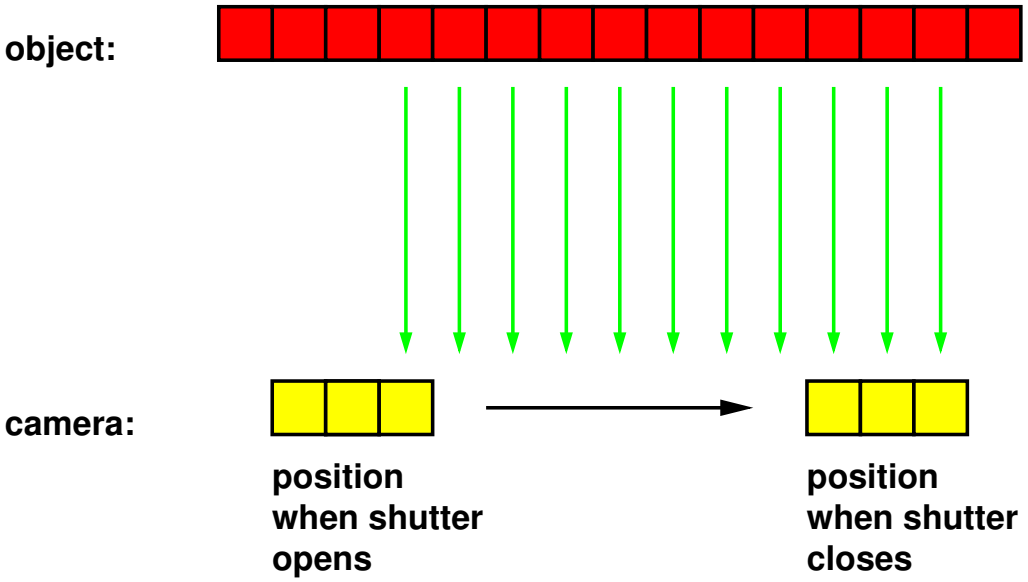
The contributing area is stretched in the direction of motion.

Example: camera moves fast
(e.g., photo taken from car passing-by):

How motion blur originates

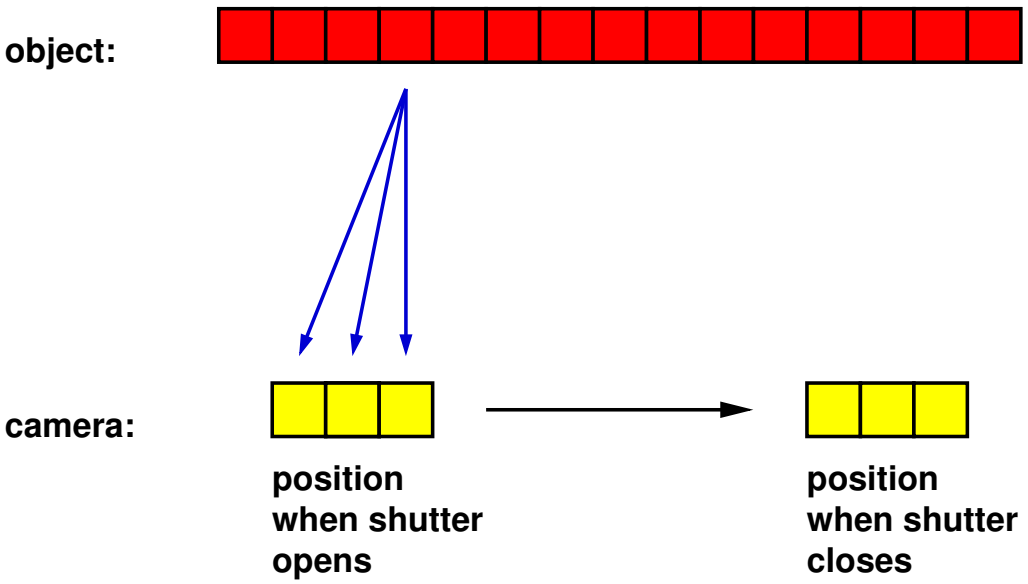


How motion blur originates



Each camera/image pixel corresponds to several object/scene “pixels”.

How motion blur originates



And vice versa:
each object/scene “pixel” contributes to several camera/image pixels.

A Model for Motion Blur

Assume the camera moves in direction of the x-axis and travels the distance w with open shutter.
Then a blurred image g originates from a scene f via:

$$g(x,y) := \frac{1}{w}\int_{x-w}^{x} f(x',y)dx'$$

If we imagine the scene to be composed of pixels, too:

$$g(x,y) := \frac{1}{w}\sum_{x'=x-w+1}^{x} f(x',y)$$

The latter equation can be used to **simulate motion blur** on already taken photos.

Undo Motion Blur

The latter equation can also be used to undo a motion blur (**deblurring**):

$$g(x,y) := \frac{1}{w}\sum_{x'=x-w+1}^{x} f(x',y)$$

$$\Longrightarrow f(x,y) = w\,g(x,y) - \sum_{x'=x-w+1}^{x-1} f(x',y)$$

and now compute recursively:

$$\hat{f}(0,y) = \quad g(0,y)$$
$$\hat{f}(1,y) = 2\,g(1,y) - \hat{f}(0,y)$$
$$\hat{f}(2,y) = 3\,g(2,y) - \hat{f}(1,y) - \hat{f}(0,y)$$
$$\vdots$$
$$\hat{f}(x,y) = (w'+1)\,g(x,y) - \sum_{x'=x-w'}^{x-1} \hat{f}(x',y), \qquad \text{with } w' := \min\{w-1, x\}$$

Example

blurred image:

unblurred:

Wrong Speed / Window Size

blurred:

window too small (10):

window optimal (20):

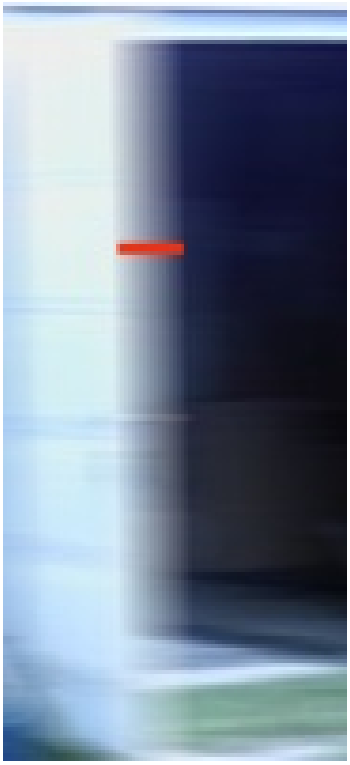
window too large (30):

Estimate Correct Window Size

To undo the blur, its window size has to be estimated correctly.

To do this manually, one can use “shadows” of sharp edges that are perpendicular to the direction of the motion.

Example: the “shadow” of the edge of the garage spreads over 20 pixels.



Out-of-Focus Blur

Besides due to motion, blur also can occur in other situations, e.g., as **out-of-focus blur** when the lens of the camera is not focused correctly on the object.

Richardson-Lucy Deconvolution [Ric72], [Luc74]

Let f be the original (unblurred) image,
k a **blur kernel** (also called a **point spread function**) and
g the blurred image:

$$g = f * k$$

If we interpret the intensity values as frequency counts, then the kernel can be interpreted as conditional probability:

$$p(g_{a,b} \mid f_{x,y}) = k_{x-a,y-b}$$

$$\begin{aligned} p(f_{x,y}) &= \sum_{(a,b)} p(f_{x,y}, g_{a,b}) = \sum_{(a,b)} p(f_{x,y} \mid g_{a,b})\, p(g_{a,b}) \\ &\overset{\text{Bayes}}{=} \sum_{(a,b)} \frac{p(g_{a,b} \mid f_{x,y})\, p(f_{x,y})}{\sum_{(x',y')} p(g_{a,b} \mid f_{x',y'})\, p(f_{x',y'})}\, p(g_{a,b}) \\ &= p(f_{x,y}) \sum_{(a,b)} \frac{k_{x-a,y-b}\, p(g_{a,b})}{\sum_{(x',y')} k_{x'-a,y'-b}\, p(f_{x',y'})} \end{aligned}$$

Richardson-Lucy Deconvolution [Ric72], [Luc74]

The resulting equation

$$p(f_{x,y}) = p(f_{x,y}) \sum_{(a,b)} \frac{k_{x-a,y-b}\, p(g_{a,b})}{\sum_{(x',y')} k_{x'-a,y'-b}\, p(f_{x',y'})}$$

can be moved from probabilites to counts:

$$f_{x,y} = f_{x,y} \sum_{(a,b)} \frac{k_{x-a,y-b}\, g_{a,b}}{\sum_{(x',y')} k_{x'-a,y'-b}\, f_{x',y'}}$$

and then be used for fixpoint interations:

$$\hat{f}^{(t+1)}_{x,y} = \hat{f}^{(t)}_{x,y} \sum_{(a,b)} \frac{k_{x-a,y-b}\, g_{a,b}}{\sum_{(x',y')} k_{x'-a,y'-b}\, \hat{f}^{(t)}_{x',y'}}$$

where the initial $\hat{f}$ is set constant to 1:

$$\hat{f}^{(0)}_{x,y} := 1 \quad \text{for all } x \text{ and } y$$

Richardson-Lucy Deconvolution [Ric72], [Luc74]

In the iteration

$$\hat{f}^{(t+1)}_{x,y} = \hat{f}^{(t)}_{x,y} \sum_{(a,b)} \frac{k_{x-a,y-b}\, g_{a,b}}{\sum_{(x',y')} k_{x'-a,y'-b}\, \hat{f}^{(t)}_{x',y'}}$$

the two sums over (a,b) and (x',y') can be limited to entries with positive kernels, e.g., if the kernel has width w, i.e.,

$$k(x,y) = 0 \quad \text{for } |x| > w \text{ or } |y| > w,$$

then

$$\hat{f}^{(t+1)}_{x,y} = \hat{f}^{(t)}_{x,y} \sum_{a=x-w}^{x+w} \sum_{b=y-w}^{y+w} \frac{k_{x-a,y-b}\, g_{a,b}}{\sum_{x'=a-w}^{a+w} \sum_{y'=b-w}^{b+w} k_{x'-a,y'-b}\, \hat{f}^{(t)}_{x',y'}}$$

and sums have to be capped at actual image width and height, i.e.,

$$a = \max\{0, x-w\}, \ldots, \min\{x+w, \text{width}\}$$
$$b = \max\{0, y-w\}, \ldots, \min\{y+w, \text{height}\}$$

Richardson-Lucy Deblurring Algorithm

deblur-richardson-lucy(blurred image g, blur kernel k, number of iterations T) :
$N := \text{width}(f) - 1, M := \text{height}(f) - 1, w := \text{kernel-width}(k)$
$\hat{f}^{(0)}(x, y) := 1$ for $x = 0, \ldots, N, y = 0, \ldots, M$
for $t := 0, \ldots, T - 1$ **do**
  for $x := 0, \ldots, N$ **do**
    for $y := 0, \ldots, M$ **do**
      $a_0 := \max\{0, x - w\}, a_1 := \min\{N, x + w\}$
      $b_0 := \max\{0, y - w\}, b_1 := \min\{M, y + w\}$
      $B := 0$
      for $a := a_0, \ldots, a_1$ **do**
        for $b := b_0, \ldots, b_1$ **do**
          $x'_0 := \max\{0, a - w\}, x'_1 := \min\{N, a + w\}$
          $y'_0 := \max\{0, b - w\}, y'_1 := \min\{M, b + w\}$
          $A := 0$
          for $x' := x'_0, \ldots, x'_1$ **do**
            for $y' := y'_0, \ldots, y'_1$ **do**
              $A := A + k(x' - a, y' - b) \cdot \hat{f}^{(t)}(x', y')$
            od
          od
          $B := B + k(x - a, y - b) \cdot g(a, b)/A$
        od
      od
      $\hat{f}^{(t+1)}(x, y) := f^{(t)}(x, y) \cdot B$
    od
  od
od

Example

blurred image:

unblurred (1st iteration):

Example

blurred image:

unblurred (2nd iteration):

Example

blurred image:

unblurred (10th iteration):

Example

blurred image:

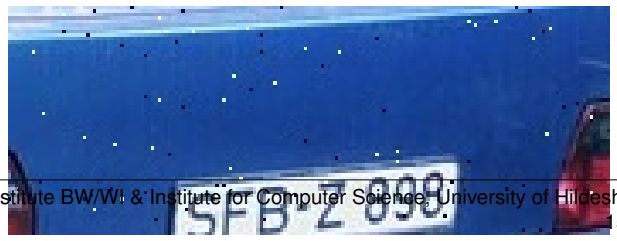
unblurred (20th iteration):

Further Readings

- See [SJA08] for a very interesting actual paper on deblurring (may be useful as topic for a Master thesis?).
- See http://www.bialith.com/Research/BARclockblur.htm (by P. J. Tadrous) for pointers to further methods and some nice examples.

Salt and Pepper Noise

Digital images often have some random pixels set to white, some others set to black (**salt and pepper noise**).



Blurring by Box Filters

Blurring can be used deliberately to reduce noise.

E.g., deblurring by a so called **box filter** with window size w:

$$g(x,y) := \frac{1}{w^2} \sum_{\Delta x=-w/2}^{w/2} \sum_{\Delta y=-w/2}^{w/2} f(x+\Delta x, y+\Delta y)$$

(The window size is usually an odd natural number.)

Blurring by Box Filters / Example (window size 3)

Blurring by Box Filters / Example (window size 7)

Blurring by Gaussian Filters (1/2)

A **Gaussian Filter** does not give the same weight to all source pixels, but weights pixels by their distance, measured in terms of the 2D Gaussian distribution:

$$\Phi_\sigma(x,y) := e^{-\frac{x^2+y^2}{2\sigma^2}}$$

where σ controlles the width (standard deviation).

$$g(x,y) := \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \Phi_\sigma(x',y')\, f(x+x', y+y')dx'dy'$$

Blurring by Gaussian Filters (2/2)

Often an interger weight matrix that approximates Φ_σ and has finite support is used, e.g.:

$$w_5 := \begin{pmatrix} 0 & 1 & 2 & 1 & 0 \\ 1 & 3 & 5 & 3 & 1 \\ 2 & 5 & 9 & 5 & 2 \\ 1 & 3 & 5 & 3 & 1 \\ 0 & 1 & 2 & 1 & 0 \end{pmatrix}$$

(conveniently indexed by -2...+2, -2...+2.)

Thus:

$$g(x,y) := \frac{1}{\sum_{\Delta x=-d/2}^{d/2} \sum_{\Delta y=-d/2}^{d/2} w(\Delta x, \Delta y)} \sum_{\Delta x=-w/2}^{w/2} \sum_{\Delta y=-w/2}^{w/2} w(\Delta x, \Delta y)\, f(x+\Delta x, y+\Delta y)$$

Blurring by Box Filters / Example (window size 5)

Blurring by Gaussian Filters / Example (window size 5)

Blurring by Mininum Filters

Random white pixels can be successfully removed by the **minimum filter**:

$$g(x,y) := \min\{f(x+\Delta x, y+\Delta y) \,|\, \Delta x = -w/2,\ldots,+w/2, \Delta y = -w/2,\ldots,+w/2\}$$

(where w is again the window size, an odd natural number.)

For color images, the pixel with the minimum brightness (sum of color intensities) is taken.

Unfortunately, black pixels get enlarged!

Blurring by Minimun Filters / Example (window size 3)

Blurring by Maximum Filters

And vice versa:
Random black pixels can be successfully removed by the **maximum filter**:

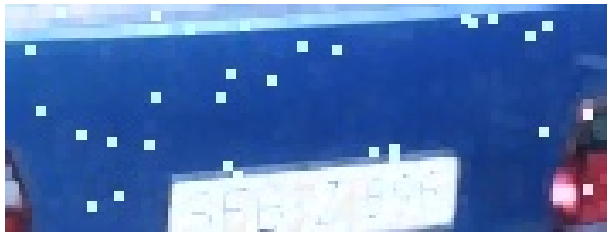
$$g(x,y) := \max\{f(x+\Delta x, y+\Delta y) \,|\, \Delta x = -w/2,\ldots,+w/2, \Delta y = -w/2,\ldots,+w/2\}$$

(where w is again the window size, an odd natural number.)

For color images, the pixel with the maximum brightness (sum of color intensities) is taken.

Unfortunately, white pixels get enlarged!

Blurring by Maximum Filters / Example (window size 3)



Blurring by Median Filters

Median filter:

$$g(x,y) := \text{median}\{f(x+\Delta x, y+\Delta y) \,|\, \Delta x = -w/2,\ldots,+w/2, \Delta y = -w/2,\ldots,+w/2\}$$

(where w is again the window size, an odd natural number.)

To compute the median of some values $x_0,\ldots,x_{N-1}$, one has to sort the values in ascending order

$$x_{(0)}, x_{(1)}, \ldots, x_{(N-1)}$$

and then return the value $x_{(N/2)}$.

For color images, pixel order is determined by their brightness (sum of color intensities).

Median filtering successfully removes salt and pepper noise.

Blurring by Median Filters / Example (window size 3)

Averaging

Several techniques we have seen so far can be uniformly described as averaging a source image with respect to some weight function, e.g.:

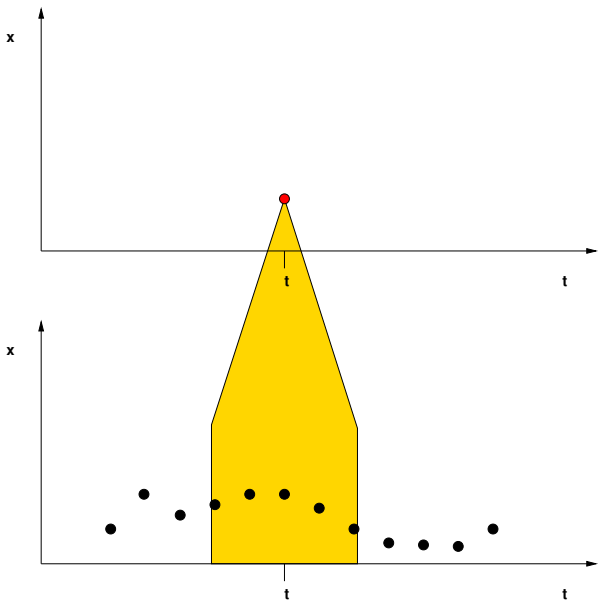
- rescaling an image,
- deblurring of motion blur,
- blurring for noise reduction.

Technically, such a one- or two-dimensional averaging is captured by the concept of a **convolution**.

Convolutions (1/5): Discrete, Unweighted, 1D

Given a time series x_t $(t = 1, \ldots, n)$, one can define a smoothed time series x', where each observation x_t is replaced by the mean of its $2k + 1$ neighbors:

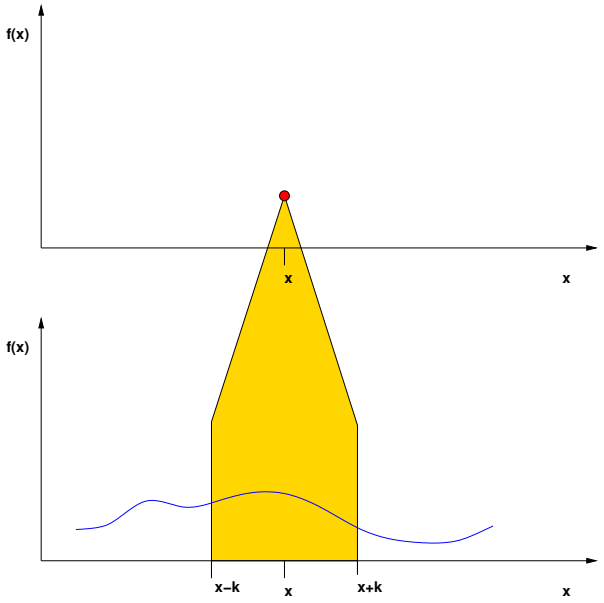
$$x'_t := \frac{x_{t-k} + x_{t-k+1} + \cdots + x_t + x_{t+1} + \cdots x_{t+k}}{2k+1} = \frac{1}{2k+1} \sum_{t'=t-k}^{t+k} x_{t'}$$



Convolutions (2/5): Continuous, Unweighted, 1D

If $f : \mathbb{R} \to \mathbb{R}$ is a function, one can define a smoothed function f', where each value $f(x)$ is replaced by the mean of the values in $[x - k, x + k]$:

$$f'(x) := \frac{1}{2k} \int_{x-k}^{x+k} f(x')dx'$$



Convolutions (3/5): Discrete, Weighted, 1D

By taking the mean, the influence of the neighbors is the same regardless of their distance. Often it is more plausible to take neighbors less into account the more far awar they are. In this case one defines **weights**:

$$w : \mathbb{Z} \to \mathbb{R}_0^+$$

where $w(d)$ specifies the influence of an observation at time distance d.

The **weighted mean** then is defined as:

$$x'_t := \frac{1}{\sum_{t' \in \mathbb{Z}} w(t' - t)} \sum_{t' \in \mathbb{Z}} w(t' - t)\, x_{t'}$$

Convolutions (3b/5): Discrete, Weighted, 1D

Simplifications:

– If w is symmetric, one often defines it just on $\mathbb{N}_0$ and writes

$$x'_t := \frac{1}{\sum_{t' \in \mathbb{Z}} w(t'-t)} \sum_{t' \in \mathbb{Z}} w(|t'-t|)\, x_{t'}$$

– If w sums to 1:

$$x'_t := \sum_{t' \in \mathbb{Z}} w(|t'-t|)\, x_{t'}$$

– If $w(d) = 0$ for $d > k$:

$$x'_t := \sum_{t'=t-k}^{t+k} w(|t'-t|)\, x_{t'}$$

The unweighted mean with window size k can be specified as special case via the weights

$$w(d) := \begin{cases} 1/(k+1), & \text{if } -k \leq d \leq k \\ 0, & \text{else} \end{cases}$$

Convolutions (4/5): Continuous, Weighted, 1D

Weights also can be introduced for averaging continuous functions:

$$w : \mathbb{R} \to \mathbb{R}_0^+ \quad \text{with} \int_{-\infty}^{+\infty} w(x)dx = 1 \text{ and } w(x) = w(-x)\ \forall x$$

The average then is defined accordingly as:

$$f'(x) := \int_{-\infty}^{+\infty} w(x'-x)\, f(x')dx' = (w * f)(x)$$

where the averaged function $w * f$ is called the **convolution of** w **and** f and defined as

$$(w * f)(x) := \int_{-\infty}^{+\infty} w(x-x')\, f(x')dx'$$

Convolutions (5/5): Continuous, Weighted, 2D

And finally, convolutions also can be defined for higher-dimensional functions, e.g., for two continuous image functions

$$f, g : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$$

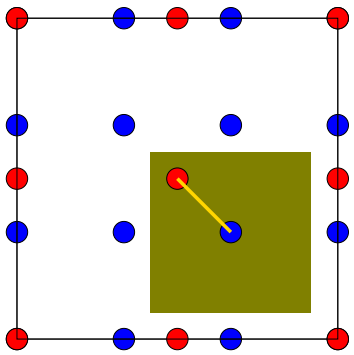
the convolution is defined as

$$(f * g)(x, y) := \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} f(x - x', y - y')\, g(x', y') dx' dy'$$

Interpolation as Convolution / Nearest Neighbor

Interpolations can be understood as convolutions, e.g., the nearest neighbor interpolation is the convolution with the weight function (usually called **kernel**)

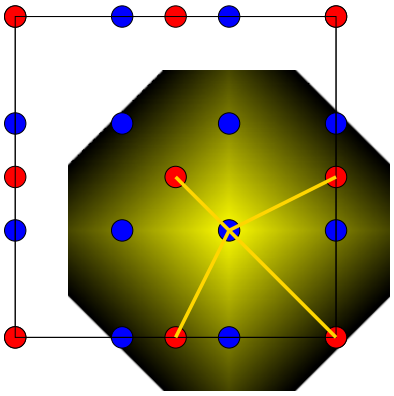
$$w(x, y) := \begin{cases} 1, & \text{if } -\frac{1}{2} \leq x \leq \frac{1}{2} \text{ and } -\frac{1}{2} \leq y \leq \frac{1}{2} \\ 0, & \text{else} \end{cases}$$



Interpolation as Convolution / Bi-Linear

The bi-linear interpolation is the convolution with the weight function

$$w(x,y) := \begin{cases} |1-x| \cdot |1-y|, & \text{if } -1 \leq x \leq 1 \text{ and } -1 \leq y \leq 1 \\ 0, & \text{else} \end{cases}$$



Blurring as Convolution

The so-called **linear filters** obviously can be understood as convolutions, i.e.,

– box filter

$$k_w(x,y) := \begin{cases} 1, & \text{if } |x| < w \text{ and } |y| < w \\ 0, & \text{else} \end{cases}$$

– Gaussian filter

$$k_\sigma(x,y) := \Phi_\sigma(x,y) := e^{-\frac{x^2+y^2}{2\sigma^2}}$$

and its finite-support approximations.

Other filters such as the minimum, maximum and median filters cannot be described as convolutions and often are called **non-linear filters**.

Deconvolutions

If a process (such as blurring) deteriorates an image f via a convolution with a kernel k

$$g = k * f$$

one can **simulate** the process by computing the convolution, e.g.,

$$\text{given } f \text{ and } k \rightsquigarrow \text{compute } g.$$

Usually one is interested in the **inverse problem**:

$$\text{given } g \text{ and } k \rightsquigarrow \text{compute } f.$$

This is often called **deconvolution**.

If the kernel k also is unknown, the problem is called **blind deconvolution** (otherwise **non-blind**).

Richardson-Lucy Deconvolution [Ric72], [Luc74]

Let f be the original (unblurred) image,
k a **blur kernel** (also called a **point spread function**) and
g the blurred image:

$$g = k * f$$

The "bayesian derivation" of the fixpoint iteration can be done in a more compact way using convolutions:

$$\begin{aligned} f &= f \cdot 1 \\ &= f \cdot (k * 1) \\ &= f \cdot (k * (g/g)) \\ &= f \cdot (k * g)/g \\ &= f \cdot (k * g)/(k * f) \end{aligned}$$

Formal Definition of Convolutions

Definition. Let $f, g : \mathbb{R} \to \mathbb{R}$ one-dimensional functions (that satisfy appropriate integrability conditions). Then the **convolution** $f * g : \mathbb{R} \to \mathbb{R}$ is defined as

$$(f * g)(x) := \int_{-\infty}^{+\infty} f(x')g(x - x')dx'$$

Let $f, g : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ two-dimensional functions (that satisfy appropriate integrability conditions). Then the **convolution** $f * g : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ is defined as

$$(f * g)(x, y) := \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} f(x', y')g(x - x', y - y')dy'dx'$$

Note: in the literature often also $f \otimes g$ is used instead of $f * g$.

Basic Properties of Convolutions

The convolution operator is **symmetric**:

$$(f * g)(x) = \int_{-\infty}^{+\infty} f(y)g(x - y)dy = \int_{-\infty}^{+\infty} f(x - y)g(y)dy = (g * f)(x)$$

(substitute $y \mapsto x - y$)

The convolution operator is **associative**:

$$\begin{aligned}(f * (g * h))(x) &= \int_{-\infty}^{+\infty} f(z) \left(\int_{-\infty}^{+\infty} g(y)h(x - z - y)dy \right) dz \\ &= \int_{-\infty}^{+\infty} \left(\int_{-\infty}^{+\infty} f(z)g(y - z)dz \right) h(x - y)dy = ((f * g) * h)(x)\end{aligned}$$

(substitute $y \mapsto y - z$)

The convolution operator is **linear**:

$$\begin{aligned}(f * (g + h))(x) &= (f * g)(x) + (f * h)(x) \\ \alpha(f * g)(x) &= ((\alpha f) * g)(x)\end{aligned}$$

with f, g, h functions and $\alpha \in \mathbb{R}$.

Dirac Distribution / Impulse

The distribution δ with
(i) $\delta(x) = 0$ for all $x \neq x_0$ and
(ii) $\int_{-\infty}^{+\infty} \delta(x)dx = 1$
is called (one-dimensional) **Dirac distribution at** x_0 (or **impulse at** x_0).

The symbol δ with
(i) $\delta(x, y) = 0$ for all $(x, y) \neq (x_0, y_0)$ and
(ii) $\int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} \delta(x, y)dxdy = 1$
is called (two-dimensional) **Dirac distribution at** (x_0, y_0) (or **impulse at** (x_0, y_0)).

Note: **distributions** (aka **generalized functions**) are a generalization of functions; see, e.g., [Vla01].

Dirac Distribution / Impulse

For a Dirac distribution δ at x_0 and a function f it is

$$(f * \delta)(x) = \int_{-\infty}^{+\infty} f(x - y)\delta(y)dy = f(x - x_0)$$

Summary

- Different forms of blur can occur such as **motion blur** and **out-of-focus blur**.
- Simple forms of motion blur (such as originating from a camera moving with constant speed) can be undone rather easily, if the parameters (direction, window length) are known.
- If the blur kernel is known, **Richardson-Lucy deconvolution** can be used for deblurring.
- Blurring often is **used to remove noise**, e.g., salt-and-pepper noise. **Median blurring** usually gives best results for this type of noise.
- Several methods such as rescaling, blurring, etc. can be understood as averaging. Averaging can formally be described by **convolutions**.

References

[Luc74] L. B. Lucy. An iterative technique for the rectification of observed distributions. *Astronomical Journal*, 79:745–+, June 1974.

[Ric72] William Hadley Richardson. Bayesian-based iterative method of image restoration. *J. Opt. Soc. Am.*, 62(1):55–59, 1972.

[SJA08] Qi Shan, Jiaya Jia, and Aseem Agarwala. High-quality motion deblurring from a single image. *ACM Transactions on Graphics (SIGGRAPH)*, 2008.

[Vla01] V. S. Vladimirov. Generalized function. In Michiel Hazewinkel, editor, *Encyclopaedia of Mathematics*. Kluwer Academic Publishers, 2001.