Image Restoration

CS 663, Ajit Rajwade

Contents

- Introduction to image restoration
- Inverse filter
- Spread spectrum filters – coded aperture camera and flutter-shutter camera
- Wiener filter – aim, assumptions, formula and derivation
- Regularized least squares deblurring
- PCA for denoising

What is image restoration?

- Image restoration = task of recovering an image from its degraded version assuming some **knowledge of the degradation phenomenon**.

- **Models** the degradation process and **inverts** it to obtain the original from the degraded (observed) image.

- Differs from image **enhancement** – which does not fully account for the nature of the degradation.

Degradation Model

$$g(x,y) = \mathcal{H}(f(x,y)) + \eta(x,y)$$

Observed image

Underlying image

noise

Forward model of the degradation process: note that this is an operator

Common Assumptions on $\mathcal{H}$:

(1) Linearity,

(2) Space Invariance

(1) $\mathcal{H}(k_1 f_1(x,y) + k_2 f_2(x,y)) = k_1 \mathcal{H}(f_1(x,y)) + k_2 \mathcal{H}(f_2(x,y)),$

(2) $\mathcal{H}(f(x - x_1, y - y_1)) = g(x - x_1, y - y_1)$

Degradation Model

- For a linear, space-invariant model, the degradation process can be modeled using a **convolution**:

$$g(x,y) = (h * f)(x,y) + \eta(x,y)$$

h is the impulse response of the system, i.e. degraded image if *f*(*x*,*y*) was a unit impulse image – also called as **convolution kernel**.

- The problem of estimating *f* from *g* and *h* is called as **deconvolution**.

Degradation Model

- Many real-world phenomena can be **approximated** as linear and space-invariant.
- Non-linear and space-variant models are more accurate, more general but more complex.
- Even with the simplifying assumption of linearity and space-invariance, we will see that inverting the degradation model has many challenges.

Models of Blur

- In image restoration, the most commonly encountered problem is that of **blur** removal given a known blur model.

- An image is said to be **blurred** when it is convolved with a low-pass filter of a certain kind.

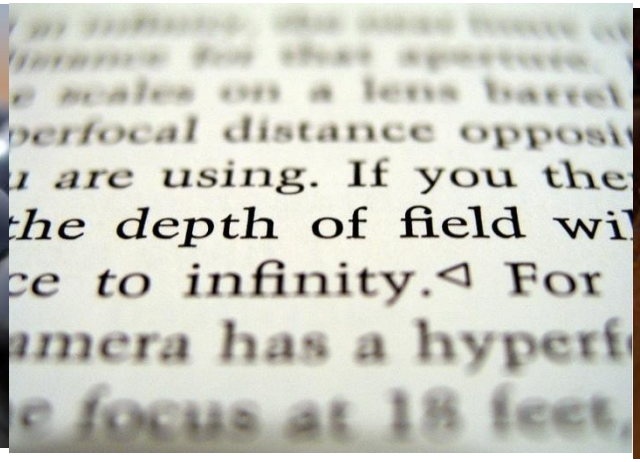
Models of Blur

- Defocus blur

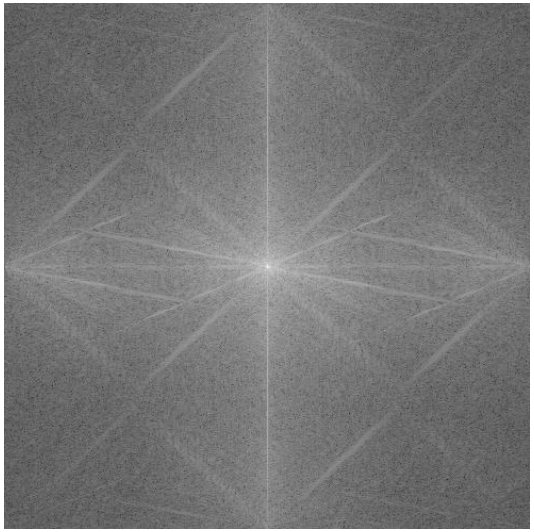
- Motion Blur

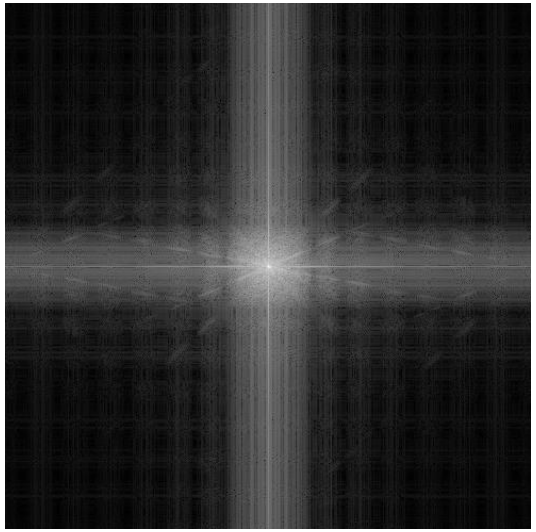
Defocus Blur

- Occurs when the scene being observed is not in focus. It is actually spatially variant dependent on the depth of each point (i.e. its distance from the camera), but we will model it here as spatially uniform for simplicity.
- Its frequency response is: $H(u,v) \propto e^{-(u^2+v^2)/\sigma^2}$



Gaussian-blurred image.
Kernel size 15 x 15.
Blur level: 5





Here we are showing plots of log(magnitude of Fourier transform +1) for easy visualization. We will call them log-Fourier plots

Log-Fourier plot of original image

Log-Fourier plot of Gaussian-blurred image

Motion blur

- A commonly occurring form of blur – when there is relative motion between the camera and the object/scene being imaged – during the process of image acquisition.

http://blog.londolozi.com/2014/01/how-to-capture-motion-blur-in-a-photograph/

https://cs.nyu.edu/~fergus/papers/deblur_fergus.pdf

Motion Blur

- A camera gathers the image of a scene as follows:

✓ Light from the scene enters the camera during the exposure time, i.e. when the shutter is open.

✓ The light passes through the lens and hits a sensor array (CCD array)

✓ The CCD array performs an integration operation during the entire exposure time.

✓ The image is formed on the CCD array after the shutter closes.

Motion Blur

- Imagine an object undergoing motion parallel to the plane of the camera sensor array.

- Let the motion be translation (for simplicity) given by $x_0(t)$ and $y_0(t)$, i.e. the motion is a function of time.

- Let $f(x,y)$ be the intensity at point (x,y) of the true (underlying) image.

Motion Blur

- Then the observed image is given by:

$$g(x,y) = \int_0^T f(x - x_0(t), y - y_0(t))dt$$

$$G(u,v) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} g(x,y)e^{-j2\pi(ux+vy)}dxdy$$

$$= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_0^T f(x - x_0(t), y - y_0(t))dt\, e^{-j2\pi(ux+vy)}dxdy$$

$$= \int_0^T \{\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x - x_0(t), y - y_0(t))\, e^{-j2\pi(ux+vy)}dxdy\}\,\mathrm{dt}$$

$$= \int_0^T F(u,v)\, e^{-j2\pi(ux_0(t)+vy_0(t))}dt$$

$$= F(u,v)\int_0^T e^{-j2\pi(ux_0(t)+vy_0(t))}\, dt = F(u,v)H(u,v)$$

Motion Blur

- Let us assume that:

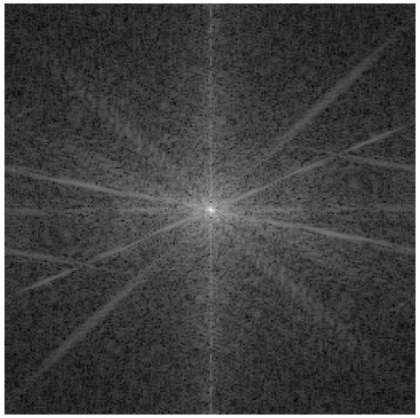
$$x_0(t) = at/T, y_0(t) = bt/T$$

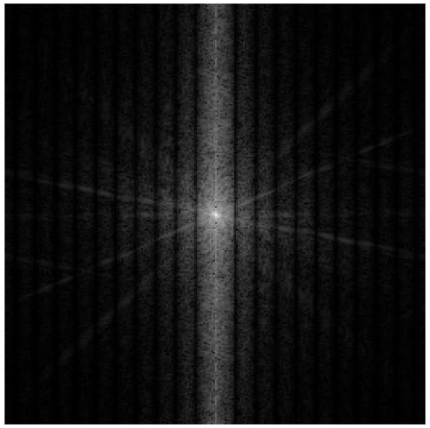
- Then the blur frequency response is:

$$H(u,v) = \int_0^T e^{-j2\pi(ux_0(t)+vy_0(t))}\, dt = \int_0^T e^{-j2\pi(au+bv)t/T} dt$$

$$= \frac{T}{\pi(au+bv)} \sin(\pi(au+bv)) e^{-j\pi(au+bv)}$$

motion blurred image (motion in X direction)





Here we are showing plots of log(magnitude of Fourier transform +1) for easy visualization. We will call them log-Fourier plots

Log-Fourier plot of original image

Log-Fourier plot of motion-blurred image (notice the strong response in the vertical direction and the sinc-like pattern parallel to the X axis!)

Example: Restoration by Inverse Filtering

Known (observed)

Known

Unknown (to be estimated)

$$g(x,y) = (h * f)(x,y),$$

$$G(u,v) = H(u,v)F(u,v)$$

$$\therefore \hat{F}(u,v) = G(u,v)/H(u,v) = F(u,v)$$

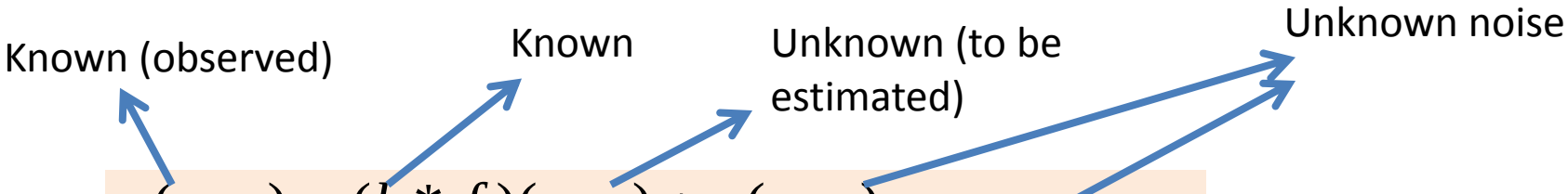
$$\hat{f}(x,y) = \mathcal{F}^{-1}(\hat{F}(u,v))$$

Assume no noise (for now)

Convolution theorem

If **H(u,v)** is zero, there is a problem in estimating **F(u,v)**. Otherwise this task is completely well-defined.

Example: Restoration by Inverse Filtering



$$g(x,y) = (h * f)(x,y) + \eta(x,y),$$

$$G(u,v) = H(u,v)F(u,v) + N(u,v)$$

$$\hat{F}(u,v) = G(u,v)/H(u,v)$$

$$F(u,v) = G(u,v)/H(u,v) - N(u,v)/H(u,v)$$

$$\hat{F}(u,v) \neq F(u,v)$$

- If *H(u,v)* has small values (this will happen for higher frequencies, i.e. higher values of *u* and *v*, if *h(x,y)* is a blur kernel, i.e. a low-pass filter), the corresponding estimates of *F(u,v)* will be hugely erroneous if there is even a tiny amount of noise.

- This is especially because the Fourier Transform of the noise, i.e. *N(u,v)* (an unknown quantity), may be greater than *F(u,v)* for high values for high *u* and *v* (**why?**) H is low pass so at high frequency H will be 0 or close to zero but N/h will be high high relative to the GU at that frequency frequency

ORIGINAL image

BLURRED IMAGE

RESTORED IMAGE

Image restored using Inverse filter with no noise (ideal, non-realistic scenario)

RESTORED IMAGE UNDER NOISE

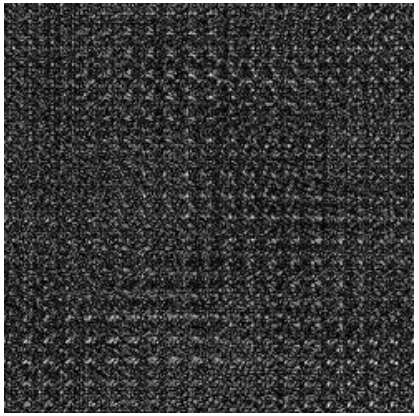


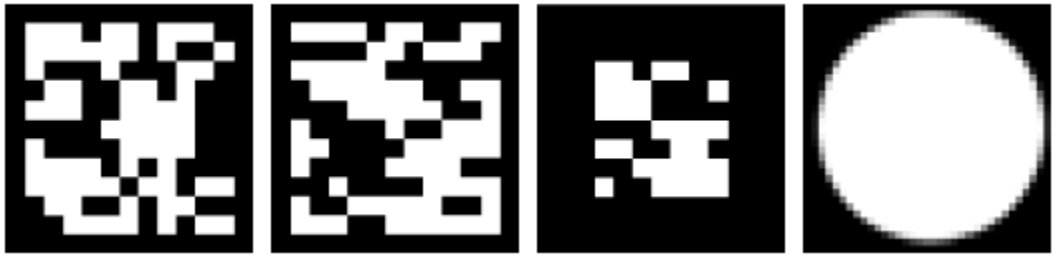
Image restored using Inverse filter with 0.1% noise

RESTORED IMAGE UNDER NOISE + LPF

Image restored using Inverse filter with 0.1% noise followed by a low-pass filter

Blurring with Holes

- Let us say we put in a cardboard piece with holes inside the camera aperture.
- The defocus blur can no more approximated as a Gaussian function.
- Rather, the blur kernel is now represented as a Gaussian dot-multiplied with a binary pattern (with values of 1 – wherever there was a hole and a 0 wherever there was no hole).



Source of images:
http://giga.cps.unizar.es/~diegog/ficheros/pdf_papers/coded_Masia.pdf

Figure 1: Left: *Images of the response to a point light of different apertures* (from top to bottom: *focused aperture, defocused circular aperture -defocus depth = 90 cm- and one of our coded apertures -defocus depth = 90 cm-, shown in the right). A LED and black cardboard were used to create the point light.* Right: *Canon EF 50mm f/1.8 lens with one of our coded apertures.*

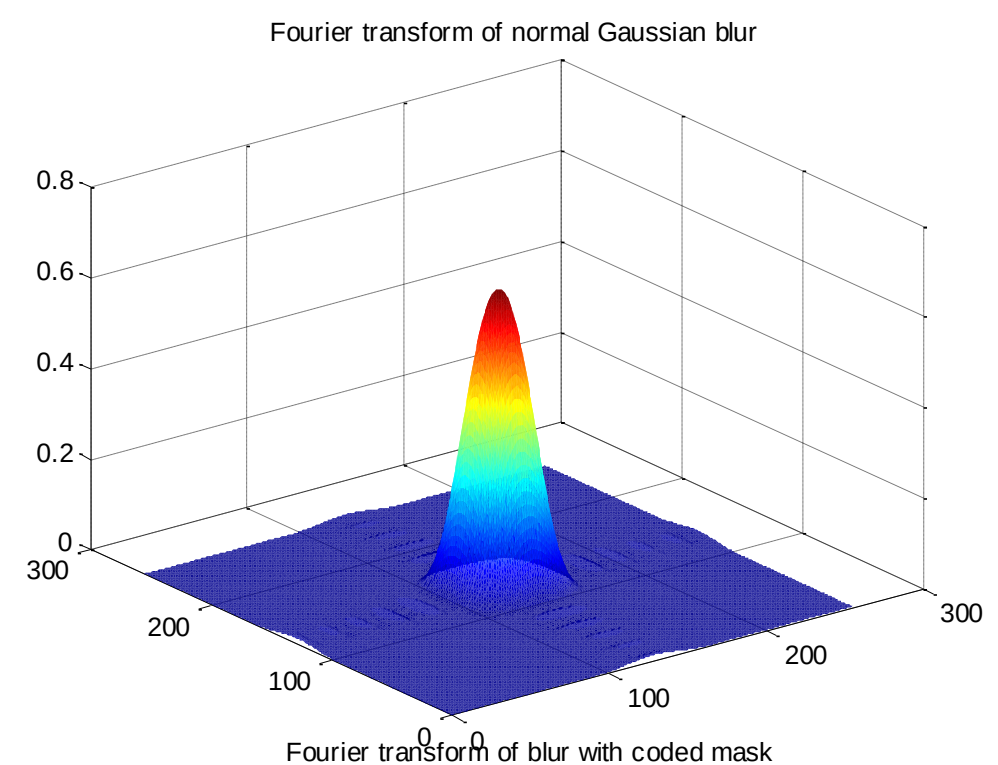
ORIGINAL image

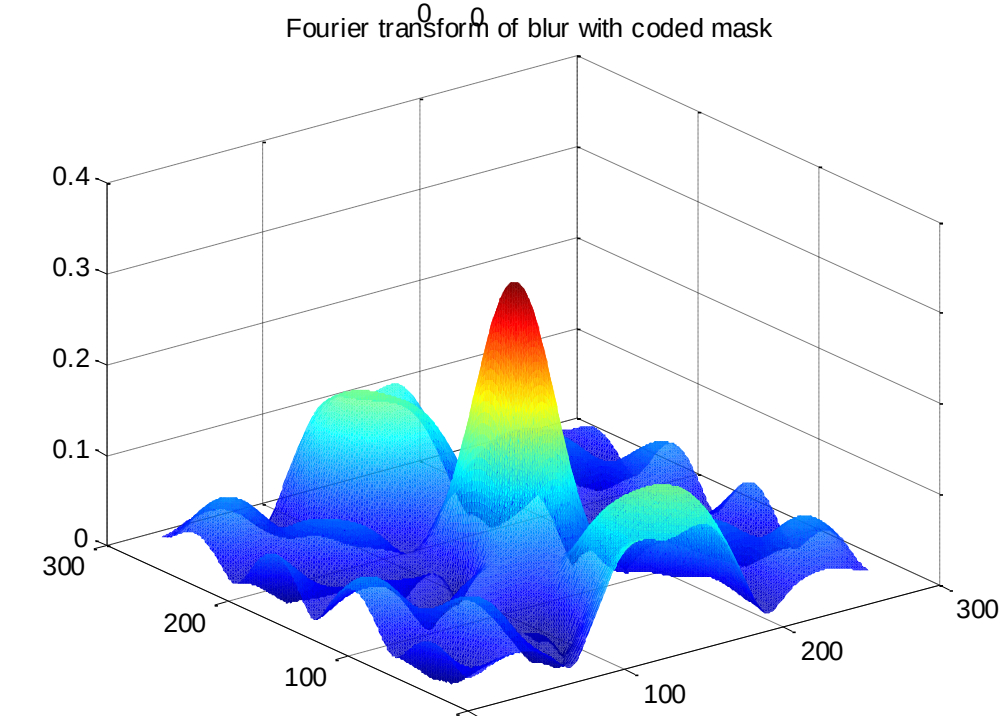
CODED MASK: BLURRED IMAGE

CODED MASK: RESTORED IMAGE

CODED MASK: RESTORED IMAGE UNDER NOISE

Using a coded mask, the restored image is of high quality even under noise (same 0.1% as before). Why does this happen?

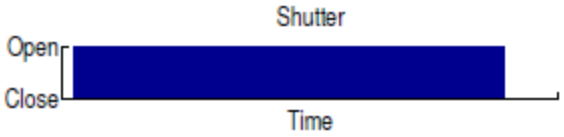




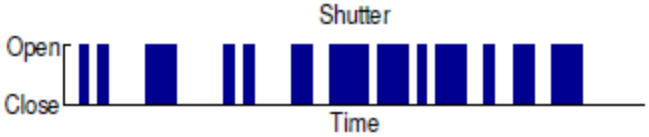
The coded mask preserves higher frequencies. It also is a spread-spectrum kernel, i.e. it is not a pure low-pass filter. Hence its higher frequency components have larger amplitudes and division (in the inverse filter) does not blow up the noise.

Flutter Shutter Camera

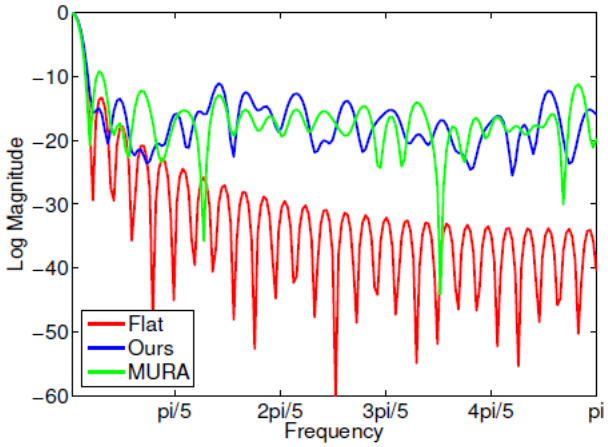
- The same principle is used in the flutter shutter camera to deal with motion blur.
- The shutter of a normal camera is usually open throughout the exposure duration (denoted by *T* in the derivation for motion blur).
- This is equivalent to convolution with a temporal box filter (a low-pass filter).
- In a **flutter-shutter camera**, the shutter is made to flutter (open and close) during the exposure time - as per a randomly generated binary sequence.



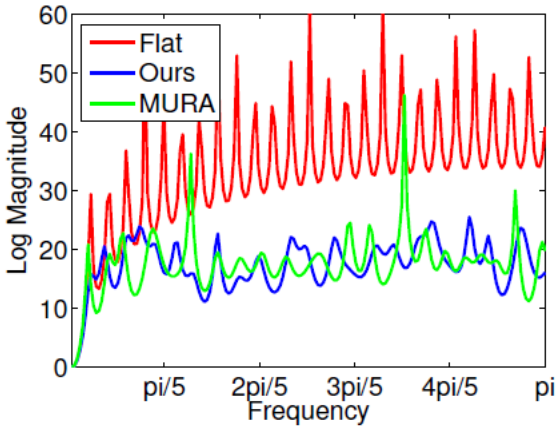
Traditional Exposure



Coded Exposure



(a) Linear Convolution Filter DFT



(b) Decoding Filter DFT

Source of images: http://web.media.mit.edu/~raskar/deblur/

Frequency response of motion blur kernel in a flutter-shutter camera (blue curve) versus traditional camera (red curve). The green curve corresponds to a different camera (we are not studying it here).

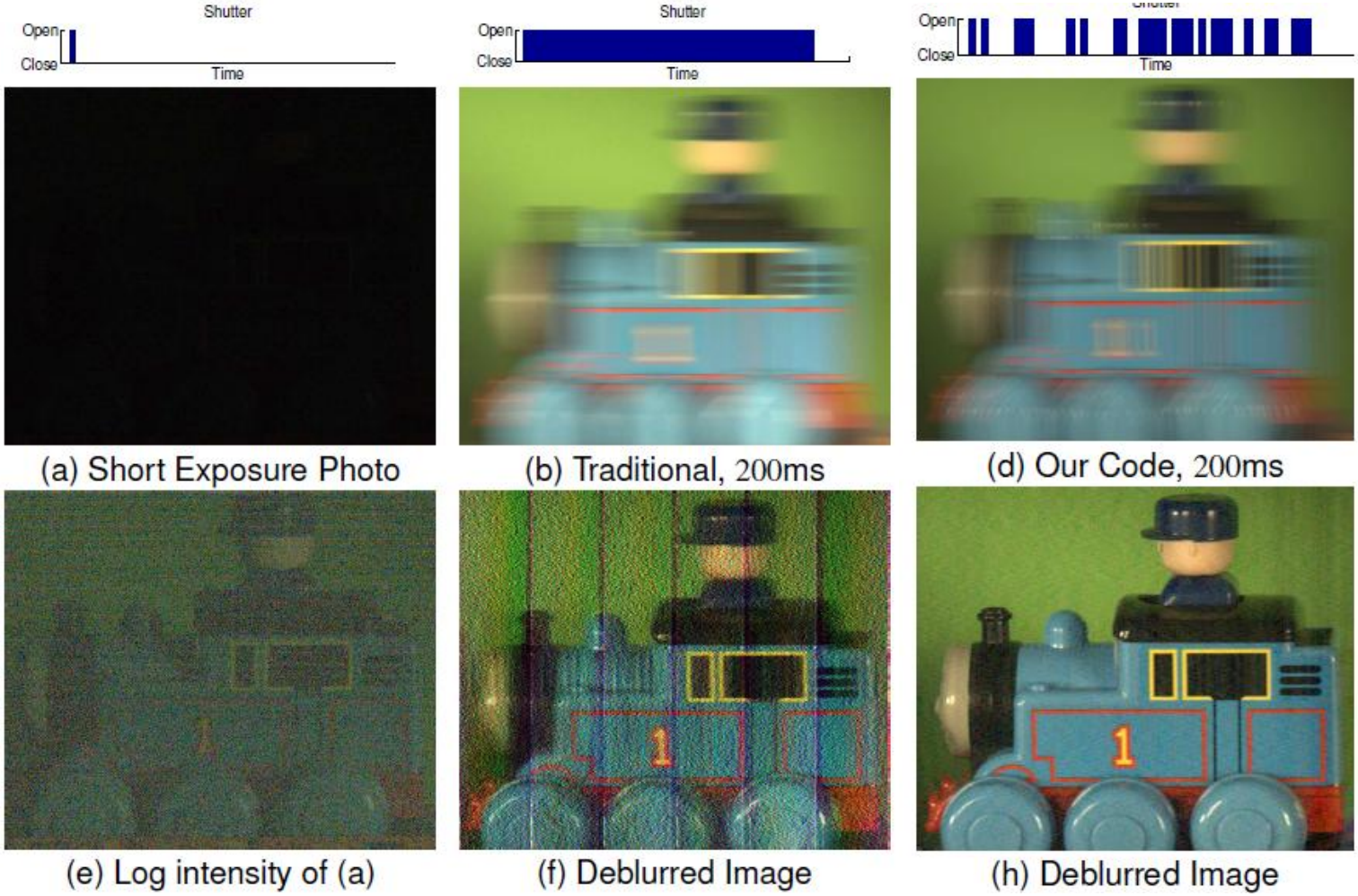
(a) Blurred Image

(b) Rectified Crop

(c) Deblurred Image

Figure 1: Coded exposure enables recovery of fine details in the deblurred image. (a) Photo of a fast moving vehicle. (b) User clicks on four points to rectify the motion lines and specifies a rough crop. (c) Deblurred result. Note that all sharp features on the vehicle (such as text) have been recovered.

Source of images: http://web.media.mit.edu/~raskar/deblur/



Source of images: http://web.media.mit.edu/~raskar/deblur/

A word of caution

- Spread spectrum filters and cameras using them do not “solve” the problem of the inverse filter.
- They merely make use of a spread spectrum filter to **work around** the issues with the inverse filter by stabilizing the deblurring process.
- These cameras by design allow less light to enter the camera during image acquisition and are not superior to normal cameras for all applications. The images acquired by these cameras may appear grainier due to lower signal to noise ratio.
- But they address one particular application, i.e. deblurring in a very principled way.
- Spread spectrum filters may not be available in many different applications – such as motion blur in an image acquired by a typical camera.

Wiener Filter

- Spread spectrum filters are not always possible in many applications.
- The inverse filter approach on previous slides made no explicit use of the knowledge of the noise model.
- The Wiener filter is one approach which makes use of knowledge of the **statistical properties of the noise** besides the degradation function.
- It attempts to remove both noise as well as the blur.

Wiener filter

- Its aim is to produce an *estimate* of the underlying image such that the **expected mean square error** between the true and estimated images is minimized:

$$e^2(\hat{f}(x,y)) = E(\iint (f(x,y) - \hat{f}(x,y))^2 dxdy)$$

$$= E(\iint (F(u,v) - \hat{F}(u,v))^2 dudv) \text{ by Parseval's theorem}$$

Wiener filter

- It asks the questions: which **linear space-invariant** filter shall I apply to the degraded image (i.e. with which filter shall I **convolve** the degraded image) to produce an estimate that is as close to the original as possible, in a least squares sense, on an **average**?
- It minimizes the blow-up of noise during deconvolution, especially at frequencies where the signal to noise ratio is very poor (i.e. low).

Wiener filter

- Assumptions made by the Wiener filter:

✓ Noise is independent of the image,

✓ Noise statistics do not change spatially

✓ Either the image or the noise (or both) has (have) zero mean

✓ Second order statistics of the image and the noise are known

- Wiener filter is also called as a **minimum mean square error filter** due to the criterion it optimizes.

Wiener filter

$$g(x,y) = (h * f)(x,y) + \eta(x,y),$$
$$G(u,v) = H(u,v)F(u,v) + N(u,v)$$

Estimate of the Fourier transform of f

$$\hat{F}(u,v) = \frac{H^*(u,v)S_f(u,v)}{|H(u,v)|^2 S_f(u,v) + S_\eta(u,v)} G(u,v)$$

$$= \frac{H^*(u,v)}{|H(u,v)|^2 + S_\eta(u,v)/S_f(u,v)} G(u,v)$$

$$= \boxed{\frac{1}{H(u,v)} \frac{|H(u,v)|^2}{|H(u,v)|^2 + S_\eta(u,v)/S_f(u,v)}} G(u,v)$$

Power spectrum of original signal = $|F(u,v)|^2$

Wiener filter

Power spectrum of noise = $|N(u,v)|^2$

Power spectrum means magnitude-squared of the Fourier transform.
Frequency spectrum means magnitude of the Fourier transform.

Noise to signal ratio (or inverse signal to noise ratio – ISNR)

$$\hat{F}(u,v) = \frac{1}{H(u,v)} \frac{|H(u,v)|^2}{|H(u,v)|^2 + S_\eta(u,v)/S_f(u,v)} G(u,v)$$

$$= \frac{1}{H(u,v)} \frac{1}{1 + \frac{S_\eta(u,v)}{|H(u,v)|^2 S_f(u,v)}} G(u,v)$$

Deblurring filter

Denoising filter

• At frequencies (*u*,*v*) where the signal is much stronger than the noise, the ISNR is 0, and the Wiener filter reduces to the inverse filter.

• At frequencies (*u*,*v*) where the signal is much weaker, the ISNR will be large and the corresponding component *G(u,v)* will be attenuated (note that the Wiener filter cannot reconstruct such components well!)

•When there is no blurring, but only noise, we have:

$$\hat{F}(u,v) = \frac{1}{1 + S_\eta(u,v)/S_f(u,v)} G(u,v), \because H(u,v) = 1$$

Wiener filter

- The Wiener filter requires us to know the ISNR at different frequencies.
- We usually do not have knowledge of this quantity, but we can provide some *estimate* (or guess-timate).
- For example, in most (photographic) images, the ISNR is low at lower frequencies and high at higher frequencies.
- Why? Images typically have high energy at lower frequencies, and low energy at higher frequencies. But noise is typically spread-spectrum.

Derivation of Wiener filter in deconvolution

http://en.wikipedia.org/wiki/Wiener_deconvolution

$$g(x,y) = h(x,y) * f(x,y) + \eta(x,y)$$
$$G(u,v) = H(u,v)F(u,v) + N(u,v)$$
$$\hat{F}(u,v) = L(u,v)G(u,v)$$

We want to find a linear filter whose Fourier transform is ***L(u,v)*** and which minimizes the following expected value.

$$E((F(u,v) - \hat{F}(u,v))^2) = E(|F(u,v) - L(u,v)G(u,v)|^2)$$
$$= E(|F(u,v) - L(u,v)(H(u,v)F(u,v) + N(u,v))|^2)$$
$$= E(|F(u,v) - L(u,v)H(u,v)F(u,v) - L(u,v)N(u,v)|^2)$$
$$= E(|F(u,v)|^2|1 - L(u,v)H(u,v)|^2) + E(|L(u,v)|^2|N(u,v)|^2)$$
$$- E((F(u,v) - L(u,v)H(u,v)F(u,v))L^*(u,v)N^*(u,v))$$
$$- E(L(u,v)N(u,v)(F(u,v) - L(u,v)H(u,v)F(u,v))^*)$$
$$= S_f^2(u,v)|1 - L(u,v)H(u,v)|^2 + |L(u,v)|^2 S_n^2(u,v) + 0 + 0$$

Both these terms are 0 because the image and the noise are uncorrelated and one or both have zero mean. So E(F(u,v)N*(u,v)) = E(N(u,v)F*(u,v)) = 0.

$E(|F(u,v)|^2) = Sf^2(u,v)$

$E(|N(u,v)|^2) = S\eta^2(u,v)$
$E(|L(u,v)|^2) = |L(u,v)|^2$ as L is not a random variable

Derivation of Wiener filter in deconvolution

http://en.wikipedia.org/wiki/Wiener_deconvolution

Note: as the image and noise are uncorrelated, we have :

$$E((F(u,v)-\mu_F)(N(u,v)-\mu_N)^*)=0$$

Either the image or the noise has zero mean, let us assume $\mu_N = 0$.

$$\therefore E((F(u,v)-\mu_F)N(u,v)^*)=0$$

$$\therefore E(F(u,v)N(u,v)^*)-E(\mu_F N(u,v)^*)=0$$

$$\therefore E(F(u,v)N(u,v)^*)-\mu_F E(N(u,v)^*)=0$$

$$\therefore E(F(u,v)N(u,v)^*)=0 \text{ as } E(N(u,v)^*)=\mu_N^*=0$$

Derivation of Wiener filter in deconvolution

http://en.wikipedia.org/wiki/Wiener_deconvolution

Taking complex derivative w.r.t. $L(u,v)$ and setting it to 0, we get :

$$S_f^2(u,v)(1-L(u,v)H(u,v))*(-H(u,v))+L*(u,v)S_\eta^2(u,v)=0$$

$$\therefore L*(u,v)=\frac{S_f^2(u,v)H(u,v)}{S_\eta^2(u,v)+S_f^2(u,v)H^2(u,v)} \rightarrow L(u,v)=\frac{S_f^2(u,v)H*(u,v)}{S_\eta^2(u,v)+S_f^2(u,v)H^2(u,v)}$$

$$\therefore L(u,v)=\frac{H*(u,v)}{\frac{S_\eta^2(u,v)}{S_f^2(u,v)}+H^2(u,v)}$$

Note: If z is a complex variable, then

$$\frac{\mathrm{d}}{\mathrm{dz}}\left(zz^*\right)=z^*$$

$$\frac{\mathrm{d}}{\mathrm{dz}}(.)=\frac{1}{2}\left(\frac{\partial}{\partial x}(.)-j\frac{\partial}{\partial y}(.)\right)$$

Interactive Wiener filter

- As the ISNR is unknown, we can substitute it by a constant *K*, which we can choose interactively based on visual appeal.

$$\hat{F}(u,v) = \frac{1}{H(u,v)} \frac{|H(u,v)|^2}{|H(u,v)|^2 + K} G(u,v)$$

- Look at figures 5.28 and 5.29 of the book (pages 355, 356) for examples of results using this method.

a b c

FIGURE 5.28 Comparison of inverse- and Wiener filtering. (a) Result of full inverse filtering of Fig. 5.25(b). (b) Radially limited inverse filter result. (c) Wiener filter result.

Image taken from the Book by Gonzalez and Woods

a b c
d e f
g h i

FIGURE 5.29 (a) Image corrupted by motion blur and additive noise. (b) Result of inverse filtering. (c) Result of Wiener filtering. (d)–(f) Same sequence, but with noise variance one order of magnitude less. (g)–(i) Same sequence, but noise variance reduced by five orders of magnitude from (a). Note in (h) how the deblurred image is quite visible through a "curtain" of noise.

Image taken from the Book by Gonzalez and Woods

a b c
d e f
g h i

FIGURE 5.29 (a) Image corrupted by motion blur and additive noise. (b) Result of inverse filtering. (c) Result of Wiener filtering. (d)–(f) Same sequence, but with noise variance one order of magnitude less. (g)–(i) Same sequence, but noise variance reduced by five orders of magnitude from (a). Note in (h) how the deblurred image is quite visible through a "curtain" of noise.

Image taken from the Book by Gonzalez and Woods

Original Image (courtesy of MIT)

Blurred Image

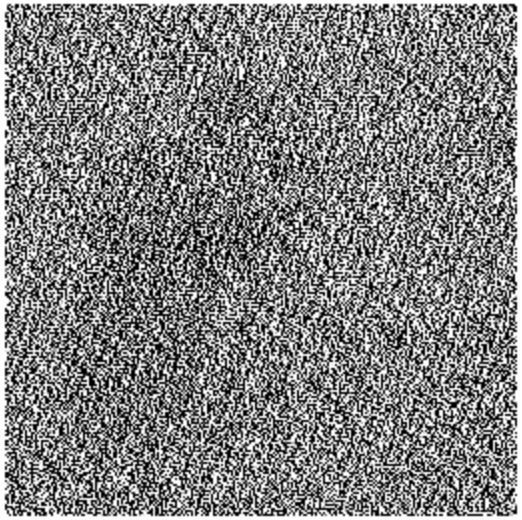
Restored Image

http://www.mathworks.in/help/images/examples/deblurring-images-using-a-wiener-filter.html

Original Image (courtesy of MIT)

Simulate Blur and Noise

Restoration of Blurred, Noisy Image Using NSR = 0



Restoration of Blurred, Noisy Image Using Estimated NSR

http://www.mathworks.in/help/images/examples/deblurring-images-using-a-wiener-filter.html

Regularized Restoration

- We know that the inverse filter is unstable if there is noise in the measured image.
- The inverse filter (given a fixed blur kernel) comes from the least squares criterion.

$$f = \arg\min_f \sum_{x,y} \left(g(x,y) - (h * f)(x,y)\right)^2$$

$$\leftrightarrow F = \arg\min_F \sum_{u,v} \left(G(u,v) - H(u,v)F(u,v)\right)^2$$

$$\leftrightarrow \hat{F}(u,v) = \arg\min_{F(u,v)} \left(G(u,v) - H(u,v)F(u,v)\right)^2$$

$$\leftrightarrow \hat{F}(u,v) = \frac{G(u,v)}{H(u,v)}$$

Regularized Restoration

- Now we modify our criterion.

- We say that in addition to a least squares solution, we want a solution (i.e. an image) whose **derivatives are not allowed to be arbitrarily large**.

Regularized Restoration

- Let's pick the Laplacian as our derivative operator:

$$\Delta f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \approx f(x+1, y) + f(x-1, y) + f(x, y+1) + f(x, y-1) - 4f(x, y)$$

- You know that the Laplacian can be represented by a convolution with the mask:

$$p = \begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}, \Delta f = p * f$$

Regularized Restoration

- The Fourier transform of the Laplacian of the image is given by $P(u,v)F(u,v)$.

- Here $P(u,v)$ is the Fourier transform of an appropriately zero-padded version of the convolution kernel.

Regularized Restoration

- Hence the frequency response (i.e. Fourier transform) of the desired filter is obtained as follows:

$$\hat{f} = \arg\min_f \sum_{x,y} \left(g(x,y) - (h * f)(x,y)\right)^2 + \gamma\left(f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) - 4f(x,y)\right)^2$$

$$\hat{F} = \arg\min_F \sum_{u,v} \left(G(u,v) - H(u,v)F(u,v)\right)^2 + \gamma\left(P(u,v)F(u,v)\right)^2$$

$$\hat{F}(u,v) = \arg\min_{F(u,v)} \left(G(u,v) - H(u,v)F(u,v)\right)^2 + \gamma\left(P(u,v)F(u,v)\right)^2$$

$$\rightarrow \hat{F}(u,v) = \frac{G(u,v)H^*(u,v)}{H^2(u,v) + \gamma P^2(u,v)}$$

The second step follows from Parseval's theorem

Regularized Restoration

- How to pick the parameter ϒ?
- Method 1: Pick it interactively till you get a **visually pleasing** result.
- Method 2: Start with some initial value of ϒ. Compute a residual vector $r = g - H\hat{f}$. Compute its squared norm $\|r\|^2 / M$. If it is too far away from the **noise variance**, readjust ϒ. Otherwise stop. Note: in many applications, the noise variance can be estimated,.

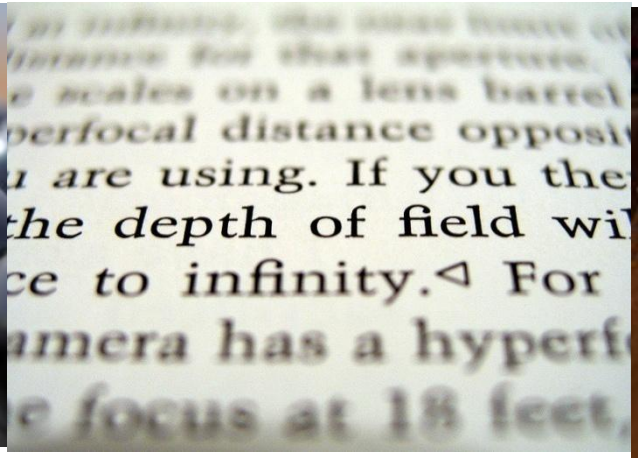
Blur is not always bad

Motion blur conveys a sense of speed:
can be used to estimate direction of motion of a moving object (or direction of camera motion) from a still image

Aesthetic effects!:

http://www.smashingmagazine.com/2008/08/24/45-beautiful-motion-blur-photos/

Blur is not always bad



Variation in blur conveys depth information

Aesthetic blur: Bokeh
http://en.wikipedia.org/wiki/Bokeh

PCA-based denoising

Using PCA for image denoising

- You have seen the non-local means (NLMeans) algorithm (patch-based filtering).
- It uses the fact that natural images have a great deal of redundancy – i.e. several patches in distant regions of the image can be very similar.

$$\hat{I}(x_i, y_i) = \frac{\sum_{x_j, y_j} w_j I(x_j, y_j)}{\sum_{x_j, y_j} w_j}$$

$$w_j = \exp\left(-\beta \| I^{(0-)}_{\text{patch}}(x_i, y_i) - I^{(0-)}_{\text{patch}}(x_j, y_j) \|^2\right)$$

Difference between patches

Use of PCA for denoising

- This **“non-local”** principle can be combined with PCA for denoising.
- Consider a clean image I corrupted by additive Gaussian noise of mean zero and standard deviation σ, to give noisy image J as follows:

  $J = I + N$, N ~ Gaussian distribution of mean 0 and standard deviation σ.
- Given J, we want to estimate I, i.e. we want to denoise J.

Use of PCA for denoising

- Consider a small p x p patch – denoted $\mathbf{q}_{ref}$ - in J.
- Step 1: We will collect together some L patches $\{\mathbf{q}_1,\mathbf{q}_2,\ldots,\mathbf{q}_L\}$ from J that are structurally *similar* to $\mathbf{q}_{ref}$ – pick the L nearest neighbors of $\mathbf{q}_{ref}$.
- Note: even if J is noisy, there is enough information in it to judge similarity if we assume $\sigma \ll$ average intensity of the true image I.
- Step 2: Assemble these L patches into a matrix of size p^2 x L. Let us denote this matrix as $\mathbf{X}_{ref}$.

Use of PCA for denoising

- Step 3: Find the eigenvectors of $\mathbf{X}_{ref}\mathbf{X}_{ref}^{T}$ to produce an eigenvector matrix **V.**
- Step 4: Project each of the (noisy) patches $\{\mathbf{q}_1,\mathbf{q}_2,\ldots,\mathbf{q}_L\}$ onto **V** and compute their eigen-coefficient vectors denoted as $\{\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2,\ldots, \boldsymbol{\alpha}_L\}$ where $\boldsymbol{\alpha}_i = \mathbf{V}^T\mathbf{q}_i$.
- Step 5: Now, we need to manipulate the eigencoefficients of $\mathbf{q}_{ref}$ in order to denoise it.

Use of PCA for denoising

- Step 5 (continued): We will follow a Wiener filter type of update:

$$\boldsymbol{\beta}_{\mathbf{ref}}(l) = \frac{1}{1 + \frac{\sigma^2}{\bar{\boldsymbol{\alpha}}^2(l)}} \boldsymbol{\alpha}_{\mathbf{ref}}(l), 0 \le l \le p^2 - 1$$

σ^2 → Noise variance (assumed known)

$\bar{\boldsymbol{\alpha}}^2(l)$ → Estimate of coefficient squared of true signal

Note : $\boldsymbol{\alpha}_{\mathbf{ref}}$ is a vector of eigencoeff icients of the reference (noisy) patch and contains p^2 elements, of which the l - th element is $\boldsymbol{\alpha}_{\mathbf{ref}}(l)$. $\boldsymbol{\beta}_{\mathbf{ref}}$ is the vector of eigencoeff icients of the filtered patch.

$$\bar{\boldsymbol{\alpha}}^2(l) = \max\left(0, \frac{1}{L}\sum_{i=1}^{L} \alpha_i^2(l) - \sigma^2\right)$$

Why this formula? We will see later.

- Step 6: Reconstruct the reference patch as follows: $\mathbf{q}_{ref}^{denoised} = \mathbf{V}\boldsymbol{\beta}_{\mathbf{ref}}$

Use of PCA for denoising

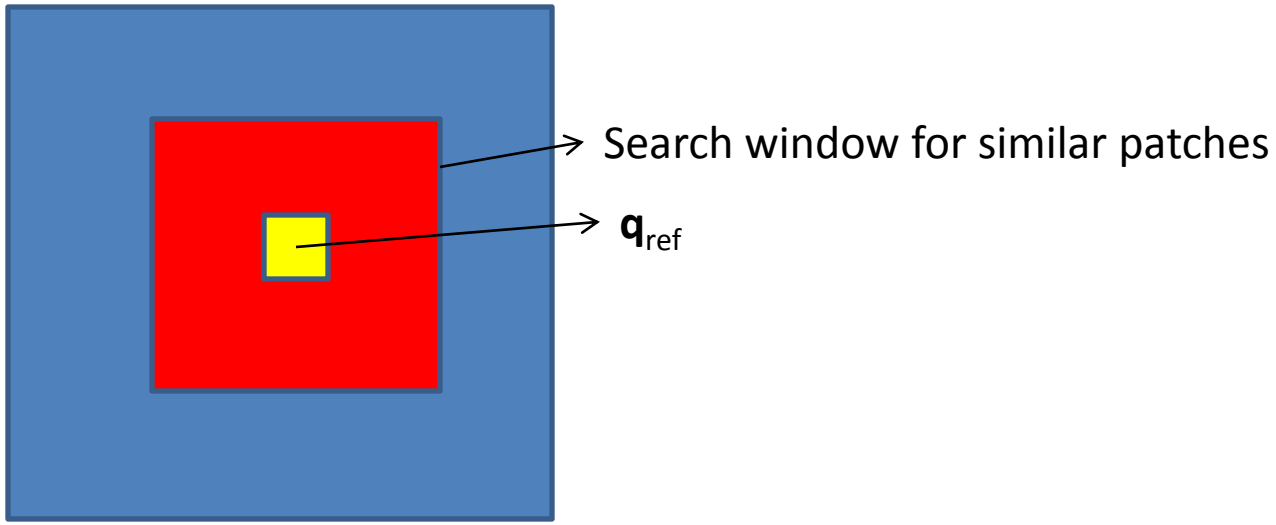
- Repeat steps 1 to 6 for all p x p patches from image J (in a sliding window fashion).
- Since we take overlapping patches, any given pixel will be covered by multiple patches (as many as p^2 different patches).
- Reconstruct the final image by averaging the output values that appear at any pixel.

Comments: Use of PCA for denoising

- Note that a **separate** eigenspace is created for each reference patch. The eigenspace is always created from patches that are similar to the reference patch.
- Such a technique is often called as **spatially varying PCA** or **non-local PCA**.

Patch similarity: Use of PCA for denoising

- To compute L nearest neighbors of $\mathbf{q}_{ref}$, restrict your search to a window around $\mathbf{q}_{ref}$.
- For every patch within the window, compute the sum of squared differences with $\mathbf{q}_{ref}$, i.e. compute: $\sum_{i=1}^{p}\sum_{j=1}^{p}(q_{ref}(i,j)-s(i,j))^2$.
- Pick *L* patches with the least distance.



Sample result

The results with a global eigenspace (consisting of all patches from the image) yield poorer results – see top row, rightmost image.

The results improve with spatially varying PCA provided the number of patches is large enough. The best results with this method generally outperform the results with a bilateral filter which is a purely local technique.

Figure 3: Left to right, top to bottom:Original image; noisy image (under zero mean, iid Gaussian noise with $\sigma = 20$); image reconstructed usign global PCA, i.e. part (a); image reconstructed using spatially vaying PCA, i.e. part (b), with $L = 25$; image reconstructed using spatially vaying PCA, i.e. part (b), with $L = 200$; result with bilateral filtering for 20 iterations with $\sigma_{spatial} = 8$ and $\sigma_{intensity} = 8$; result with bilateral filtering for 20 iterations with $\sigma_{spatial} = 8$ and $\sigma_{intensity} = 5$

Use of PCA in denoising: why Wiener-like update?

$$\beta_i(l) = k(l)\alpha_i(l), 0 \le l \le p^2 - 1$$

$$k^* = \arg\min_{k(l)} E(\beta_i(l) - k(l)\alpha_i(l))^2$$

$$= \arg\min_{k(l)} E((\beta_i(l))^2 + k(l)^2\alpha_i^2(l) - 2k(l)\alpha_i(l)\beta_i(l))$$

Eigen-coefficients of the "true patch". We are looking for a linear update which motivates this equation.

$$Consider: \mathbf{q}_i^{noisy} = \mathbf{q}_i^{true} + \mathbf{n}_i \rightarrow V^T\mathbf{q}_i^{noisy} = V^T(\mathbf{q}_i^{true} + \mathbf{n}_i)$$

$$\therefore \alpha_i(l) = \beta_i(l) + \gamma_i(l)$$

$\mathbf{n}_i$ represents a vector of pure noise values which degrades the true patch to give the noisy patch. Its projection onto the eigenspace gives vector $\boldsymbol{\Upsilon}_i$.

We will drop the index l and subscript i for better readability:

$$\therefore k^* = \arg\min_k E(\beta^2 + k^2\alpha^2 - 2k\beta\alpha)$$

$$= \arg\min_k E(\beta^2 + k^2(\beta+\gamma)^2 - 2k\beta(\beta+\gamma))$$

$$= \arg\min_k E(\beta^2) + k^2E(\beta^2 + \gamma^2) - 2kE(\beta^2), \text{ as } E(\beta\gamma) = E(\beta)E(\gamma) = 0$$

since $E(\gamma) = E(V^Tn) = V^TE(n) = 0$

As the noise is zero mean

As the image and the noise are independent

Use of PCA in denoising: why Wiener-like update?

Setting derivative w.r.t. k to 0, we get

$$kE(\beta^2+\gamma^2)=E(\beta^2)\rightarrow k=\frac{E(\beta^2)}{E(\beta^2+\gamma^2)}=\frac{E(\beta^2)}{E(\beta^2)+\sigma^2}$$

How should we estimate $E(\beta^2)$?

Recall:

$$\alpha_i(l)=\beta_i(l)+\gamma_i(l)\approx\beta(l)+\gamma_i(l)$$

$$\therefore E(\alpha_i^2(l))=E(\beta^2(l))+E(\gamma_i^2(l))$$

$$\therefore E(\beta^2(l))=E(\alpha_i^2(l))-\sigma^2$$

$$=\frac{1}{L}\sum_{i=1}^{L}\alpha_i^2(l)-\sigma^2$$

This may be a negative value, so we set it to be

$$E(\beta^2(l))=\max(0,\frac{1}{L}\sum_{i=1}^{L}\alpha_i^2(l)-\sigma^2)$$

Since we are dealing with *L* similar patches, we can assume (approximately) that the *l*-th eigen-coefficient of each of those *L* patches are very similar.