Introduction to Image Processing
Part 1/2 – Images, pixels and sampling

Rafał Mantiuk
Computer Laboratory, University of Cambridge
What are Computer Graphics & Image Processing?

- Scene description
- Digital image
- Image capture
- Image display
- Computer graphics
- Image analysis & computer vision
- Image processing
Where are graphics and image processing heading?
What is a (computer) image?

- A digital photograph? (“JPEG”)
- A snapshot of real-world lighting?

From computing perspective (discrete):
- 2D array of pixels
  - To represent images in memory
  - To create image processing software

From mathematical perspective (continuous):
- 2D function
  - To express image processing as a mathematical problem
  - To develop (and understand) algorithms
Image

- 2D array of pixels
- In most cases, each pixel takes 3 bytes: one for each red, green and blue
- But how to store a 2D array in memory?
Stride

- Calculating the pixel component index in memory
  - For row-major order (grayscale)
    \[ i(x, y) = x + y \cdot n_{cols} \]
  - For column-major order (grayscale)
    \[ i(x, y) = x \cdot n_{rows} + y \]
  - For interleaved row-major (colour)
    \[ i(x, y, c) = x \cdot 3 + y \cdot 3 \cdot n_{cols} + c \]
  - General case
    \[ i(x, y, c) = x \cdot s_x + y \cdot s_y + c \cdot s_c \]

where \( s_x, s_y \) and \( s_c \) are the strides for the \( x, y \) and colour dimensions.
Padded images and stride

- Sometimes it is desirable to “pad” image with extra pixels
  - for example when using operators that need to access pixels outside the image border
- Or to define a region of interest (ROI)

- How to address pixels for such an image and the ROI?
Padded images and stride

\[ i(x, y, c) = i_{first} + x \cdot s_x + y \cdot s_y + c \cdot s_c \]

- For row-major, interleaved
  - \( s_x = ? \)
  - \( s_y = ? \)
  - \( s_c = ? \)
Pixel (PIcture EElement)

- Each pixel (usually) consist of three values describing the color
  
  \[ \text{red, green, blue} \]

- For example
  
  - \((255, 255, 255)\) for white
  - \((0, 0, 0)\) for black
  - \((255, 0, 0)\) for red

- Why are the values in the 0-255 range?

- Why red, green and blue? (and not cyan, magenta, yellow)

- How many bytes are needed to store 5MPixel image? (uncompressed)
Pixel formats, bits per pixel, bit-depth

- Grayscale – single **color channel**, 8 bits (1 byte)
- Highcolor – $2^{16}=65,536$ colors (2 bytes)
- Truecolor – $2^{24} = 16,8$ million colors (3 bytes)
- Deepcolor – even more colors (>= 4 bytes)

**But why?**
Color banding

- If there are not enough bits to represent color
- Looks worse because of the **Mach band** illusion
- Dithering (added noise) can reduce banding
  - Printers
  - Many LCD displays do it too
What is a (computer) image?

- A digital photograph? (“JPEG”)
- A snapshot of real-world lighting?

**From computing perspective (discrete)**
- 2D array of pixels
  - To represent images in memory
  - To create image processing software

**From mathematical perspective (continuous)**
- 2D function
  - To express image processing as a mathematical problem
  - To develop (and understand) algorithms
Image – 2D function

- Image can be seen as a function $I(x,y)$, that gives intensity value for any given coordinate $(x,y)$. 
Sampling an image

- The image can be sampled on a rectangular sampling grid to yield a set of samples. These samples are pixels.
What is a pixel? (math)

- A pixel is not
  - a box
  - a disk
  - a teeny light

- A pixel is a point
  - it has no dimension
  - it occupies no area
  - it cannot be seen
  - it has coordinates

- A pixel is a sample

From: http://groups.csail.mit.edugraphics/classes/6.837/F01/Lecture05/lecture05.pdf
Sampling and quantization

- Physical world is described in terms of continuous quantities
- But computers work only with discrete numbers
- Sampling – process of mapping continuous function to a discrete one
- Quantization – process of mapping continuous variable to a discrete one
Resampling

- Some image processing operations require to know the colors that are in-between the original pixels.

- What are those operations?
- How to find these resampled pixel values?
Example of resampling: magnification

Input image

Output image
Example of resampling: scaling and rotation
How to resample?

- In 1D: how to find the most likely resampled pixel value knowing its two neighbors?
(Bi)Linear interpolation (resampling)

- Linear – 1D
- Bilinear – 2D

![Diagram of pixel value interpolation](image)
(Bi)cubic interpolation (resampling)
Bi-linear interpolation

Given the pixel values:

\[ I(x_1, y_1) = A \]
\[ I(x_2, y_1) = B \]
\[ I(x_1, y_2) = C \]
\[ I(x_2, y_2) = D \]

Calculate the value of a pixel \( I(x, y) = ? \) using bi-linear interpolation.

Hint: Interpolate first between A and B, and between C and D, then interpolate between these two computed values.
Advanced Graphics & Image Processing

Introduction to Image Processing

Part 2/2 – Point ops, filters and pyramids

Rafał Mantiuk

Computer Laboratory, University of Cambridge
Point operators and filters

Original

Blurred

Sharpened

Edge-preserving filter
Point operators

- Modify each pixel independent from one another
- The simplest case: multiplication and addition

\[
g(\mathbf{x}) = a f(\mathbf{x}) + b
\]
Pixel precision for image processing

- Given an RGB image, 8-bit per color channel (uchar)
- What happens if the value of 10 is subtracted from the pixel value of 5?
- $250 + 10 = ?$
- How to multiply pixel values by 1.5?
  - a) Using floating point numbers
  - b) While avoiding floating point numbers
Image blending

- Cross-dissolve between two images

\[ g(x) = (1 - \alpha)f_0(x) + \alpha f_1(x) \]

- where \( \alpha \) is between 0 and 1
Image matting and compositing

- Matting – the process of extracting an object from the original image
- Compositing – the process of inserting the object into a different image
- It is convenient to represent the extracted object as an RGBA image
Transparency, alpha channel

- RGBA – red, green, blue, alpha
  - alpha = 0 – transparent pixel
  - alpha = 1 – opaque pixel

- Compositing
  - Final pixel value:

\[ P = \alpha C_{pixel} + (1 - \alpha)C_{background} \]

- Multiple layers:

\[ P_0 = C_{background} \]
\[ P_i = \alpha_i C_i + (1 - \alpha_i)P_{i-1} \quad i = 1..N \]
Image histogram

- histogram / total pixels = probability mass function
  - what probability does it represent?
Histogram equalization

- Pixels are non-uniformly distributed across the range of values

- Would the image look better if we uniformly distribute pixel values (make the histogram more uniform)?
- How can this be done?
Histogram equalization

- Step 1: Compute image histogram

- Step 2: Compute a normalized cumulative histogram

\[ c(I) = \frac{1}{N} \sum_{i=0}^{I} h(i) \]

- Step 3: Use the cumulative histogram to map pixels to the new values (as a look-up table)

\[ Y_{out} = c(Y_{in}) \]
Linear filtering (revision)

- Output pixel value is a weighted sum of neighboring pixels

\[ g(i, j) = \sum_{k,l} f(i - k, j - l)h(k, l) \]

compact notation \[ g = f \ast h \]
Linear filter: example

Why is the matrix $g$ smaller than $f$?
Padding an image

Padded image

Padded and blurred image

zero  wrap  clamp  mirror

blurred: zero  normalized zero  clamp  mirror
What is the computational cost of the convolution?

\[ g(i, j) = \sum_{k,l} f(i - k, j - l)h(k, l) \]

- How many multiplications do we need to do to convolve 100x100 image with 9x9 kernel?
  - The image is padded, but we do not compute the values for the padded pixels
Separable kernels

- Convolution operation can be made much faster if split into two separate steps:
  - 1) convolve all rows in the image with a 1D filter
  - 2) convolve columns in the result of 1) with another 1D filter
- But to do this, the kernel must be separable

\[
\begin{bmatrix}
h_{11} & h_{12} & h_{13} \\
h_{21} & h_{22} & h_{23} \\
h_{31} & h_{32} & h_{33}
\end{bmatrix} =
\begin{bmatrix}
u_1 \\
u_2 \\
u_3
\end{bmatrix} \cdot
\begin{bmatrix}
v_1 & v_2 & v_3
\end{bmatrix}
\]

\[\hat{h} = \hat{u} \cdot \hat{v}\]
Examples of separable filters

- Box filter:
  \[
  \begin{bmatrix}
  1 & 1 & 1 \\
  9 & 9 & 9 \\
  1 & 1 & 1 \\
  9 & 9 & 9 \\
  1 & 1 & 1 \\
  9 & 9 & 9 
  \end{bmatrix}
  =\begin{bmatrix}
  1 \\
  3 \\
  1 \\
  3 \\
  1 \\
  3 
  \end{bmatrix} \cdot
  \begin{bmatrix}
  1 & 1 & 1 \\
  3 & 3 & 3 
  \end{bmatrix}
  \]

- Gaussian filter:
  \[
  G(x, y; \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}
  \]

- What are the corresponding 1D components of this separable filter \(u(x)\) and \(v(y)\)?

  \[G(x, y) = u(x) \cdot v(y)\]
Unsharp masking

- How to use blurring to sharpen an image?

\[ g_{\text{sharp}} = f + \gamma(f - h_{\text{blur}} \ast f) \]
Why “linear” filters?

- Linear functions have two properties:
  - Additivity: \( f(x) + f(y) = f(x + y) \)
  - Homogenity: \( f(ax) = af(x) \) (where “f” is a linear function)

- Why is it important?
  - Linear operations can be performed in an arbitrary order
    \[ \text{blur}(aF + b) = a \text{blur}(F) + b \]
  - Linearity of the Gaussian filter could be used to improve the performance of your image processing operation
  - This is also how the separable filters work:

\[ (u \cdot v) * f = u * (v * f) \]
Operations on binary images

- Essential for many computer vision tasks

- Binary image can be constructed by thresholding a grayscale image

\[
\theta(f, c) = \begin{cases} 
1 & \text{if } f \geq c, \\
0 & \text{else},
\end{cases}
\]
Morphological filters: dilation

- Set the pixel to the maximum value of the neighboring pixels within the structuring element
- What could it be useful for?
Morphological filters: erosion

- Set the value to the minimum value of all the neighboring pixels within the structuring element
- What could it be useful for?
Morphological filters: opening

- Erosion followed by dilation
- What could it be useful for?
Morphological filters: closing

- Dilation followed by erosion
- What could it be useful for?
Binary morphological filters: formal definition

\[ c = f \otimes s \]

- **dilation**: \( \text{dilate}(f, s) = \theta(c, 1) \);
- **erosion**: \( \text{erosode}(f, s) = \theta(c, S) \);
- **majority**: \( \text{maj}(f, s) = \theta(c, S/2) \);
- **opening**: \( \text{open}(f, s) = \text{dilate}(\text{erosode}(f, s), s) \);
- **closing**: \( \text{close}(f, s) = \text{erosode}(\text{dilate}(f, s), s) \).

\[ \theta(f, c) = \begin{cases} 1 & \text{if } f \geq c, \\ 0 & \text{else}, \end{cases} \]
Multi-scale image processing (pyramids)

- Multi-scale processing operates on an image represented at several sizes (scales)
  - Fine level for operating on small details
  - Coarse level for operating on large features

- Example:
  - Motion estimation
    - Use fine scales for objects moving slowly
    - Use coarse scale for objects moving fast
  - Blending (to avoid sharp boundaries)
Two types of pyramids

Gaussian pyramid

Level 1
Level 2
Level 3
Level 4

Laplacian pyramid (a.k.a. DoG Difference of Gaussians)

Level 1
Level 2
Level 3
Level 4 (base band)

Gaussian Pyramid

Blur the image and downsample (take every 2\textsuperscript{nd} pixel)

Why is blurring needed?
Laplacian Pyramid - decomposition
Laplacian Pyramid - synthesis
Reduce and expand

**Reduce**
- Filter rows
- Subsample rows
- Filter columns
- Subsample rows

**Expand**
- Upsample rows
- Filter rows
- Upsample columns
- Filter columns

Frequency response of Laplacian pyramid bands

\[ K = \]

![Graph showing frequency response of Laplacian pyramid bands](frequency_response_chart.png)
Example: stitching and blending

Combine two images:

Image-space blending

Laplacian pyramid blending
References

  - Chapter 3
  - [http://szeliski.org/Book](http://szeliski.org/Book)