DIGITAL CULTURE
ITMO UNIVERSITY

Frequency Filtering

High School of Digital Culture
ITMO University
dc@itmo.ru

Contents

1 Basic Image Processing Techniques

We've discussed multiple image processing techniques in spatial domain, but there are many more techniques which are best performed in the frequency domain. In this domain, pixel location is represented by its x- and y-frequencies and its value is represented by an amplitude. And frequency in an image tells about the rate of change of pixel values.

1D image To better understand what's it all about, let's take a look it 1D picture - one pixel row from original 2D image. Here you can see that first there are gray pixels which correspond to intensity values about 150. then it is followed by a white region with high intensity values, close to maximum - 255. Then there is a black region with very low intensity values, then white, black again and white again.

Now let's look at intensity values in a row of pixel for another image, Lena. You can see that here intensity levels change much more frequently. How to describe this property, which tells how frequently intensity changes?

1.1 Frequency representation – the main idea

The main idea behind frequency representation belongs to the French mathematician Jean Baptiste Joseph Fourier. He stated that any function that periodically repeats itself can be expressed as the sum of sines and cosines of different frequencies, each multiplied by a different coefficient. We now call this sum a Fourier series. It doesn't matter how complicated the function is; as long as it is periodic and meets some mild mathematical conditions, it can be represented by such a sum.

1.2 Frequency representation – example

On this slide we see another example of a function decomposed into sines. The Fourier series represents a periodic signal as the sum of harmonics. Essentially, it maps a given function of time (or space) into a frequency spectrum. The frequency spectrum is a simple way of showing the amplitude at each of the frequencies constituting a given function. It tells us which frequencies are present in the original signal, and now strong the corresponding amplitudes are. The frequency spectrum retains the frequency information but discards the phase information. So, for example, the frequency spectrum of a sine wave would be the same as that of a cosine wave of the same frequency, even though the complete Fourier transforms of sine and cosine waves are different in phase.

On this slide we can see the frequency spectrum of a given function. In this example we can see that our original function is composed of only two frequencies if 1 and 3 with amplitudes of 1 and 0.5.

Fourier transform But what if my function is not periodic? Even functions that are not periodic, but whose area under the curve is finite, can be expressed as the integral of sines and cosines multiplied by some weights. This process of decomposition of a function into its constituent frequencies is called Fourier transform. The difference between Fourier series and Fourier transform is that Fourier series is applied on periodic signals and Fourier transform is applied for non periodic signals. In our case, images are functions of finite duration, so Fourier transform is a tool to use with images. The Fourier transform of a function of time or space is a complex-valued function of frequency, whose magnitude (absolute value) represents the amount of that frequency present in the original function, and whose argument is the phase offset of the basic sinusoid in that frequency. There is also an inverse Fourier transform that mathematically synthesizes the original function from its frequency domain representation, without any loss of information.

1D Fourier Transform and its Inverse The Fourier transform of a continuous function $f(x)$ of a continuous variable x is defined by the equation on this slide, where u is also a continuous variable. Because x is integrated out, the resulting function F is a function of u.

$$F(u) = \int_{-\infty}^{\infty} f(x) e^{-\imath 2\pi u x} \mathrm{d}x \tag{1}$$

Using Euler's formula, we can express $F(u)$ differently.

$$F(u) = \int_{-\infty}^{\infty} f(x) \left[\cos(2\pi u x) - \imath \sin(2\pi u x)\right] \mathrm{d}x \tag{2}$$

If $f(x)$ is real, we see that it's transform in general is complex. Note that the Fourier transform is an expansion of $f(x)$ multiplied by sinusoidal terms whose frequencies are determined by the values of u. Because the only variable left after integration is frequency, we say that the domain of Fourier transform is the frequency domain.

Conversely, given $F()u)$, we can obtain $f(x)$ back using the inverse Fourier transform.

$$f(x) = \int_{-\infty}^{\infty} F(u) e^{\imath 2\pi u x} \mathrm{d}u \tag{3}$$

Discrete Fourier Transform So far we've looked at Fourier transform on continuous functions. But digital images aren't continuous - they can be represented as 2D discrete functions. Original continuous analog signal is converted into a

sequence of discrete values by using sampling and quantization. Fourier transform of a discrete function can be derived from Fourier transform of a continuous function using the concept of sampling. Discrete Fourier transform (DFT) converts a finite sequence of equally-spaced samples of a function into a same-length sequence of equally-spaced samples of the discrete Fourier transform, and can be expressed as follows.

$$F(u) = \frac{1}{M} \sum_{x=0}^{M-1} f(x) e^{-\imath 2\pi u x/M}, \ u = 0, 1, 2, \ldots, M-1 \tag{4}$$

Given a set f_x consisting of M samples of $f(x)$, this equation yields a sample set F_u of M complex discrete values corresponding to the discrete Fourier transform of the input sample set. Each term of the Fourier transform (that is, the value if $F(u)$ for each value of u) is composed of the sum of all values of the function $f(x)$. The values of $f(x)$ are multiplied by sines and cosines of various frequencies. Each of the M terms of $F(u)$ is called a frequency component of the transform.

And conversely, given a set of samples of $F(u)$, we can recover the samples set f_x by using the inverse discrete Fourier transform (IDFT):

$$f(x) = \frac{1}{M} \sum_{u=0}^{M-1} F(u) e^{\imath 2\pi u x/M}, \ x = 0, 1, 2, \ldots, M-1 \tag{5}$$

It can be shown that both the forward and inverse discrete transforms are infinitely periodic, with period M.

The $1/M$ multiplier in fromt of the Fourier transform sometimes is placed in front of the inverse instead. Other times (not as often though)both equations are multiplied by $1/\sqrt{1/M}$. the location of the multiplier doesn't matter. If two multipliers are used, the only requirement is that their product be equal to $1/M$.

Fourier spectrum In general, the Fourier transform contains complex terms, and it is customary for display purposes to work with the magnitude of the transform (a real quantity), which is called the Fourier spectrum or the frequency spectrum, or sometimes the magnitude of the Fourier transform:

$$|F(u)| = \left(R^2(u) + I^2(u)\right)^{1/2} \tag{6}$$

Here, $R(u)$ and $I(u)$ are the real and imaginary parts of $F(u)$, respectively.

In image processing we will be mostly dealing with properties of the spectrum. Another quantity that is used sometimes is the power spectrum, defined as the square of the Fourier spectrum:

$$P(u) = |F(u)|^2 = (R^2(u) + I^2(u) \tag{7}$$

And another quantity which is sometimes used is phase spectrum or phase angle:

$$\phi(u) = \arctan\left(\frac{I(u)}{R(u)}\right) \tag{8}$$

Fourier transform of a simple function before proceeding, let's consider a simple one-dimensional example of Fourier transform. The Fourier transform of the function of the slide, can be derived as follows:

$$\begin{aligned} F(u) = \int_{-\infty}^{\infty} f(x) e^{-\imath 2\pi u x} \mathrm{d}x = \int_{-W/2}^{W/2} A e^{-\imath 2\pi u x} \mathrm{d}x \\ = \frac{-A}{\imath 2\pi u}\left(e^{-\imath\pi u W} - e^{\imath\pi u W}\right) = \frac{A}{\imath 2\pi u}\left(e^{\imath\pi u W} - e^{-\imath\pi u W}\right) \\ = AW\frac{\sin(\pi u W)}{\pi u W} \end{aligned} \tag{9}$$

Here to make the last step we used the trigonometric identity $sin\theta = (e^{\imath\theta} - e^{-\imath\theta})/2\imath$. In this case, the complex terms of the Fourier transform combined nicely into a real sine function.

Let's look at the plot of this function and the plot of its magnitude. All functions extend to infinity in both directions. The key properties to note are (1) that the locations of the zeros of both $F(u)$ and $|F(u)|$ are inversely proportional to the width,W, of the "box" function; (2) that the height of the lobes decreases as a function of distance from the origin; and (3) that the function extends to infinity for both positive and negative values of u. We will see later, that these properties are quite helpful in interpreting the spectra of two dimensional Fourier transforms of images.

Fourier transform for images As any other discrete 2-D function, an image can be represented as a weighted sum of simple sinusoids. Irrespectively of how irregular may be an image, it can be decomposed into a set of sinusoidal components having each a well-defined frequency. The sine and cosine functions for the decomposition are called the basis functions of the decomposition.

The components of the spectrum of the DFT determine the amplitudes of the sinusoids that combine to form an image. At any given frequency in the DFT of an image, a large amplitude implies a greater prominence of a sinusoid of that frequency in the image. If an image has high contrast patterns, with intensity being non-uniform, its Fourier transform will have non-zero components

corresponding to high frequency sinusoids. If an image is smooth, with uniform intensity, then low-frequency sinusoids will have non-zero coefficients.

2D Fourier transform Let $f(x, y)$ be a continuous function of two continuous variables, x and y. The two-dimensional, continuous Fourier transform pair is given by the expressions

$$F(u, v) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y) e^{-\imath 2\pi(ux+vy)} \mathrm{d}x\mathrm{d}y \tag{10}$$

$$f(x, y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} F(u, v) e^{\imath 2\pi(ux+vy)} \mathrm{d}u\mathrm{d}v \tag{11}$$

Here u and v are the frequency variables. When referring to images, x and y are interpreted to be continuous spatial variables. As in the 1-D case, the domain of the variables u and v defines the continuous frequency domain.

Basis functions are 2-D waves.

Similar to 1D case, there exists 2-D Discrete Fourier Transform, which is used in image processing.

Fourier spectrum visualization Let's talk a bit how one can interpret visualization of Fourier spectrum in 2D case.

Fourier spectrum is an array of magnitudes of all Fourier coefficients $|F(u, v)|$. It is often rendered as an image itself for visualisation and interpretation. It reveals the presence of particular basis images in an image. Each position in the rendered spectrum corresponds to a particular basis function, defined by its frequencies u and v. If a value of $F(u, v)$ corresponding to that basis function is high, than it will correspond to bright spot in position (u, v).

The center of the spectrum corresponds to zero frequencies u and v. So if the center of the rendered spectrum is light, this means that the original image mostly contains areas with uniform intensities, without high contrast changes.

If the edges of the spectrum are light, this means that the original image has a lot of areas with high contrast changes.

the right hand side of the image contains visualization of basis functions placed on top of their respectful positions in rendered spectrum. So, if the right top corner of the spectrum is bright, this means that the original image contains strong edges that run approximately at 45°. If the right bottom corner of the spectrum is bright, this means that the original image contains strong edges directed at $-45°$.

Some practicalities Before we look at examples of images and their Fourier spectra, let's cover a few properties of image Fourier transform and typical tricks used to visualize Fourier spectrum of an image.

One property of the DFT which is used frequently is that the Fourier transform of a real function, $f(x, y)$, is conjugate symmetric:

$$F(u, v) = F^*(-u, -v) \tag{12}$$

This implies that the Fourier spectrum of an image has even symmetry about the origin:

$$|F(u, v)| = |F(-u, -v)| \tag{13}$$

As in the 1-D case, the 2-D Fourier transform and its inverse are infinitely periodic in the u and v directions. In case of DFT, in 1-D the transform data in the interval from 0 to M where M is a number of values of a discrete function, consists of two half periods meeting at point $M/2$. For display and filtering purposes, it is more convenient to have in this interval a complete period of the transform in which the data are contiguous and ordered properly. To achieve that, we need to multiply original spatial function $f(x)$ by $(-1)^x$. In 2-D case, to shift the data so that $F(0, 0)$ is in the center of the resulting frequency rectangle, we can multiply the original $f(x, y)$ function by $(-1)^{x+y}$. It is a common practice to do so.

Another interesting observation is that the zero-frequency term of the DFT

$$F(0, 0) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) \tag{14}$$

is essentially the average of $f(x, y)$, that is the average intensity of the original image.

It is typical for images to be composed of mostly low-frequency components. This results in bright spots in the middle of the spectrum, and very limited visibility for higher frequencies. For better visualization of the full spectrum, it is common to apply a log transformation to the spectrum before the visualization.

Visualization of Fourier spectrum: a natural image Now, let's look at some examples of images and their respectful Fourier spectra. This slide shows a natural image and its spectrum. From the visualization of the spectrum, we can say that the image doesn't have very sharp edges (there are almost no bright spots besides the central part of the spectrum). And also that this image contains edges of all directions - the bright spot in the middle looks like a circle, without deformations in one particular dimension. A horizontal line in the middle of the spectrum tells us that there should be vertical edges in the original image, and it is there due to the fact that vertical edges of the original image (right and left)

have different intensities. If you imagine the original image being just one period of an infinite periodic function, there is a sharp change of intensity between one copy of an image and the next one along the vertical edges - from very bright melon along the left edge to dark apples on the right. The intensities of the top and bottom edges are not as different, and as a result, a vertical line in the middle of the spectrum image is not as bright.

Visualization of Fourier spectrum: synthetic images Now let's take a look at a pair of synthetic images, one of them being a blurred version of another. You can see that the first top image has black and white stripes in horizontal direction, with very sharp transition from black to white. And we can observe presence of high frequency components in the image of its spectrum - you can see a bright vertical line across all values of v. It corresponds to sharp changes in intensity along vertical direction in the original image. There are no bright spots anywhere else, because there are no changes in intensity in any other direction on the original image. Also, in opposite to the previous slide, we don't see a horizontal line in the middle of the spectrum, because there are is discontinuity between left and right edges of this image.

Now let's look at the example in the bottom of the slide. It is a blurred version of the image above. there are no long sharp edges between black and white stripes, the transition is smooth. As a result, the spectrum of this image doesn't have high frequency components anymore. We have only fragment of a vertical line in the spectrum image, corresponding to smooth changes in intensity in vertical direction on the original image. And as in the top case, there are no other edges in the original image, and thus there are no other bight spots in the spectrum image.

Visualization of Fourier spectrum: more examples And this slide has a couple more examples of images with their spectra. Again, a natural image has more smooth intensity transitions compared to synthetic one, which results in its spectrum to be more concentrated around the central part, corresponding to low frequencies.

Translation and rotation This slide demonstrates a couple of other properties of DFT. The top row shows images of a rectangle the bottom row shows its corresponding spectra (after log transformation is applied). Two things are apparent here. The spectrum is insensitive to image translation (the spectrum images of the first two images are identical), but it rotates by the same angle of a rotated image. Again, the spectrum describes directions and frequencies of intensity changes in the original images. When rectangle is translated, neither directions no "sharpness" of its edges is changed. Hence, we have identical spectra.

When the rectangle is rotated, directions of edges on the image are changed - thy are rotated. As a result, we'll see bright spots in the spectrum corresponding to other directions on intensity changes.

Convolution theorem Fourier transform is important for signal processing not only because it gives us a tool to represent any image as a weighted sum of the same basis functions and thus provides another way to compare images, but also because it makes it possible to filter images in frequency domain. And convolution theorem plays here a key role.

Let's consider three three functions operating is spatial domain, f, g and h, such that g is a result of the spatial convolution of f and h. Now lets assume that F, G, and H are their corresponding Fourier transforms. Then $G = FH$. In other words, the Fourier transform of the convolution of two functions in the spatial domain is equal to the product in the frequency domain of the Fourier transforms of those two functions. Conversely, if we have the product of the two transforms, we can obtain the convolution in the spatial domain by computing the inverse Fourier transform.

And convolution in the frequency domain is analogous to multiplication in the spatial domain, the two being related by the forward and inverse Fourier transforms, respectively. The convolution theorem is the foundation for filtering in the frequency domain.

Convolving an image with a certain kernel has the same effect on that image as multiplying the spectrum of that image by the Fourier transform of the kernel. Therefore, linear filtering can always be performed either in the spatial or the spectral domains. Filtering in the spectral domain is computationally simpler because convolution in the spatial domain is replaced with the point-to-point multiplication in frequency domain. So instead of performing a computationally expensive operation of spatial convolution, it is cheaper to compute Fourier transforms of the original image and filter, multiply the results, and then apply the inverse Fourier Transform.

Filtering in frequency domain Filtering techniques in the frequency domain are based on modifying the Fourier transform to achieve a specific objective, and then computing the inverse DFT to get us back to the spatial domain.

To summarize, filtering in frequency domain consists of the following steps:

1. Multiply an input by $(-1)^x + y$, so that its Fourier transform is centered.

2. Compute the DTF, $F(u, v)$ of the image from step 1.

3. Form the product $G(u, v) = H(u, v)F(u, v)$ using elementwise multiplication, where $H(u, v)$ is a filter - a real symmetric function.

4. Compute the IDFT of $G(u, v)$.

5. Obtain the real part of the result from step 4.

6. Multiply the result of step 5 by $(-1)^x + y$.

Examples of filtering in the frequency domain Top left image in this slide is a scanning electron microscope image of an integrated circuit, and next to it on the right is its Fourier spectrum. If we look at the original image, we can note strong edges that run approximately at $\pm 45^\circ$ and two white. The Fourier spectrum shows prominent components along the $\pm 45^\circ$ directions that correspond to these edges. Also the original image shows two white, oxide protrusions resulting from thermally induced failure. Looking carefully along the vertical axis of the spectrum image, we can see a vertical component of the transform that is off-axis, slightly to the left. This component was caused by the edges of the oxide protrusions. Note how the angle of the frequency component with respect to the vertical axis corresponds to the inclination (with respect to the horizontal axis of the image) of the long white element. Note also the zeros in the vertical frequency component, corresponding to the narrow vertical span of the oxide protrusions.

These are typical of the types of associations we can make in general between the frequency and spatial domains.

One of the simplest filters we can construct is a function $H(u, v)$ that is 0 at the center of the (centered) transform, and 1's elsewhere. This filter would reject the term corresponding to $(u, v) = (0, 0)$ and "pass" (i.e., leave unchanged) all other terms of $F(u, v)$ when we form the product $H(u, v)$ and $F(u, v)$. Now recall that that the $F(0, 0)$ term is responsible for the average intensity of an image, so setting it to zero will reduce the average intensity of the output image to zero. Figure in the bottom of the slide shows the result of this operation. As expected, the image became much darker. An average of zero implies the existence of negative intensities. Therefore, although it illustrates the principle, it is not a true representation of the original, as all negative intensities were clipped to 0 by the display.

As it was mentioned earlier, low frequencies in the transform are related to slowly varying intensity components in an image, such as the walls of a room or a cloudless sky in an outdoor scene. On the other hand, high frequencies are caused by sharp transitions in intensity, such as edges and noise. Therefore, we would expect that a function $H(u, v)$ that zeros high frequencies while passing low frequencies (called a lowpass filter, as noted before) would blur an image, while a filter with the opposite property (called a highpass filter) would enhance sharp detail, but cause a reduction in contrast in the image. This slides illustrates these effects. The first column shows a lowpass filter and the corresponding filtered image. The second column shows similar results for a highpass filter. Note that the

result of the highpass filter is dark. The reason is that this highpass filter function eliminates the $F(0,0)$ term, resulting in the same basic effect was demonstrated on the previous slide. In the third column, we have a similar filter but with addition of a small constant. This addition does not affect sharpening, but it does prevent elimination of the $F(0,0)$ term and thus preserves tonality.

Detection of discontinuities The filters (both spatial and spectral) can be used to enhance images, and also to highlight some properties of interest. Two basic properties of image intensity values, discontinuity and similarity, are key to many image processing and image analysis methods. Traditional image segmentation algorithms are based on these properties.

Sharp, local changes in intensity are common indicators of an edge, object boundary. Image areas with discontinuities contain more semantic information than homogeneous areas. Often, based on object boundary only, one can derive semantic class of that object. So, discontinuities are important, and for many years traditional approaches to detect discontinuities have been fundamental to image analysis.

The three types of image characteristics in which we are interested are isolated points, lines, and edges. Edge pixels are pixels at which the intensity of an image changes abruptly, and edges (or edge segments) are sets of connected edge pixels. Edge detectors are local image processing tools designed to detect edge pixels. A line may be viewed as a thin edge segment in which the intensity of the background on either side of the line is either much higher or much lower than the intensity of the line pixels. In fact, as we will discuss later, lines give rise to so-called "roof edges." Finally, an isolated point may be viewed as a foreground (background) pixel surrounded by background (foreground) pixels.

Using derivatives So, we want to detect sharp, local changes in intensity. It is intuitive that they can be detected using derivatives - a great mathematical tool to detect discontinuities and abrupt, local changes of a function. First- and second-order derivatives are particularly well suited for this purpose.

Let's look at a ramp edge fragment, with smoothly changing intensity. Under the fragment you can see its intensity profile. And on the right the first and second derivatives of this intensity profile. Moving from left to right along the intensity profile, we note that the first derivative is positive at the onset of the ramp and at points on the ramp, and it is zero in areas of constant intensity. The second derivative is positive at the beginning of the ramp, negative at the end of the ramp, zero at points on the ramp, and zero at points of constant intensity. The intersection between the zero intensity axis and a line extending between the extrema of the second derivative marks a point called the zero crossing of the second derivative.

Derivatives of a digital function can be defined in terms of finite differences. The first-order derivative can be approximated as a difference in intensity between adjacent pixels:

$$f'(x) = f(x+1) - f(x) \tag{15}$$

This approximation can be derived using Taylor Series of $f(x+1)$.

Similarly, we can obtain an approximation of the second-order derivative:

$$f''(x) = f(x+1) - 2f(x) + f(x+1) \tag{16}$$

Example Let's look at an image fragment and its simplified intensity profile below. It'll help us to understand how the first- and second-order derivatives behave as they encounter a point, a line, and the edges of objects. Ler's traverse this simplified profile from left to right. Initially, the first-order derivative is nonzero at the onset and along the entire intensity ramp, while the second-order derivative is nonzero only at the onset and end of the ramp. Because the edges of digital images resemble this type of transition, we conclude that first-order derivatives produce "thick" edges, and second-order derivatives much thinner ones. Next we encounter the isolated noise point. Here, the magnitude of the response at the point is much stronger for the second- than for the first-order derivative. Thus, we can expect second-order derivatives to enhance fine detail (including noise) much more than first-order derivatives. The line in this example is rather thin, so it too is fine detail, and we see again that the second derivative has a larger magnitude. Finally, note in both the ramp and step edges that the second derivative has opposite signs (negative to positive or positive to negative) as it transitions into and out of an edge. This "double-edge" effect is an important characteristic that can be used to locate edges. As we move into the edge, the sign of the second derivative is used also to determine whether an edge is a transition from light to dark (negative second derivative), or from dark to light (positive second derivative).

In summary, we arrive at the following conclusions:

1. First-order derivatives generally produce thicker edges.

2. Second-order derivatives have a stronger response to fine detail, such as thin lines, isolated points, and noise.

3. Second-order derivatives produce a double-edge response at ramp and step transitions in intensity.

4. The sign of the second derivative can be used to determine whether a transition into an edge is from light to dark or dark to light.

To compute first and second derivatives at every pixel location in an image it is common to use spatial convolution with specially designed filters. Let's look at a few examples.

Detection of isolated points An isolated point is characterized by sharp changes in intensity along all directions. It can be detected with a filter similar to the one presented on the slide. With this filter we will simply measure the weighted differences between a pixel and its 8-neighbors. Intuitively, the idea is that the intensity of an isolated point will be quite different from its surroundings, and thus will be easily detectable by this type of kernel. Note that the coefficients of the filter sum to zero, indicating that the filter response will be zero in areas of constant intensity.

Differences in intensity that are considered of interest are those that large enough, above a certain threshold. A point has been detected at a location (x, y) on which the kernel is centered if the absolute value of the response of the filter at that point exceeds a specified threshold. Such points are labeled 1 and all others are labeled 0 in the output image, thus producing a binary image.

This slide shows an original image, the response of the filter, and binary result image after a threshold is applied.

Laplacian The filter used in the previous slide to detect isolated points is a Laplacian kernel - the simplest derivative operator, which, for a function (image) $f(x, y)$ of two variables, is defined as:

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \tag{17}$$

To express this equation in discrete form, we can use discrete forms of second-order derivatives. The discrete Laplacian of two variables can be defined as:

$$\nabla^2 f(x, y) = f(x+1, y) + f(x-1, y) + f(x, y+1) + f(x, y-1) - 4f(x, y) \tag{18}$$

This equation can be implemented using convolution with the kernel on the left. This kernel is isotropic for rotations in increments of 90° with respect to the x- and y-axes. This means its response is independent of the direction of intensity discontinuities along x- and y-axes . To make it isotropic along diagonals too, we can add four more terms and adjust the central term so the total sum of all elements in the kernel is zero.

Laplacian kernels are frequently used for image sharpening.

Line detection Lines can be detected in a similar way as isolated points. According to our previous observations, we can expect second derivatives to result in a stronger filter response, and to produce thinner lines than first derivatives. Thus, we can use the Laplacian kernel for line detection also.

The leftmost image on the slide is a binary portion of a wire-bond mask for an electronic circuit. The next image shows its Laplacian. The Laplacian contains negative values, so to display it properly it was scaled: mid gray represents zero, darker shades of gray represent negative values, and lighter shades are positive. The double-line effect is clearly visible in the magnified region. At first, it might appear that the negative values can be handled simply by taking the absolute value of the Laplacian image. The next image shows absolute values of the Laplacian, As you can see, this approach doubles the thickness of the lines. A more suitable approach is to use only the positive values of the Laplacian. the rightmost image shows, this approach results in thinner lines that generally are more useful. You can also notice, that when the lines are wide with respect to the size of the Laplacian kernel, the lines are separated by a zero "valley." This is not unexpected. For example, when the 3 3 kernel is centered on a line of constant intensity 5 pixels wide, the response will be zero. So the demonstrated approach is useful for detecting lines which are thin with respect to the size of the detector. Lines that do not satisfy this assumption can be handled by the edge detection methods, which we will discuss next.

But before that, let's look at another example of line detection.

Detecting lines of specific orientations The Laplacian kernel is isotropic, so its response is independent of direction (with respect to the four directions of the 3 3 kernel: vertical, horizontal, and two diagonals). Sometimes, lines of only one specific direction are of interest. How can we detect those?

Consider the kernels presented on the slide. Suppose that an image with a constant background and containing various horizontal, vertical and diagonal lines is filtered with the first kernel. The maximum responses would occur at image locations in which a horizontal line passes through the middle row of the kernel. This is easily verified by sketching a simple array of 1's with a line of a different intensity (say, 5s) running horizontally through the array. A similar experiment would reveal that the second kernel in the row responds best to lines oriented at $+45°$, the third kernel to vertical lines, and the fourth kernel to lines in the $-45°$ direction. The preferred direction of each kernel is weighted with a larger coefficient (i.e., 2) than other possible directions. The coefficients in each kernel sum to zero, indicating a zero response in areas of constant intensity.

Suppose that we are interested in finding all the lines that are one pixel thick and oriented at $+45°$ in the left image in the second row. For this purpose, we use the second kernel in the top row. The image in the middle of the bottom row

is the result of filtering with that kernel. There are two principal segments in the image oriented in the +45° direction, one in the top left and one at the bottom right. The straight line segment at the bottom right is brighter than the segment in the top left because the line segment in the bottom right of the original image is one pixel thick, while the one at the top left is not. The kernel is "tuned" to detect one-pixel-thick lines in the +45° direction, so we expect its response to be stronger when such lines are detected. Because we are interested in the strongest response, we set a high threshold T, equal to 254 (the maximum value minus one). Bottom right image shows in white the points whose values passed the threshold - we can see our one pixel thick line oriented at +45° being properly detected.

Edge detection Now let's look at different types or models of edges. They can be classified according to their intensity profiles. An ideal step edge is characterized by a transition between two intensity levels occurring over the distance of one pixel. Left image on the slide shows a section of a vertical step edge and a horizontal intensity profile through the edge. Step edges occur, for example, in images generated by a computer for use in areas such as solid modeling and animation. These clean, ideal edges can occur over the distance of one pixel, provided that no additional processing (such as smoothing) is used to make them look "real."

In practice, digital images have edges that are blurred and noisy. In such situations, edges can be more closely modeled as having an intensity ramp profile, such as the edge on the second image on the slide. The slope of the ramp is inversely proportional to the degree to which the edge is blurred. In this model, we no longer have a single "edge point" along the profile. Instead, an edge point now is any point contained in the ramp, and an edge segment would then be a set of such points that are connected.

A third type of edge is the so-called roof edge, having the characteristics illustrated on the last image. Roof edges represent lines, with the width of the edge being determined by the thickness and sharpness of the line. In the limit, when its base is one pixel wide, a roof edge is nothing more than a one-pixel-thick line running through a region in an image.

Images often contain all these three types of edges.

Gradient By now we know that we can detect changes in intensity for the purpose of finding edges using first- or second-order derivatives. So far we've been mostly talking about 1D intensity profiles. But images are 2D. So, the tool of choice for finding edge strength and direction at an arbitrary location (x, y) of an image f is the gradient, defined as the vector

$$\nabla f(x,y) = \begin{bmatrix} \frac{\partial f(x,y)}{\partial x} \\ \frac{\partial f(x,y)}{\partial y} \end{bmatrix} \tag{19}$$

This vector has the well-known property that it points in the direction of maximum rate of change of f at (x, y). So for vertical edges, the second element of this vector will be 0, showing that partial derivative with respect to y is zero. For horizontal edges the first element of this vector is zero, because the partial derivative with respect to x is zero.

When evaluated for all applicable values of x and y, $f(x, y)$ becomes a vector image, each element of which is a gradient vector. The magnitude of this gradient vector at a point (x, y) is given by its Euclidean vector norm. This is the value of the rate of change in the direction of the gradient vector at point (x, y). Note that arrays of partial derivatives, gradient and its magnitude are of the same size as f, created when x and y are allowed to vary over all pixel locations in f. It is common practice to refer to the magnitude array as the gradient image.

The direction of the gradient vector at a point (x, y) is given by arctangent of the ratio of partial derivatives with respect to y and x.

Computing partial derivatives via convolution Obtaining the gradient of an image requires computing the partial derivatives $\partial f/\partial x$ and $\partial f/\partial y$ at every pixel location in the image. A partial derivative of a function of two variables $f(x, y)$ with respect to variabl x is defined as

$$\frac{\partial f}{\partial x} = \lim_{x \to \epsilon} \left(\frac{f(x+\epsilon, y)}{\epsilon} - \frac{f(x,y)}{\epsilon} \right) \tag{20}$$

For discrete function, we can obtain an approximation to the first-order derivative via finite difference

$$\frac{\partial f}{\partial x} \approx f(x+1, y) - f(x,y) \tag{21}$$

Partial derivative is linear and invariant to translation. Which means that it can be represented as convolution. In particular, this equation for $\partial f/\partial x$ can be implemented by filtering $f(x, y)$ with the 1-D kernel presented on the slide.

Similarly, partial derivative with respect to y

$$\frac{\partial f}{\partial y} \approx f(x, y+1) - f(x,y) \tag{22}$$

can be computed as a result of filtering with vertical kernel.

Gradient operators This slides shows a few other popular gradient operators.

When diagonal edge direction is of interest, we need 2-D kernels. The Roberts cross-gradient operators are one of the earliest attempts to use 2-D kernels with a diagonal preference. Consider the 3×3 mask on the top of the slide. The Roberts operators are based on implementing the diagonal differences

$$\frac{\partial f}{\partial x} = (z_9 - z_5) \tag{23}$$

and

$$\frac{\partial f}{\partial y} = (z_8 - z_6) \tag{24}$$

These derivatives can be implemented by filtering an image with the kernels shown on the slide.

Kernels of size 2×2 are simple conceptually, but they are not as useful for computing edge direction as kernels that are symmetric about their centers, the smallest of which are of size 3×3. These kernels take into account the nature of the data on opposite sides of the center point, and thus carry more information regarding the direction of an edge. The simplest digital approximations to the partial derivatives using kernels of size 3×3 are Prewitt operators, also shown on the slide. In this formulation, the difference between the third and first rows of the 3×3 region approximates the derivative in the x-direction, and the difference between the third and first columns approximate the derivative in the y-direction. Intuitively, these approximations are more accurate than the approximations obtained using the Roberts operators.

A slight variation of the preceding two equations uses a weight of 2 in the center coefficient. Using a 2 in the center location provides image smoothing. These kernels are called the Sobel operators.

The Prewitt kernels are simpler to implement than the Sobel kernels, but the slight computational difference between them typically is not an issue. The fact that the Sobel kernels have better noise-suppression (smoothing) characteristics makes them preferable.

Note that coefficients of all the kernels on this slide sum to zero, thus giving a response of zero in areas of constant intensity, as expected of derivative operators.

Example Any of the pairs of kernels from previous slide are convolved with an image to obtain the gradient components $G_x = \partial f/\partial x$ and $G_y = \partial f/\partial y$ at every pixel location. These two partial derivative arrays are then used to estimate edge strength and direction. Obtaining the magnitude of the gradient requires the computation of Euclidean vector norm, which requires computing squares and

square roots, which is expensive. An approach used frequently is to approximate the magnitude of the gradient by absolute values:

$$M(x, y) \approx |G_x| + |G_y| \tag{25}$$

This equation is more attractive computationally, and it still preserves relative changes in intensity levels.

Let's look at an example. The top left image on the slide is an original image with intensity values scaled to the range $[0, 1]$. The top right image is the component of the gradient in the x-direction, and the bottom left is the component of the gradient in the y-direction, both obtained using corresponding Sobel kernels. And the bottom right image is the gradient image formed from the sum of these two components. The directionality of the horizontal and vertical components of the gradient is evident in the images of partial gradients. Note, for example, how strong the roof tile, horizontal brick joints, and horizontal segments of the windows are in top right image compared to other edges. In contrast, the bottom left image favors features such as the vertical components of the façade and windows.

It is common terminology to use the term edge map when referring to an image whose principal features are edges, such as gradient magnitude images.

Derivatives in noisy images The edge models we've been using so far were free of noise. However, natural images are often noisy, and derivatives can amplify that noise. Let's look at what happens to derivatives of intensity profiles with noise.

The image segments in the row on this slide show close-ups of four ramp edges that transition from a black region on the left to a white region on the right (keep in mind that the entire transition from black to white is a single edge). The leftmost image segment is free of noise. The other three images are corrupted by additive Gaussian noise with zero mean and standard deviation of 0.1, 1.0, and 10.0 intensity levels, respectively. The graph below each image is a horizontal intensity profile passing through the center of the image. All images have 8 bits of intensity resolution, with 0 and 255 representing black and white, respectively.

These Image fragments look very similar. Even the noisiest rightmost image is clearly perceived by humans as a ramp transition from black to white.

Now let's look at the derivatives for these images. The first derivative of the image fragment without noise looks as expected - it is zero in the constant areas, which corresponds to the two black bands shown in the derivative image. The derivatives at points on the ramp are constant and equal to the slope of the ramp. These constant values in the derivative image are shown in gray.

As we move to images with greater and greater noise, the derivatives become increasingly different from the noiseless case. In fact, it would be difficult to

associate the derivative image and profile of the last image in the row with the first one of a noiseless ramp edge. At the same time, the noise is almost visually undetectable in the original images in the top row. this example shows us the first derivatives are very sensitive to noise.

And what about second-order derivatives? They are even more sensitive to noise. The second derivative of the noiseless image is shown at the left bottom corner of the slide. The thin white and black vertical lines are the positive and negative components of the second derivative. The gray in these images represents zero. The only noisy second derivative image that barely resembles the noiseless case corresponds to noise with a standard deviation of 0.1. The remaining second-derivative images and profiles clearly illustrate that it would be extremely difficult to detect their positive and negative components.

Noise impact on derivatives We've seen that even very little visual noise can have a significant impact on the two key derivatives used for detecting edges. Due to noise, values of adjacent pixels can be very different, and this difference will be picked up by kernels based on finite differences. Greater noise will result in greater responses of derivative filters.

What can be done here? Image smoothing can help. Image smoothing filters should be applied prior to the use of derivatives in applications with noisy images.

Smoothing before derivatives Let's look at another example. Here we see a noisy function and its first derivative. It's impossible to detect edge here.

Now, let's apply a Gussian smoothing kernel. Here the first row corresponds to the same noisy function as before. The second row shows Gaussian filter which we would like to apply. The third row is a result of convolution of our noisy function with the Gaussian kernel. And the last row is the first derivative of the filtered signal. In contrast to the derivative of the original signal, this one can be used to detect edges. To detect an edge, we just need to find an extremum of the derivative.

Using convolution properties Convolution and differentiation are associative:

$$\frac{\mathrm{d}}{\mathrm{d}x}(f \star g) = f \star \frac{\mathrm{d}}{\mathrm{d}x} g \tag{26}$$

This means, we can change order of these operations. And if we pre-compute the derivative of Gaussian, we can apply that filter directly to a noisy image, instead of smoothing with Gaussian kernel first and then applying a derivative kernel.

Gaussian derivative The Gaussian derivative function is widely used in image analysis. It has many interesting properties.

Here on this picture you can see a plot of a 2D Gaussian and its first-order derivative with respect to x . As discussed before, the first-order derivative with respect to x can be computed via convolution with filter $[1, -1]$. The result of applying this filter to Gaussian is on the slide. Instead of one "bump" of the original function, its derivative has two "bumps" in opposite directions.

Here we can see both partial derivatives - with respect to x and with respect to y. If we render those as images, we will get images in the bottom row. Gaussian derivatives are closely related to Gabor filters - widely used for texture analysis of images in traditional computer vision.

Smoothing and edge detection Although smoothing before differentiation is extremely useful for noise suppression, it also blurs edges. So wider Gaussian derivative operator results in smoothing over the greater area, and thus better suppresses noise, but at the same time it results in more blur in the edges map. It also shifts edge locations. On this slide we see a comparison of three different Gaussian derivative operators of larger and larger size.

Laplacian of a Gaussian Similar to Gaussian derivative, we can combine smoothing with a Gaussian filter and edge detection with the second-order derivative, or Laplacian. $\nabla^2 G$ - Laplacian of Gaussian, often shorten to LoG, is another popular edge detection operator.

$$\nabla^2 G(x, y) = \left(\frac{x^2 + y^2 - 2\sigma^2}{\sigma^4} \right) e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{27}$$

The slide shows a 3-D plot, image, and cross-section of the negative of the LoG function. On the cross-section plot, you can notice that the zero crossings of the LoG occur at points $x^2 + y^2 = 2\sigma^2$, which defines a circle of radius 2σ centered on the peak of the Gaussian function. Because of its shape, the LoG function sometimes is called the Mexican hat operator. You can also see on the slide a 5 5 kernel that approximates the shape of LoG. This approximation is not unique. Its purpose is to capture the essential shape of the LoG function: a positive, central term surrounded by an adjacent, negative region whose values decrease as a function of distance from the origin, and a zero outer region. The coefficients must sum to zero so that the response of the kernel is zero in areas of constant intensity.

Filter kernels of arbitrary size (and at the same time of fixed σ) can be generated by sampling LoG function and scaling the coefficients so that they sum to zero.

Gradient is not ideal edge detector Although gradient, Gaussian gradient and LoG can be used to enhance edges on the images, they are not ideal edge detectors. These operators result in shifted edge positions. And they do not produce continuous edges. Like on this picture, you can see that edges are not ideally detected.

Properties of a good edge detector Let's think about desirable properties of a good edge detector.

1. Good detection, low error rate: all edges should be found, and there should be no spurious responses. A blue example here on the slide is an example of a poor detector, with some edge points undetected, and others being very far from the true edge.

2. Good localization: edge points should be well localized. The edges located must be as close as possible to the true edges. That is, the distance between a point marked as an edge by the detector and the center of the true edge should be minimum. A green example is an example of poor localization. Detected edge points are not exactly where the true edge is.

3. Single edge point response. The detector should return only one point for each true edge point. That is, the number of local maxima around the true edge should be minimum. This means that the detector should not identify multiple edge pixels where only a single edge point exists. the black example demonstrates too many responses to edge points.

Canny detector is one of the popular edge detectors, which was designed to satisfy the properties listed on this slide.

Canny detector The Canny edge detection algorithm consists of the following steps:

1. Apply a Gaussian derivative operator to an input image: this will smooth the image, remove noise and detect discontinuities in intensity .

2. Compute the gradient magnitude and angle images.

3. Apply nonmaxima suppression to the gradient magnitude image. Gradient image typically contains wide ridges around local maxima. This step is to thin those ridges. The essence of this approach is to specify a number of discrete orientations of the edge normal (gradient vector) and keep only points with the largest magnitude for every orientation.

4. Use double thresholding and connectivity analysis to detect and link edges. Canny's algorithm uses hysteresis thresholding, which uses two thresholds: a low threshold, T_L and a high threshold, T_H. If a pixel passes the high threshold, it is considered a "strong" edge pixel, pixels passing the low threshold are considered "weak" edge pixels. Every strong pixel is assumed to be a valid edge pixel. A weak esge pixel is marked as a valid edge pixel, only if it is connected to another strong edge pixel.

The performance of the Canny edge detector is superior to other edge detectors we discussed so far.

Example Let's look at results of every step of Canny edge detector applied to the Lena image.

The first step is to apply Gaussian derivative operator and compute its magnitude.

The next step is to apply thresholding.

And the final step is to apply non-maxima suppression.

And one more example to compare the results of edge detection using Sobel operator and Canny edge detection algorithm. Note, how the result of Canny edge detector has thinner edges.