Lecture 03: Image Processing II

[AIX7021] Computer Vision

Seongsik Park (s.park@dgu.edu)

AI Department, Dongguk University

Overview

- Image Processing I
  - Point process
  - Linear filtering and convolution
  - Smoothing: Gaussian, median, bilateral
  - Fourier transform
- Image Processing II
  - Fourier transform (cont'd), aliasing and scaling
  - Edge detection, corner detection
  - ~~Binary image~~

Fourier Transform, aliasing and scaling

Fourier Transform

We need a new technique to deal with two related problems so far left open:

- Although it is clear that a discrete image version cannot represent the full information in a signal, we have not yet indicated what is lost.
- It is clear that we cannot shrink an image simply by taking every k-th pixel–this could turn a checker board image all white or all black–and we would like to know how to shrink an image safely.

Fourier Transform

The change of basis is effected by a *Fourier transform*. We define the Fourier transform of a signal $g(x, y)$ to be

$$\mathcal{F}(g(x,y))(u,v) = \int\int_{-\infty}^{\infty} g(x,y)e^{-i2\pi(ux+vy)}dxdy \tag{1}$$

$$\text{where } e^{-i\theta} = \cos\theta + i\sin\theta$$

Fix u and v, and let us consider the meaning of the value of the transform at that point. The exponential can be written

$$e^{-i2\pi(ux+vy)} = \cos\Big(2\pi(ux+vy)\Big) + i\sin\Big(2\pi(ux+vy)\Big) \tag{2}$$

These terms are sinusoids on the x, y plane, whose orientation and frequency are given by u, v.

Fourier Transform: Fourier basis

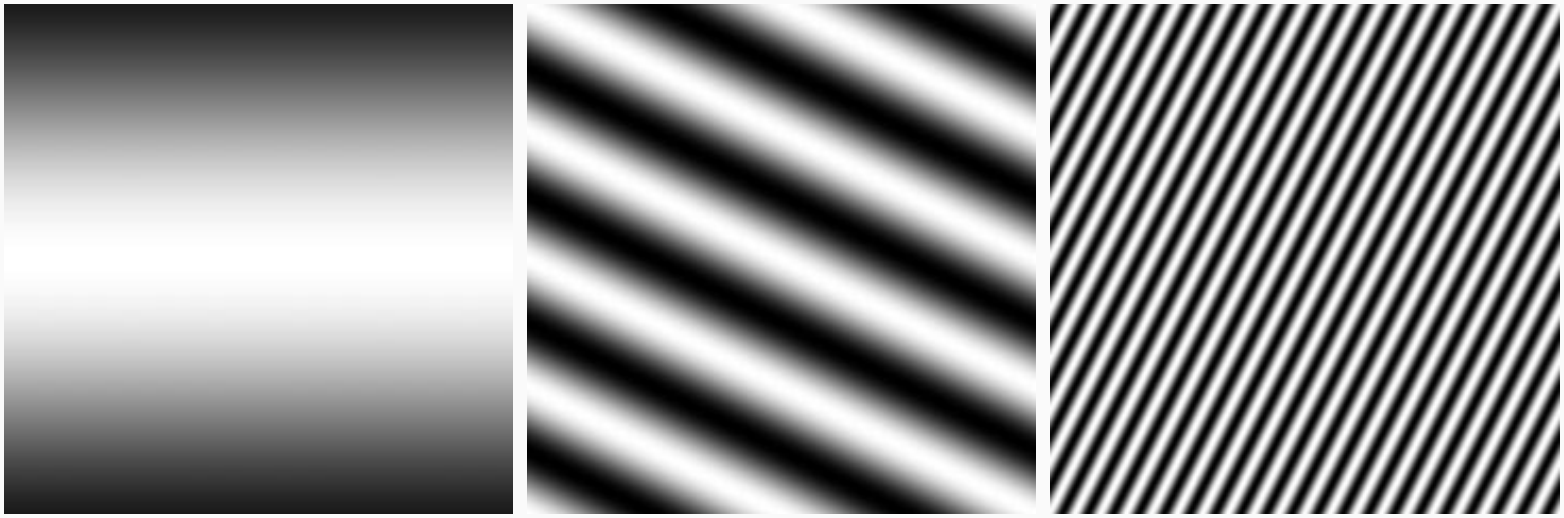

FIGURE 4.5: The real component of Fourier basis elements shown as intensity images. The brightest point has value one, and the darkest point has value zero. The domain is $[-1, 1] \times [-1, 1]$, with the origin at the center of the image. On the **left**, $(u, v) = (0, 0.4)$; in the **center**, $(u, v) = (1, 2)$; and on the **right** $(u, v) = (10, -5)$. These are sinusoids of various frequencies and orientations described in the text.

Fourier Transform: magnitude and phase spectrum

Phase spectrum is more important in image process?

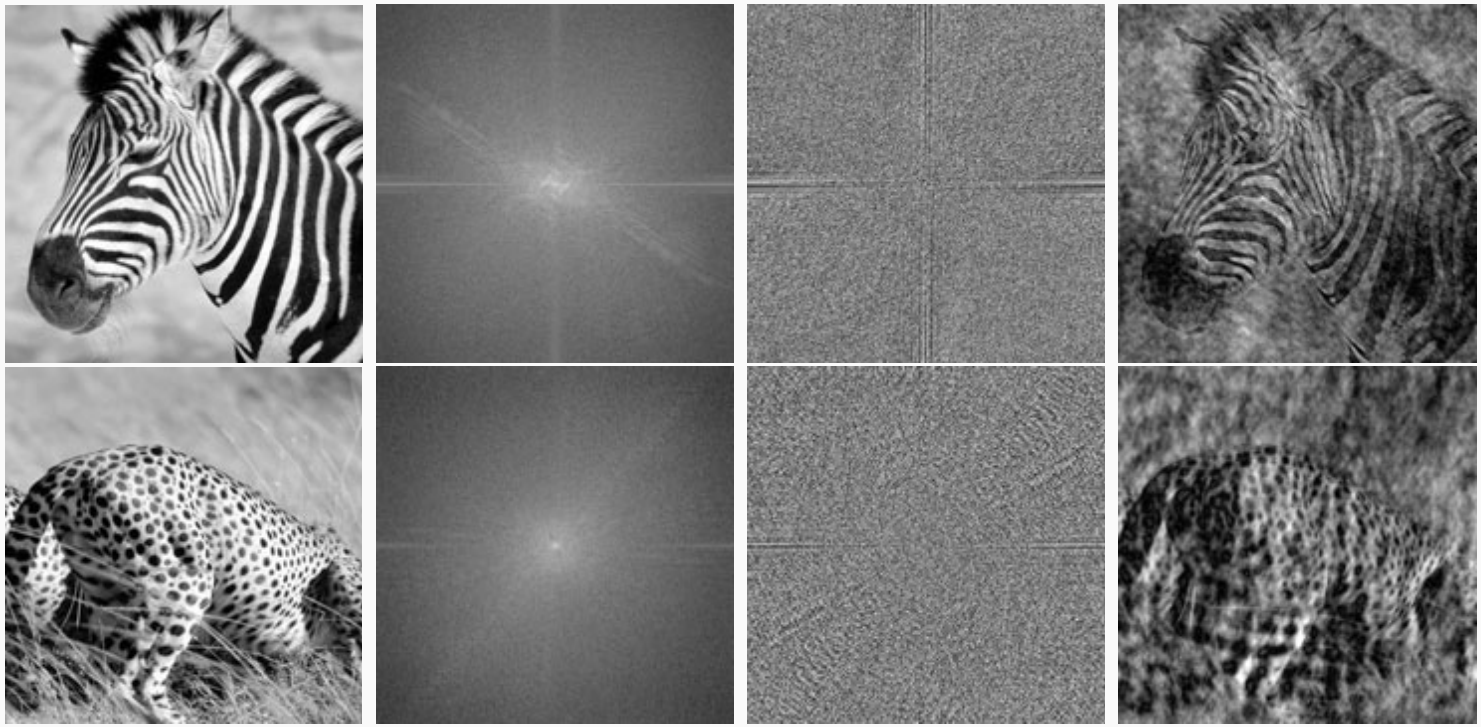

FIGURE 4.6: The second image in each row shows the log of the magnitude spectrum for the first image in the row; the third image shows the phase spectrum scaled so that $-\pi$ is dark and π is light. The final images are obtained by swapping the magnitude spectra. Although this swap leads to substantial image noise, it doesn't substantially affect the interpretation of the image, suggesting that the phase spectrum is more important for perception than the magnitude spectrum.

Fourier Transform

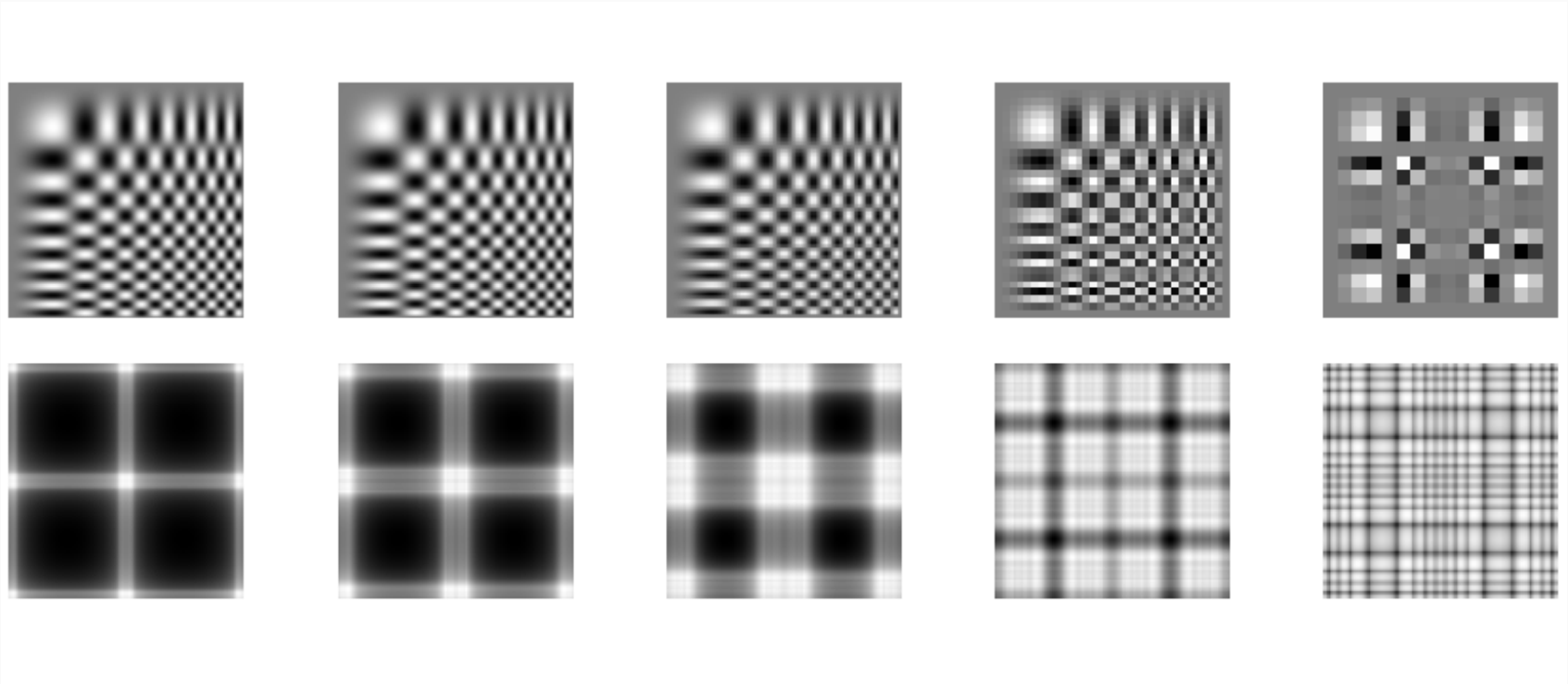

Figure 1: Sinusoidal images with increasing frequency and their magnitude spectrum

Fourier Transform: forward transform revisited

Fourier transform

$$\mathcal{F}(g(x,y))(u,v) = \int\int_{-\infty}^{\infty} g(x,y)e^{-i2\pi(ux+vy)}dxdy \tag{3}$$

$$\text{where } e^{-i\theta} = \cos\theta + i\sin\theta$$

Real and imaginary term

$$\mathcal{F}(g(x,y))(u,v) = \int\int_{-\infty}^{\infty} g(x,y)\cos 2\pi(ux+vy)dxdy \tag{4}$$

$$+ i\int\int_{-\infty}^{\infty} g(x,y)\sin 2\pi(ux+vy)dxdy \tag{5}$$

$$= \Re(\mathcal{F}(g)) + i * \Im(\mathcal{F}(g)) \tag{6}$$

$$= \mathcal{F}_R(g) + i * \mathcal{F}_I(g) \tag{7}$$

Magnitude and phase spectrum

$$\text{magnitude} = |\mathcal{F}(g)| = \sqrt{|\mathcal{F}_R(g)|^2 + |\mathcal{F}_I(g)|^2} \tag{8}$$

$$\text{phase} = \measuredangle\mathcal{F}(g) = \text{atan2}(\mathcal{F}_I(g), \mathcal{F}_R(g)) \tag{9}$$

Fourier Transform: inverse transform revisited

Inverse Fourier transform

$$g(x,y) = \int\int_{-\infty}^{\infty} \mathcal{F}(g)(u,v) \exp\Big(i2\pi(ux+vy)\Big) du dv \tag{10}$$

$$= \int\int_{-\infty}^{\infty} |\mathcal{F}(g)(u,v)| \exp\Big(i2\pi(ux+vy) + i\measuredangle\mathcal{F}(g)(u,v)\Big) du dv \tag{11}$$

Aliasing: sampling of checkerboard

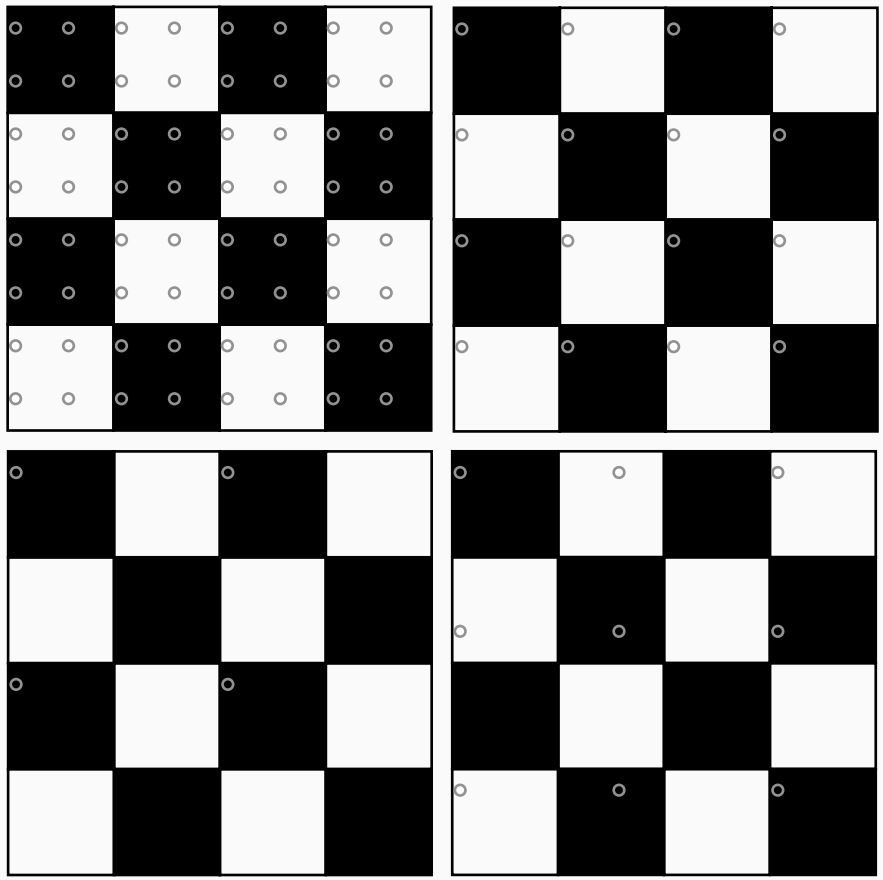

FIGURE 4.7: The two checkerboards on the **top** illustrate a sampling procedure that appears to be successful (whether it is or not depends on some details that we will deal with later). The gray circles represent the samples; if there are sufficient samples, then the samples represent the detail in the underlying function. The sampling procedures shown on the **bottom** are unequivocally unsuccessful; the samples suggest that there are fewer checks than there are. This illustrates two important phenomena: first, successful sampling schemes sample data often enough; and second, unsuccessful sampling schemes cause high-frequency information to appear as lower-frequency information.

Nyquist sampling theorem

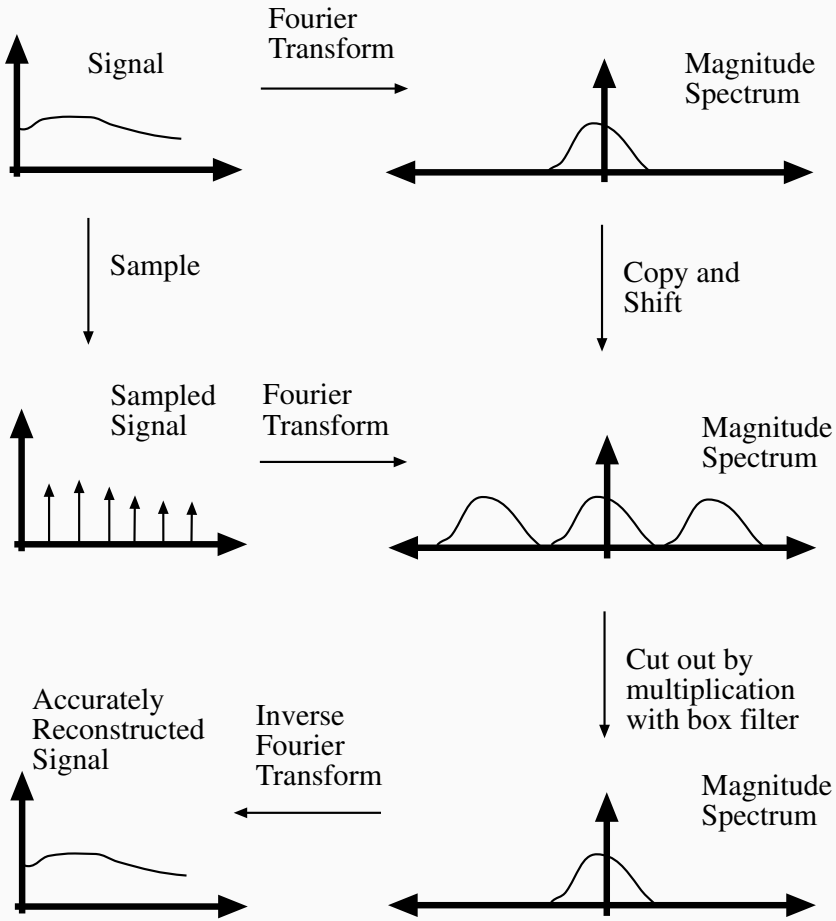

FIGURE 4.10: The Fourier transform of the sampled signal consists of a sum of copies of the Fourier transform of the original signal, shifted with respect to each other by the sampling frequency. Two possibilities occur. If the shifted copies do not intersect with each other (as in this case), the original signal can be reconstructed from the sampled signal (we just cut out one copy of the Fourier transform and inverse transform it). If they do intersect (as in Figure 4.11), the intersection region is added, and so we cannot obtain a separate copy of the Fourier transform, and the signal has aliased.

Aliasing and Nyquist sampling theorem

Sampling involves a loss of information. A signal sampled too slowly is misrepresented by the samples; high spatial frequency components of the original signal appears as low spatial frequency components in the sampled signal–an effect known as *aliasing*.

Nyquist sampling theorem

in order to adequately reproduce a signal, it should be periodically sampled at a rate that is 2 times the highest frequency you wish to record

Proper image reconstruction should follow:

1. sampling
2. Fourier transform
3. band-pass filter (box filter)
4. inverse Fourier transform

Anti-aliasing and Gaussian pyramid

Fourier transform of convolution

convolution: $f(x) * h(x)$

Fourier: $F(\omega)H(\omega)$

Hence,

Gaussian blurring = low-pass filter

= sub-sampling itself

So, Gaussian pyramid = blur + subsample

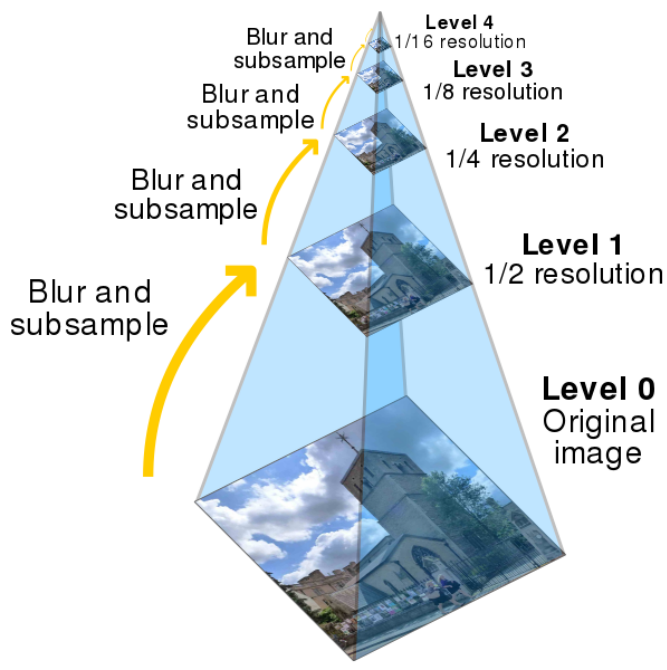

Figure 2: Gaussian pyramid

Anti-aliasing and Gaussian pyramid

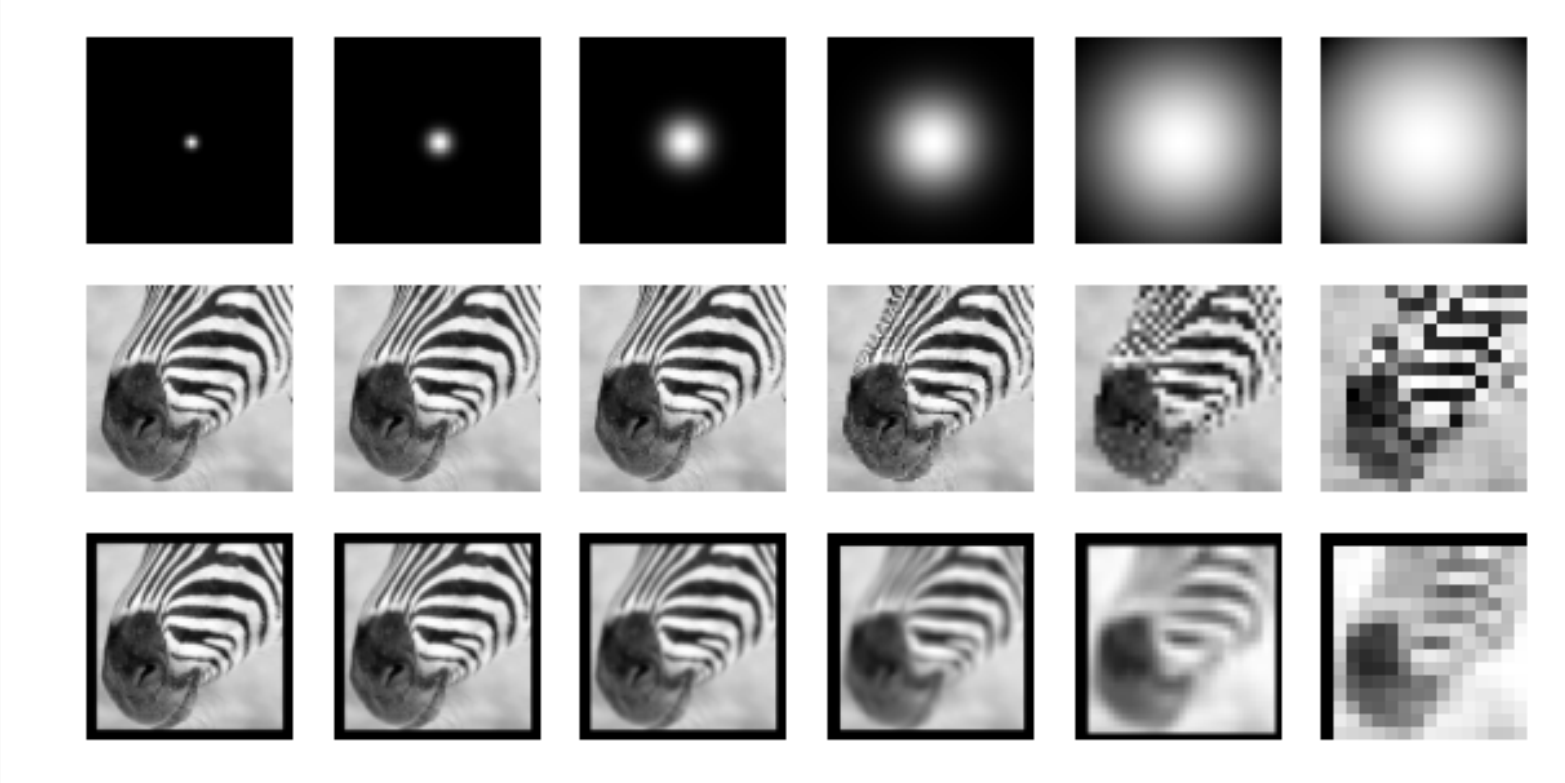

Figure 3: Gaussian pyramid with *sigma* = 1, , 4, 8, 16, 32, 64. Gaussian kernel (top), simple subsampling (middle) and Gaussian pyramid (bottom)

Anti-aliasing and Gaussian pyramid

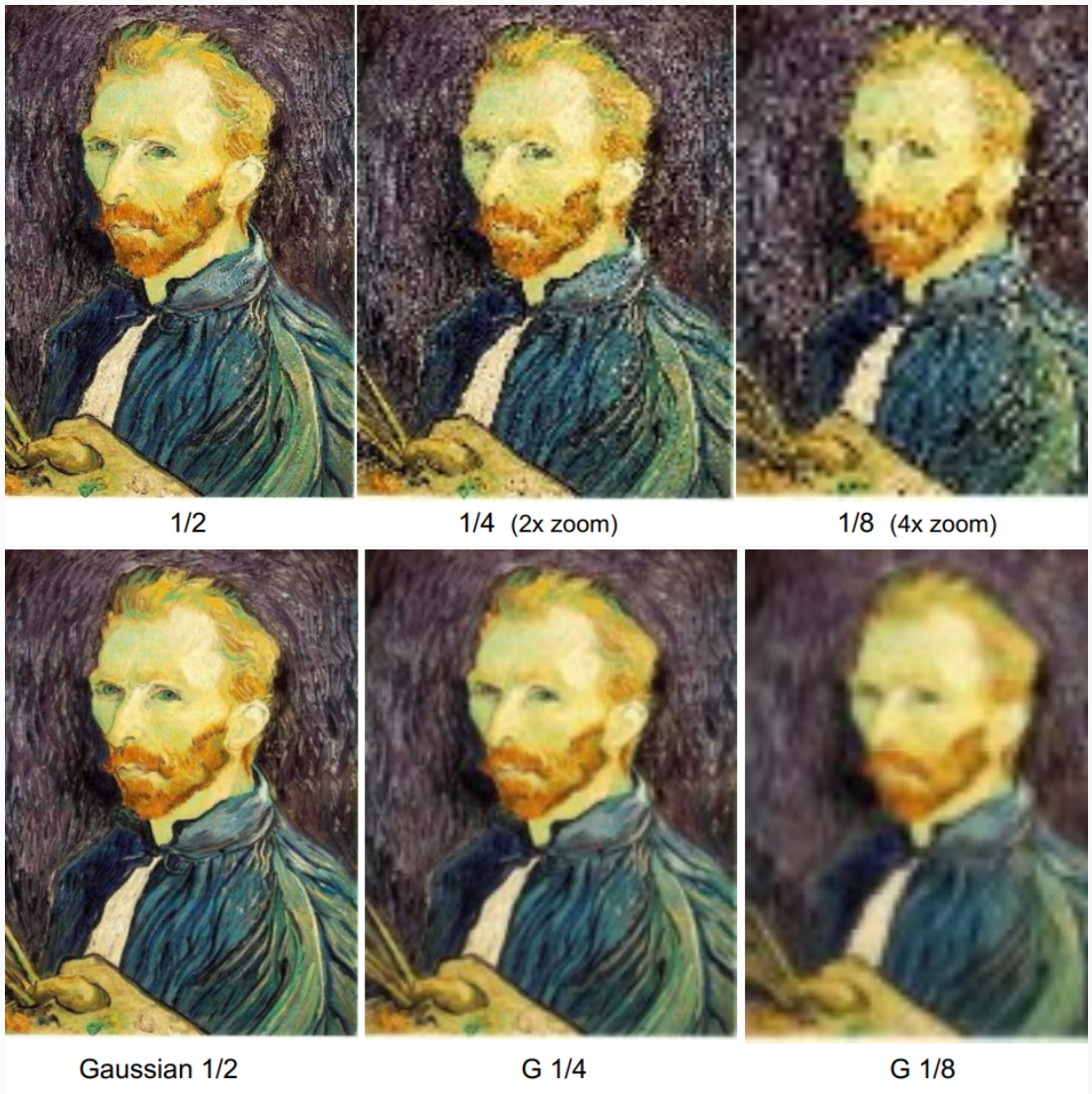

Figure 4: Subsampling alone causing aliasing (top) and Gaussian pyramid for anti-aliasing (bottom)

Edge and Corner

Derivative and finite differences

Image derivatives can be approximated using a convolution process. We might estimate a partial derivative as a symmetric *finite difference*

$$\text{Horizontal: } \frac{\partial h}{\partial x} \approx h_{i+1,j} - h_{i-1,j} \quad \mathcal{H} = \begin{Bmatrix} 0 & 0 & 0 \\ 1 & 0 & -1 \\ 0 & 0 & 0 \end{Bmatrix} \tag{12}$$

$$\text{Vertical: } \frac{\partial h}{\partial x} \approx h_{i,j+1} - h_{i,j-1} \quad \mathcal{H} = \begin{Bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & -1 & 0 \end{Bmatrix} \tag{13}$$

Derivative and finite differences

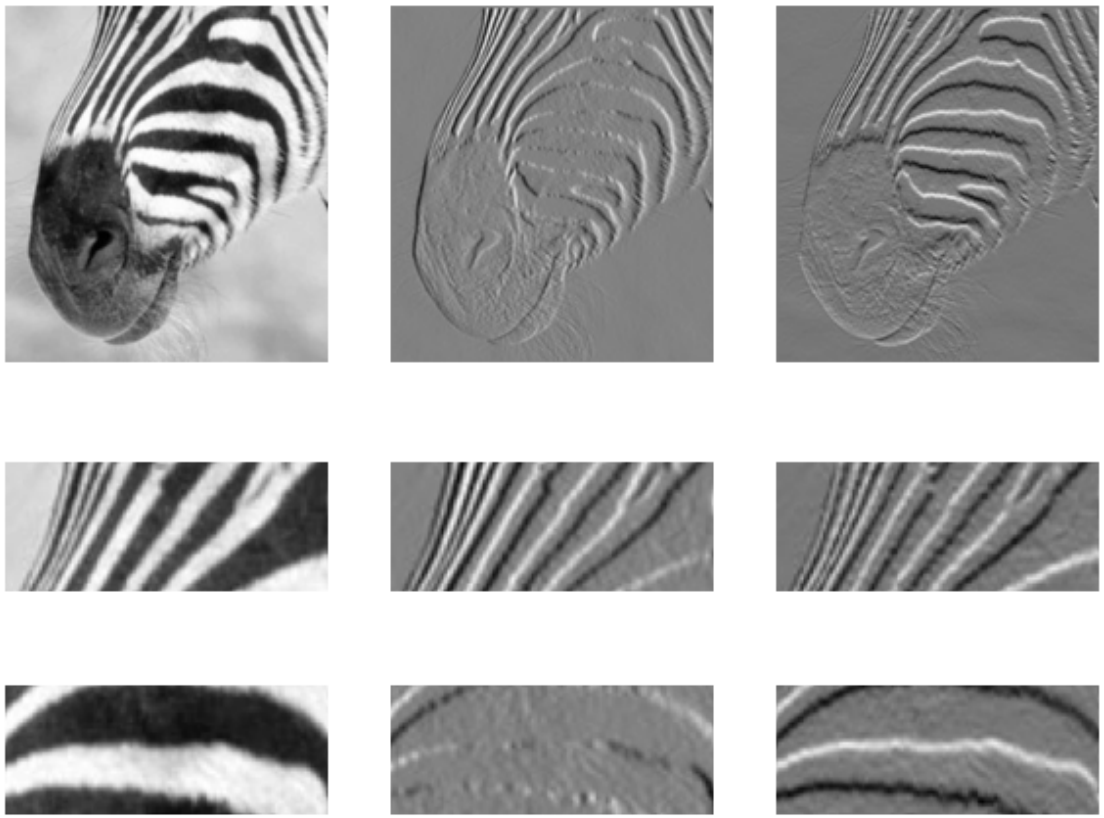

Figure 5: Horizontal and vertical difference using convolution filter

Image gradient and Canny edge detection

The gradient of an image is a vector of its partials derivatives

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} \tag{14}$$

$$\text{magnitude} = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2} \tag{15}$$

$$\text{direction} = \text{atan2}\left(\frac{\partial f}{\partial y}, \frac{\partial f}{\partial x}\right) \tag{16}$$

Canny edge detection

1. Gaussian blur
2. Image gradient magnitude
3. Threshold

Image gradient and Canny edge detection

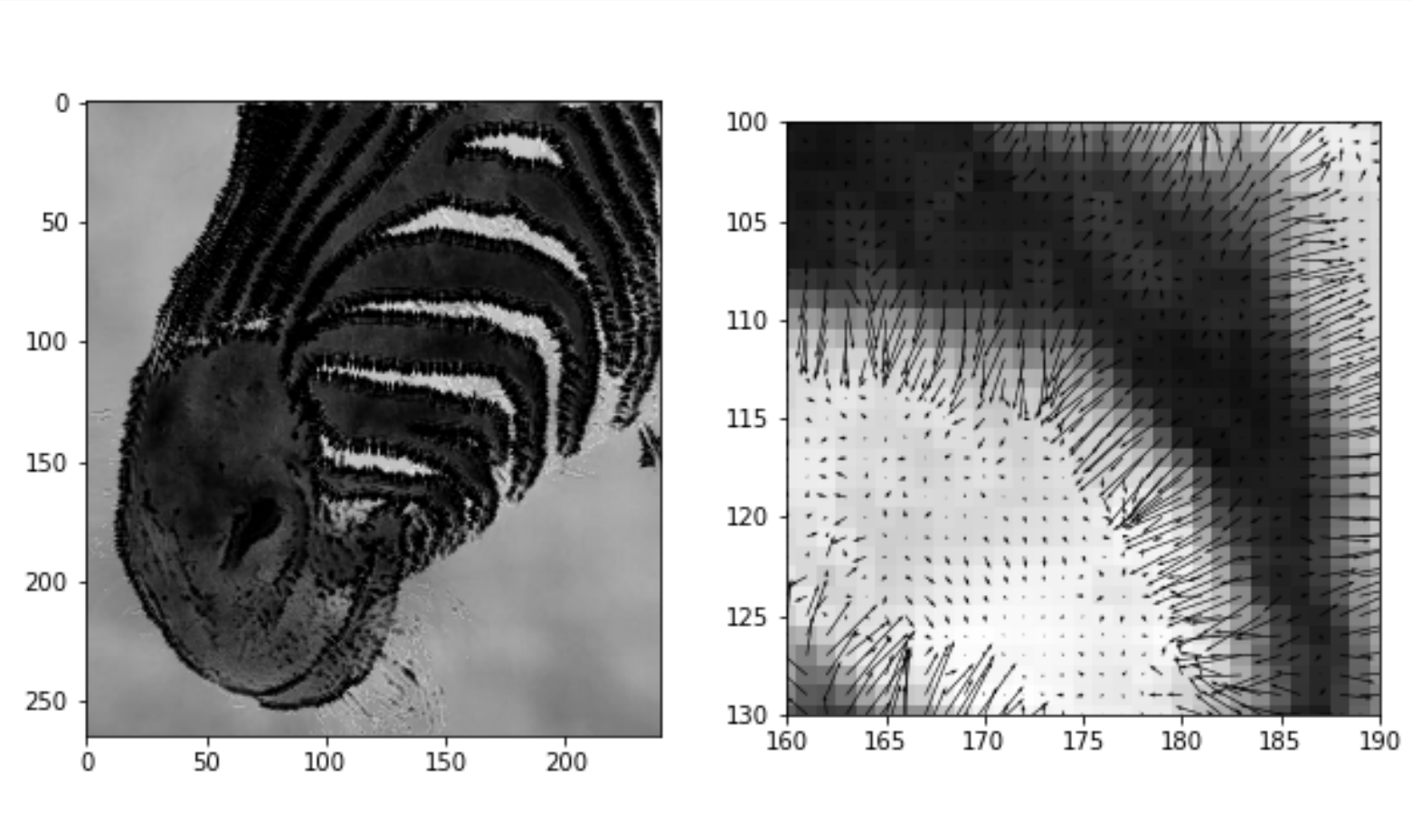

Figure 6: Quiver plot of image gradient

Image gradient and Canny edge detection

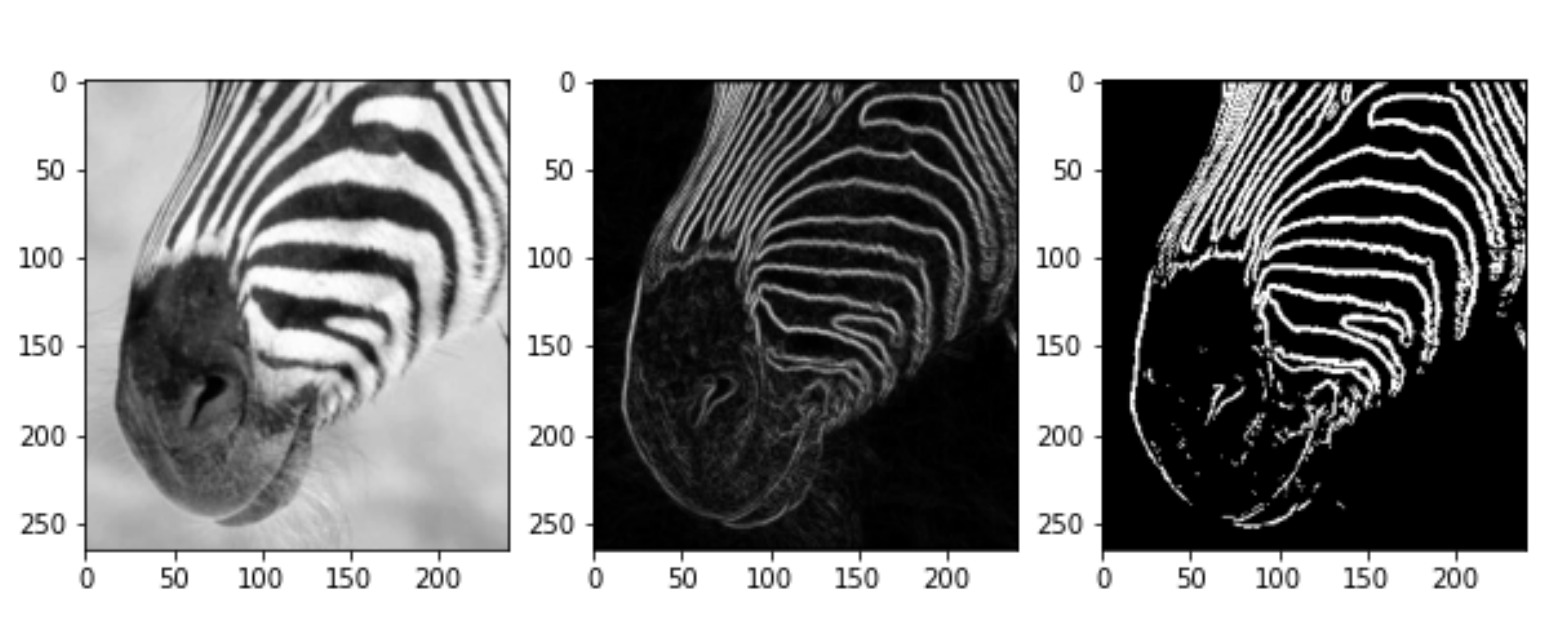

Figure 7: Canny edge detection. Original image, gradient magnitude and thresholding (canny edge detection) image.

Laplacian of Gaussian filter (LoG filter)

More sophisticated kernels can be created by first smoothing the image with a Gaussian filter,

$$G(x, y; \sigma) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right), \tag{17}$$

and then taking the first and second derivatives. Such filters are known collectively as *band-pass filters*, since they filter out both low and high frequencies.

The undirected second derivative of a two-dimensional image,

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}, \tag{18}$$

is known as the *Laplacian* operator. Blurring an image with a Gaussian and then taking its Laplacian is equivalent to convolving directly with the *Laplacian of Gaussian* (LoG) filter,

$$\nabla^2 G(x, y; \sigma) = \left(\frac{x^2 + y^2}{\sigma^4} - \frac{2}{\sigma^2}\right) G(x, y; \sigma). \tag{19}$$

Laplacian of Gaussian filter (LoG filter)

LoG filter

$$\nabla^2 G(x, y; \sigma) = \left(\frac{x^2 + y^2}{\sigma^4} - \frac{2}{\sigma^2} \right) G(x, y; \sigma). \tag{20}$$

Or, simply

$$\mathcal{H} = \begin{Bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{Bmatrix} \tag{21}$$

Laplacian of Gaussian filter (LoG filter)

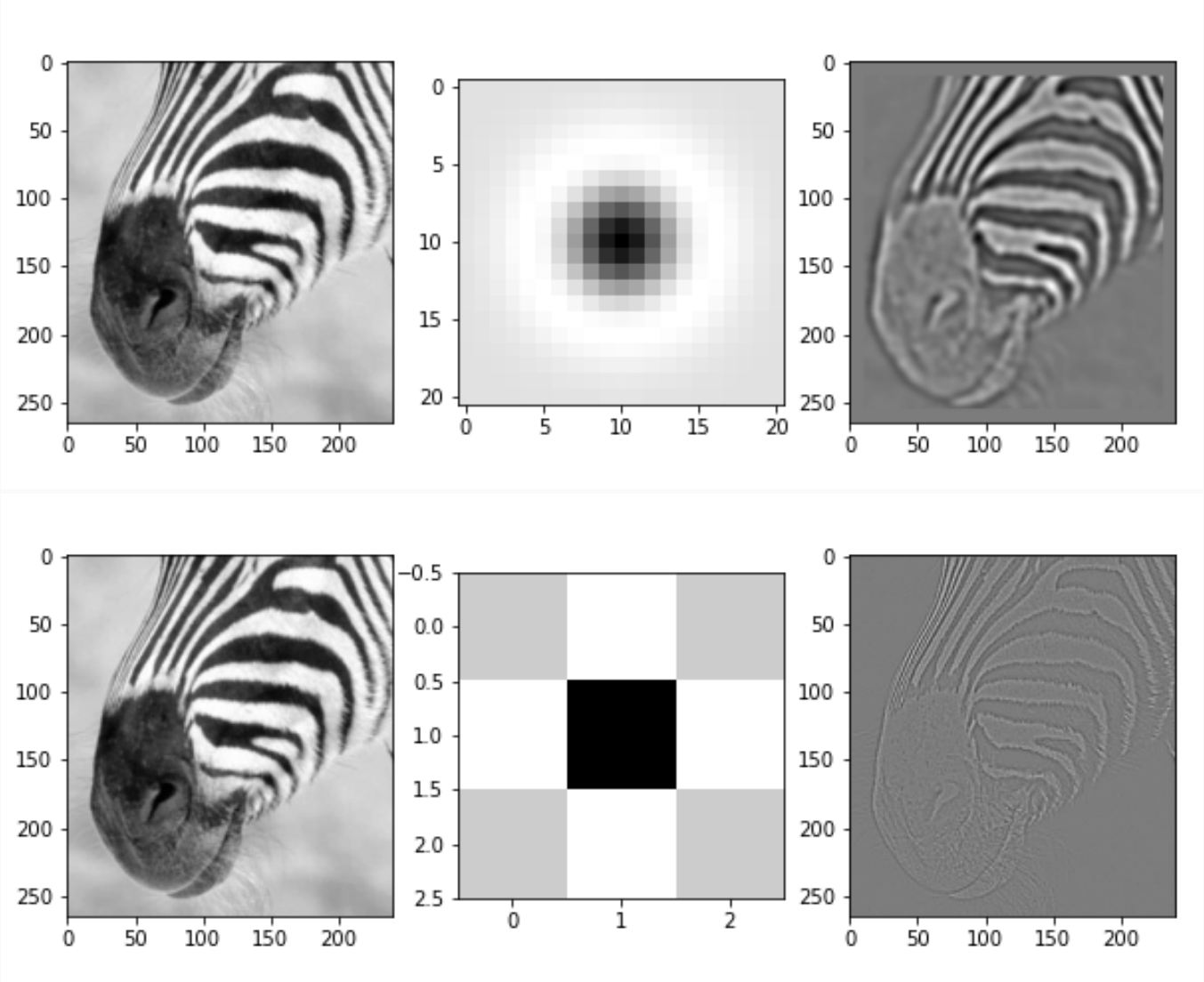

Figure 8: Laplacian of Gaussian filter

Sobel operator

The *Sobel operator* is a simple approximation to a *directional* or *oriented* filter, which can be obtained by smoothing with a Gaussian (or some other filter) and then taking a *directional derivative* $\nabla_{\hat{\mathbf{u}}} = \frac{\partial}{\partial \hat{\mathbf{u}}}$, which is obtained by taking the dot product between the gradient field ∇ and a unit direction $\hat{\mathbf{u}} = (\cos\theta, \sin\theta)$,

$$\hat{\mathbf{u}} \cdot \nabla(G * f) = \nabla_{\hat{\mathbf{u}}}(G * f) = (\nabla_{\hat{\mathbf{u}}} G) * f. \tag{22}$$

The smoothed directional derivative filter,

$$G_{\hat{\mathbf{u}}} = u_1 G_x + u_2 G_y = u_1 \frac{\partial G}{\partial x} + u_2 \frac{\partial G}{\partial y} \tag{23}$$

where $\hat{\mathbf{u}} = (u_1, u_2)$, is an example of a *steerable* filter, since the value of an image convolved with $G_{\hat{\mathbf{u}}}$ can be computed by first convolving with the pair of filter (G_x, G_y) and then *steering* the filter by multiplying this gradient field with a unit vector $\hat{\mathbf{u}}$.

Sobel operator

Sobel operator for a given $\hat{\mathbf{u}} = (u_1, u_2)$

$$G_{\hat{\mathbf{u}}} = u_1 G_x + u_2 G_y = u_1 \frac{\partial G}{\partial x} + u_2 \frac{\partial G}{\partial y} \tag{24}$$

with

$$G(x, y; \sigma) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right) \tag{25}$$

$$\frac{\partial G}{\partial x} = -\frac{x}{\sigma^2} G(x, y; \sigma) \qquad \frac{\partial G}{\partial y} = -\frac{y}{\sigma^2} G(x, y; \sigma) \tag{26}$$

Sobel operator

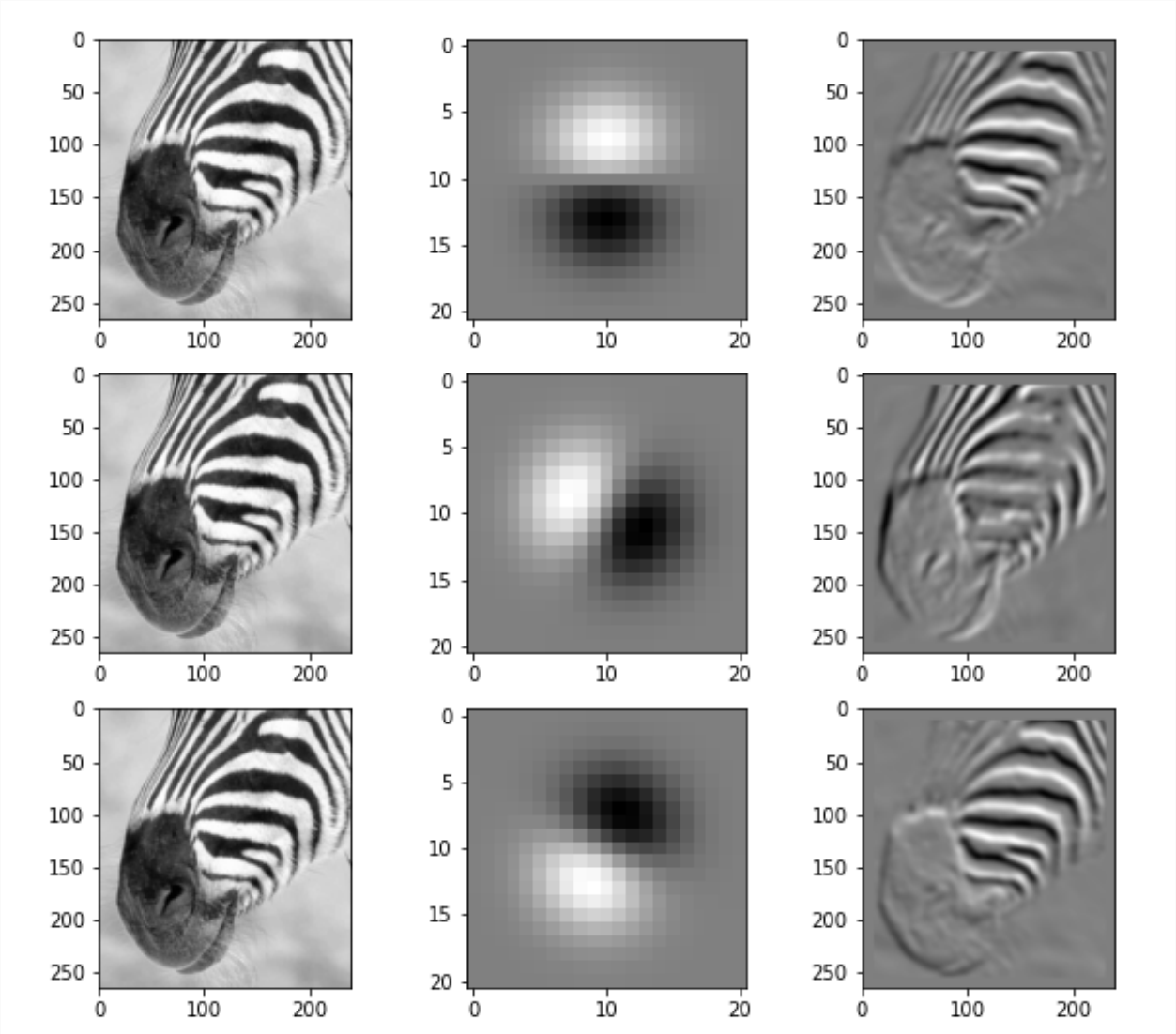

Figure 9: Sobel operator with u = (1, 0) (top), u = (0.447, 0.894) (middle), u = (-0.894, 0.447) (bottom)

Appendix

Reference and further reading

- “Chap 3” and “Chap 7” of R. Szeliski, Computer Vision: Algorithms and Applications
- “Chap 4” and “Chap 5” of Forsyth and Ponce, Computer Vision: A Modern Approach

HW #3

Due: 9월 21일 월요일, 23시 59분까지

- python 파일이나 jupyter notebook 파일을 업로드 해주세요. (압축파일도 가능)
- 주어진 이미지가 같은 경로 상에 있을 때, 정상적으로 실행되어야 합니다.

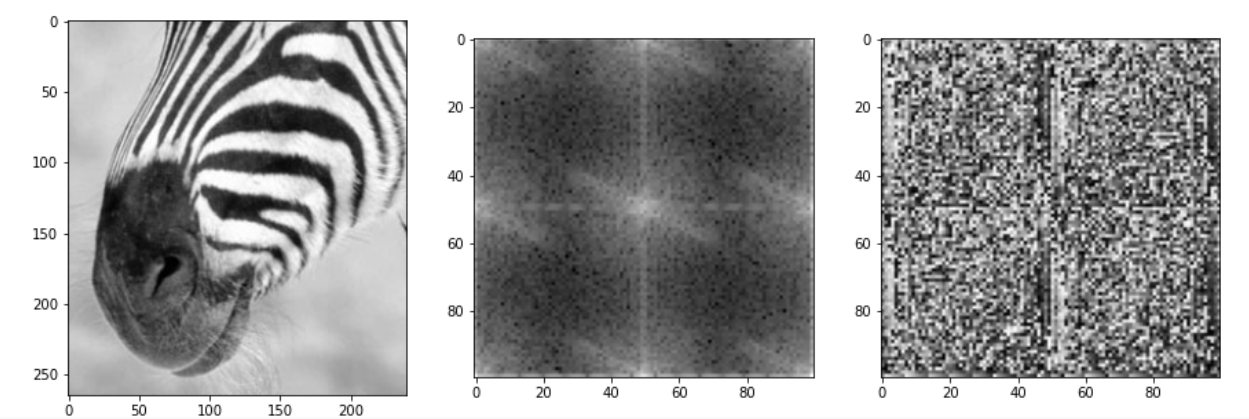

Figure 10: Original image, log10(magnitude) and phase spectrum

1. 주어진 이미지 zebra.png에 Fourier transform을 취하여 그 magnitude와 phase spectrum을 구하세요. (15점)

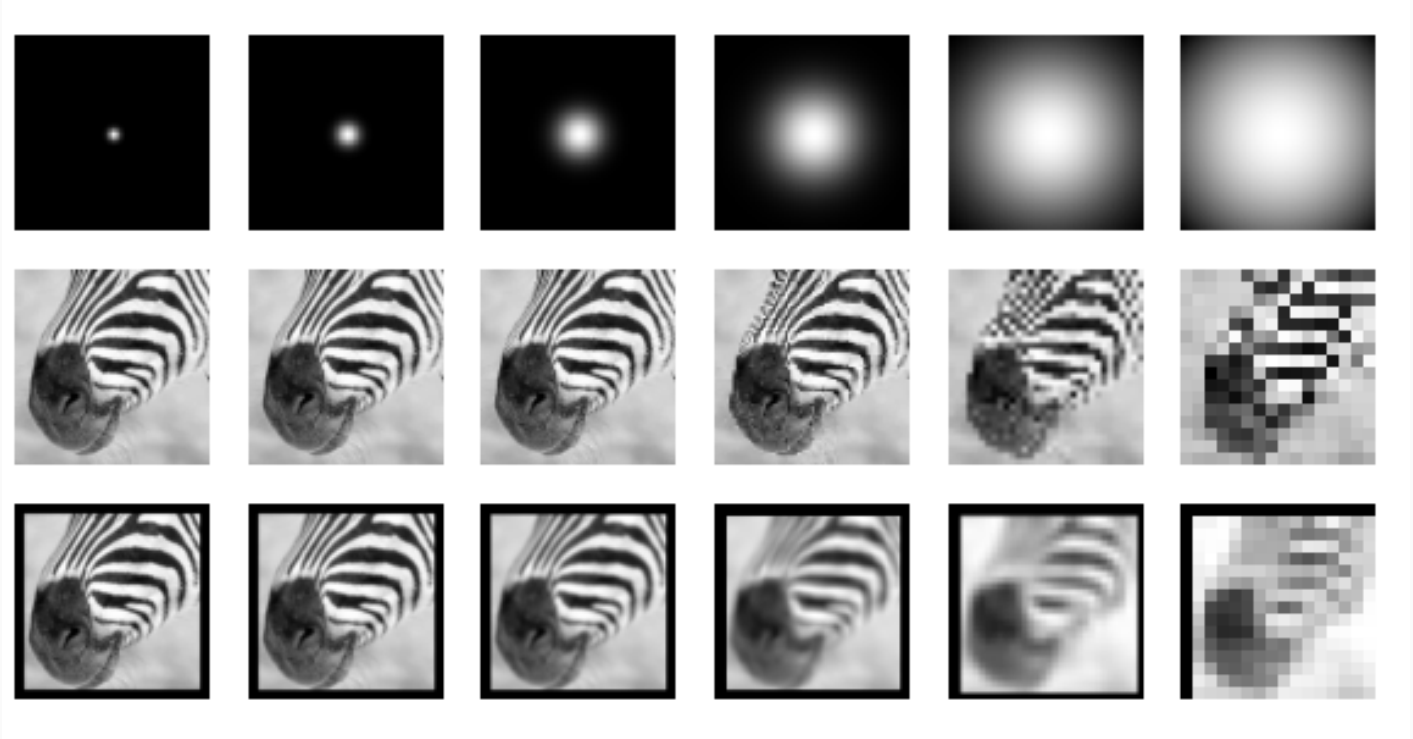

Figure 11: Gaussian pyramid

2. 주어진 512 × 512 픽셀의 이미지 zebra.png의 simple subsampling과 Gaussian pyramid를 256, 128, 64, 32, 16 사이즈로 구하세요. (10점)