EECS 4422/5323 Computer Vision

Image Representation Lecture 1

Calden Wloka

11 September, 2019

Announcments

- The schedule on the course website has been updated
- A submission policy has been added to the syllabus
- More suggested engineering projects have been added

A Local Conference

Mathematics of Vision Workshop

Final Thoughts from Monday

- We now begin looking at directly processing and analyzing image content

Final Thoughts from Monday

- We now begin looking at directly processing and analyzing image content
- Image capture is not the focus of this course, but it is important to be aware of the vocabulary and concepts

Final Thoughts from Monday

- We now begin looking at directly processing and analyzing image content
- Image capture is not the focus of this course, but it is important to be aware of the vocabulary and concepts
    - The more you can control your input, the easier your job will be

Final Thoughts from Monday

- We now begin looking at directly processing and analyzing image content
- Image capture is not the focus of this course, but it is important to be aware of the vocabulary and concepts
    - The more you can control your input, the easier your job will be
    - There are rich research questions in image capture, and it is an industry area with high demand

Outline

- Topic Introduction
- Pixels and Colour Spaces
- Point Operators
- Linear Filtering

Image Representation

Once an image is captured by a camera or otherwise obtained, we must represent the visual content in some manner. This is most commonly in the form of a matrix of *pixels*.

However, in order to analyze an image (whether computationally or as a human observer), it is often advantageous to change or manipulate the image representation. While sometimes these changes will be reversible, often they are not.

The Pixel

A pixel is the basic building block of a digital image.

Image source: CIFAR10 dog3

For any given pixel p, there are a number of important attributes associated with it:

- Coordinates, *e.g.* (x, y)
- Value, *e.g.* $[R, G, B]$
- Sometimes others, *e.g.* α

Pixel Coordinates

It is important to know the convention under which you are operating!

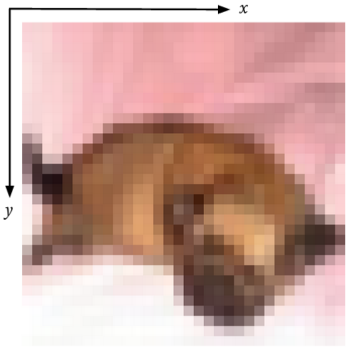


Image source: CIFAR10 dog3

Most common conventions follow:

- The origin is in the upper left corner
- The *width* is the total number of pixels in the x direction
- The *height* is the total number of pixels in the y direction

Pixel Coordinates - Rows and Columns

Unfortunately, there is not a unified standard across programming languages for coordinate order, (x, y) or (y, x)

The convention of any one particular language or library is usually set by whether arrays are viewed as *row-major* or *column-major*.

For the purposes of this course, note:

- Mathematical notation will follow the standard convention of (x, y)

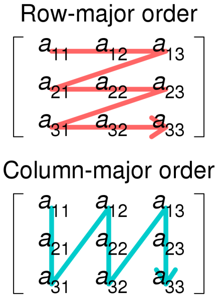


Image source: Wikipedia

Pixel Coordinates - Rows and Columns

Unfortunately, there is not a unified standard across programming languages for coordinate order, (x, y) or (y, x)

The convention of any one particular language or library is usually set by whether arrays are viewed as *row-major* or *column-major*.

For the purposes of this course, note:

- Mathematical notation will follow the standard convention of (x, y)
- OpenCV indexes images as $[y, x]$

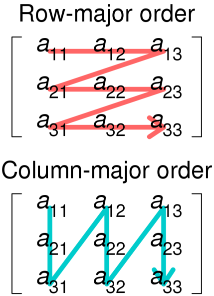


Image source: Wikipedia

Pixel Coordinates - Rows and Columns

Unfortunately, there is not a unified standard across programming languages for coordinate order, (x, y) or (y, x)

The convention of any one particular language or library is usually set by whether arrays are viewed as *row-major* or *column-major*.

For the purposes of this course, note:

- Mathematical notation will follow the standard convention of (x, y)
- OpenCV indexes images as $[y, x]$
- Pixel coordinates in OpenCV start at $[0, 0]$ and run to $[height - 1, width - 1]$

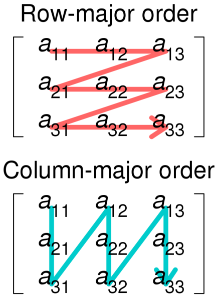


Image source: Wikipedia

Pixel Value

The value a pixel has represents some sort of information. Possible image types and data formats include:

An intensity image.

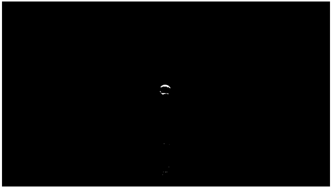
$\forall p, p_i = s_i$ where $s \in L$.

- $L = [0, 255]$, 8-bit integer image
- $L = [0, 1.0]$, floating point image

Pixel Value

The value a pixel has represents some sort of information. Possible image types and data formats include:

A binary image
(often referred to as a "mask").
$\forall p, p \in 0, 1$



All pixels from the previous image with value over 250

Pixel Value

The value a pixel has represents some sort of information. Possible image types and data formats include:

A colour image.

$\forall p, p_i = [a_i, b_i, c_i]$
where $a_i \in L_a, b_i \in L_b, c_i \in L_c$
and $[a_i, b_i, c_i]$ correspond to specific colour channels.

Colour Representation

The information carried in channels a, b, c and range available to each (L_a, L_b, L_c) differs depending on what model and standard is being used to represent colour information.

- The most common is 8-bit [Red, Green, Blue], with $L_R = L_G = L_B = [0, 255]$

Colour Representation

The information carried in channels a, b, c and range available to each (L_a, L_b, L_c) differs depending on what model and standard is being used to represent colour information.

- The most common is 8-bit [Red, Green, Blue], with $L_R = L_G = L_B = [0, 255]$
- Minor variants of RGB format include BGR or floating-point representation $(I = [0, 1.0])$

Colour Representation

The information carried in channels a, b, c and range available to each (L_a, L_b, L_c) differs depending on what model and standard is being used to represent colour information.

- The most common is 8-bit [Red, Green, Blue], with $L_R = L_G = L_B = [0, 255]$
- Minor variants of RGB format include BGR or floating-point representation $(I = [0, 1.0])$
- CIEL*a*b* (sometimes referred to simply as "Lab" or "LAB")

Colour Representation

The information carried in channels a, b, c and range available to each (L_a, L_b, L_c) differs depending on what model and standard is being used to represent colour information.

- The most common is 8-bit [Red, Green, Blue], with $L_R = L_G = L_B = [0, 255]$
- Minor variants of RGB format include BGR or floating-point representation $(I = [0, 1.0])$
- CIEL*a*b* (sometimes referred to simply as “Lab” or “LAB”)
  - L* is the lightness value, and has range $[0, 100]$

Colour Representation

The information carried in channels a, b, c and range available to each (L_a, L_b, L_c) differs depending on what model and standard is being used to represent colour information.

- The most common is 8-bit [Red, Green, Blue], with $L_R = L_G = L_B = [0, 255]$
- Minor variants of RGB format include BGR or floating-point representation $(I = [0, 1.0])$
- CIEL*a*b* (sometimes referred to simply as "Lab" or "LAB")
  - L* is the lightness value, and has range $[0, 100]$
  - a* is the green-red axis, range differs by implementation (*e.g.* $[-100, 100]$ or $[-128, 127]$)

Colour Representation

The information carried in channels a, b, c and range available to each (L_a, L_b, L_c) differs depending on what model and standard is being used to represent colour information.

- The most common is 8-bit [Red, Green, Blue], with $L_R = L_G = L_B = [0, 255]$
- Minor variants of RGB format include BGR or floating-point representation $(I = [0, 1.0])$
- CIEL*a*b* (sometimes referred to simply as "Lab" or "LAB")
  - L* is the lightness value, and has range $[0, 100]$
  - a* is the green-red axis, range differs by implementation (*e.g.* $[-100, 100]$ or $[-128, 127]$)
  - b* is the blue-yellow axis, has the same range as a*

Colour Representation

The information carried in channels a, b, c and range available to each (L_a, L_b, L_c) differs depending on what model and standard is being used to represent colour information.

- The most common is 8-bit [Red, Green, Blue], with $L_R = L_G = L_B = [0, 255]$
- Minor variants of RGB format include BGR or floating-point representation $(I = [0, 1.0])$
- CIEL*a*b* (sometimes referred to simply as "Lab" or "LAB")
  - L* is the lightness value, and has range $[0, 100]$
  - a* is the green-red axis, range differs by implementation (*e.g.* $[-100, 100]$ or $[-128, 127]$)
  - b* is the blue-yellow axis, has the same range as a*
  - Designed for perceptual-numerical equivalence

Colour Representation

The information carried in channels a, b, c and range available to each (L_a, L_b, L_c) differs depending on what model and standard is being used to represent colour information.

- The most common is 8-bit [Red, Green, Blue], with $L_R = L_G = L_B = [0, 255]$
- Minor variants of RGB format include BGR or floating-point representation $(I = [0, 1.0])$
- CIEL*a*b* (sometimes referred to as simply as “Lab” or “LAB”)
  - L* is the lightness value, and has range $[0, 100]$
  - a* is the green-red axis, range differs by implementation (*e.g.* $[-100, 100]$ or $[-128, 127]$)
  - b* is the blue-yellow axis, has the same range as a*
  - Designed for perceptual-numerical equivalence
- YCbCr is similar to LAB, except the design is more practical and less based on perceptual properties.

Colour Conversions and Grayscale

Converting from one colour space to another may or may not be reversible, depending on whether both colour spaces have equivalent representational power.

A common (irreversible) conversion is converting from colour to grayscale. Note that there are many different ways to do this!

- GIMP default uses the equation $L = 0.21R + 0.72G + 0.07B$

Colour Conversions and Grayscale

Converting from one colour space to another may or may not be reversible, depending on whether both colour spaces have equivalent representational power.

A common (irreversible) conversion is converting from colour to grayscale. Note that there are many different ways to do this!

- GIMP default uses the equation $L = 0.21R + 0.72G + 0.07B$
- MATLAB `rgb2gray` and OpenCV both use the equation $L = 0.299R + 0.587G + 0.114B$

Visualizing Pixel Values

It is often convenient to display or process pixel channels independently, treating each one as a grayscale image, or to treat non-luminance information as an intensity image.

Colour

Red Channel

Green Channel

Blue Channel

Point Operator: Definition

A *point operation* is a an image transformation in which the output value of a given pixel is determined solely as a function of the input pixel value corresponding to that location (or values, if more than one input image is used, and potentially subject to global modulation).

Point Operator: Definition

A *point operation* is a an image transformation in which the output value of a given pixel is determined solely as a function of the input pixel value corresponding to that location (or values, if more than one input image is used, and potentially subject to global modulation).

Converting a colour image to grayscale is an example of a point operation.

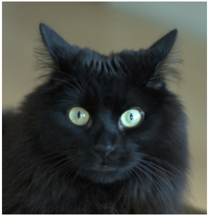
Brightness Adjustment

One example operation is the adjustment of global image brightness. The brightness adjustment operation is given by:

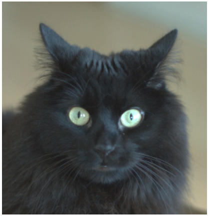
$$p'_i = p_i + b$$

where b is a constant value.

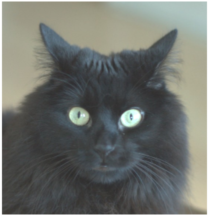
Brightness Adjustment Example



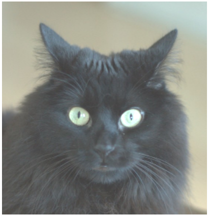
Original Image



$b = 20$

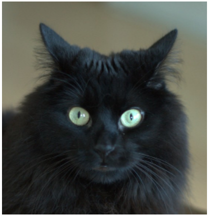


$b = 40$

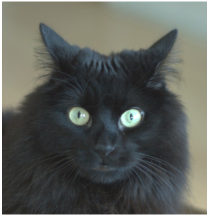


$b = 60$

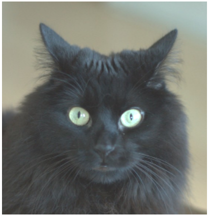
Brightness Adjustment Example



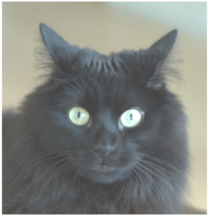
Original Image



$b = 20$



$b = 40$



$b = 60$

For dark images and reasonable values of b this can improve the perceived quality of the image, but too large of a value can lead to over saturation or "bleaching".

Normalization

When dealing with issues of saturation, we can re-map a range of values which is outside our available range onto our available range.

Normalization

When dealing with issues of saturation, we can re-map a range of values which is outside our available range onto our available range.

Let L_{max} be the maximum allowable value and L_{min} be the minimum allowable value for a given image representation.

Normalization

When dealing with issues of saturation, we can re-map a range of values which is outside our available range onto our available range.

Let L_{max} be the maximum allowable value and L_{min} be the minimum allowable value for a given image representation.

To remap a set of pixel values, we first need to compute the current global maximum, p_{max}, and minimum, p_{min}.

Normalization

When dealing with issues of saturation, we can re-map a range of values which is outside our available range onto our available range.

Let L_{max} be the maximum allowable value and L_{min} be the minimum allowable value for a given image representation.

To remap a set of pixel values, we first need to compute the current global maximum, p_{max}, and minimum, p_{min}.

For each pixel in the image, we can compute its remapped value according to the equation:

$$p' = (p - p_{min})\frac{L_{max}-L_{min}}{p_{max}-p_{min}} + L_{min}$$

Contrast Enhancement

Contrast refers to the difference in nearby pixel values. Multiplication is one way of manipulating contrast, because a multiplicative factor will increase the relative difference between pixels by increasing the value of bright pixels more than dark pixels. However, it is very easy to saturate values with multiplication, so it is often useful to re-normalize after applying our multiplicative gain.

Original Image

No normalization

Normalized

Alpha Compositing

Point operations may also be used to combine information from multiple images. One convenient tool for such a process is the α-matte.

Alpha Compositing

Point operations may also be used to combine information from multiple images. One convenient tool for such a process is the α-matte.

To perform alpha compositing, we associate an additional attribute to our pixels, α, which tells us how much of that pixel's value we want to use. Colour image formats which include α will sometimes be written as RGBα, or will be referred to as including a "transparency channel".

Alpha Compositing

Given two images, I_a and I_b, with pixel formats $[r, g, b, \alpha]$, we can compute an output image I_o according to the equation:

$$p = [r_o, g_o, b_o] = \alpha_a[r_a, g_a, b_a] + \alpha_b[r_b, g_b, b_b]$$

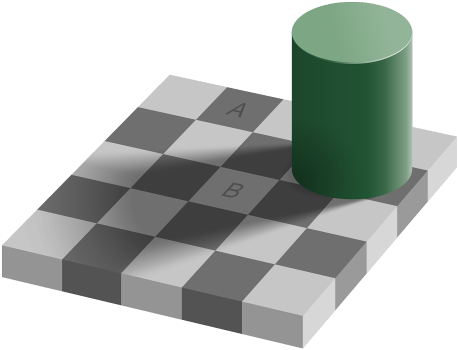
for all p in I_o.

Alpha Compositing

Demo.

Image Context

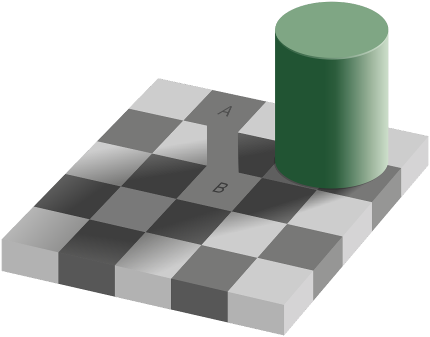
Sometimes it is not enough to look at pixels in isolation, and we instead need to integrate information from the local surroundings in order to interpret its value.



Point A and B are the same shade of grey.
Image Source: Adelson, 1995

Image Context

Sometimes it is not enough to look at pixels in isolation, and we instead need to integrate information from the local surroundings in order to interpret its value.



Linking the squares shows that they are the same shade.
Image Source: Adelson, 1995

Image Context

Sometimes it is not enough to look at pixels in isolation, and we instead need to integrate information from the local surroundings in order to interpret its value.

Removing the surrounding context makes it even more clear.
Image Source: Adelson, 1995

Local Neighbourhoods

- It is common in computer vision to take a local area around a pixel (the pixel's *neighbourhood*) and combine that information in some way

Local Neighbourhoods

- It is common in computer vision to take a local area around a pixel (the pixel's *neighbourhood*) and combine that information in some way
- If this combination takes the form of a weighted sum, then we refer to this as a *linear* operation

Local Neighbourhoods

- It is common in computer vision to take a local area around a pixel (the pixel's *neighbourhood*) and combine that information in some way
- If this combination takes the form of a weighted sum, then we refer to this as a *linear* operation
- Applying an operation like this is frequently referred to as *filtering* an image

Local Neighbourhoods

- It is common in computer vision to take a local area around a pixel (the pixel's *neighbourhood*) and combine that information in some way
- If this combination takes the form of a weighted sum, then we refer to this as a *linear* operation
- Applying an operation like this is frequently referred to as *filtering* an image
- The specific size and weights of a given operation are usually referred to as a *kernel*

Kernel Operations

- Assume we have an intensity image. Let $p_{x,y}$ be the value of the pixel with coordinates (x, y).

Kernel Operations

- Assume we have an intensity image. Let $p_{x,y}$ be the value of the pixel with coordinates (x, y).
- We will define a kernel K to be a 2D matrix of weights $w_{i,j}$, where (i, j) are indices in K

Kernel Operations

- Assume we have an intensity image. Let $p_{x,y}$ be the value of the pixel with coordinates (x, y).
- We will define a kernel K to be a 2D matrix of weights $w_{i,j}$, where (i, j) are indices in K

To apply this filter to our image, we compute the following equation:

$$p'_{x,y} = \sum_{i,j} w_{i,j} p_{x+i,y+j}$$

Worked Example

$$\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\1 & 1 & 1\\1 & 1 & 1\end{bmatrix}$$

Our kernel

18	27	54	9	81
45	27	72	36	63
63	63	27	54	36
45	72	27	36	27
9	81	45	54	54
27	63	54	36	27

Our image patch

Image Borders

What do we do for edge pixels?

Image Borders

What do we do for edge pixels?

The most common numerical method is *padding*, where you add extra pixels to the outer edge. But what do we pad with?

Image Borders

What do we do for edge pixels?

The most common numerical method is *padding*, where you add extra pixels to the outer edge. But what do we pad with?

- Zero-padding

Image Borders

What do we do for edge pixels?

The most common numerical method is *padding*, where you add extra pixels to the outer edge. But what do we pad with?

- Zero-padding
- Wrap

Image Borders

What do we do for edge pixels?

The most common numerical method is *padding*, where you add extra pixels to the outer edge. But what do we pad with?

- Zero-padding
- Wrap
- Clamp (aka replicate)

Image Borders

What do we do for edge pixels?

The most common numerical method is *padding*, where you add extra pixels to the outer edge. But what do we pad with?

- Zero-padding
- Wrap
- Clamp (aka replicate)
- Mirror

Image Borders

What do we do for edge pixels?

The most common numerical method is *padding*, where you add extra pixels to the outer edge. But what do we pad with?

- Zero-padding
- Wrap
- Clamp (aka replicate)
- Mirror
- Plus other more sophisticated heuristics

Image Padding Examples

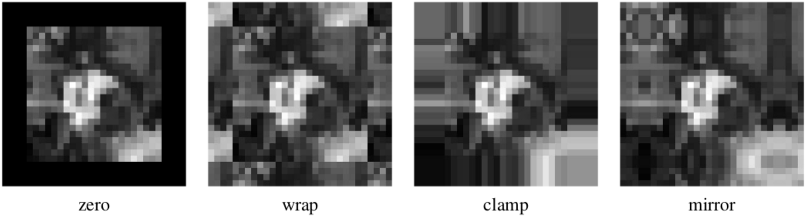


Image Source: Szeliski, 2011

Some Common Kernels

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Some Common Kernels

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Sharpen kernel

Some Common Kernels

$$\frac{1}{16}\begin{bmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{bmatrix}$$

Some Common Kernels

$$\frac{1}{16}\begin{bmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{bmatrix}$$

Gaussian blur (approximate)