ENGG1003 - Thursday Week 10

Assignment 2: Image processing

Steve Weller

University of Newcastle

13 May 2021

Last compiled: May 13, 2021 5:05pm +10:00

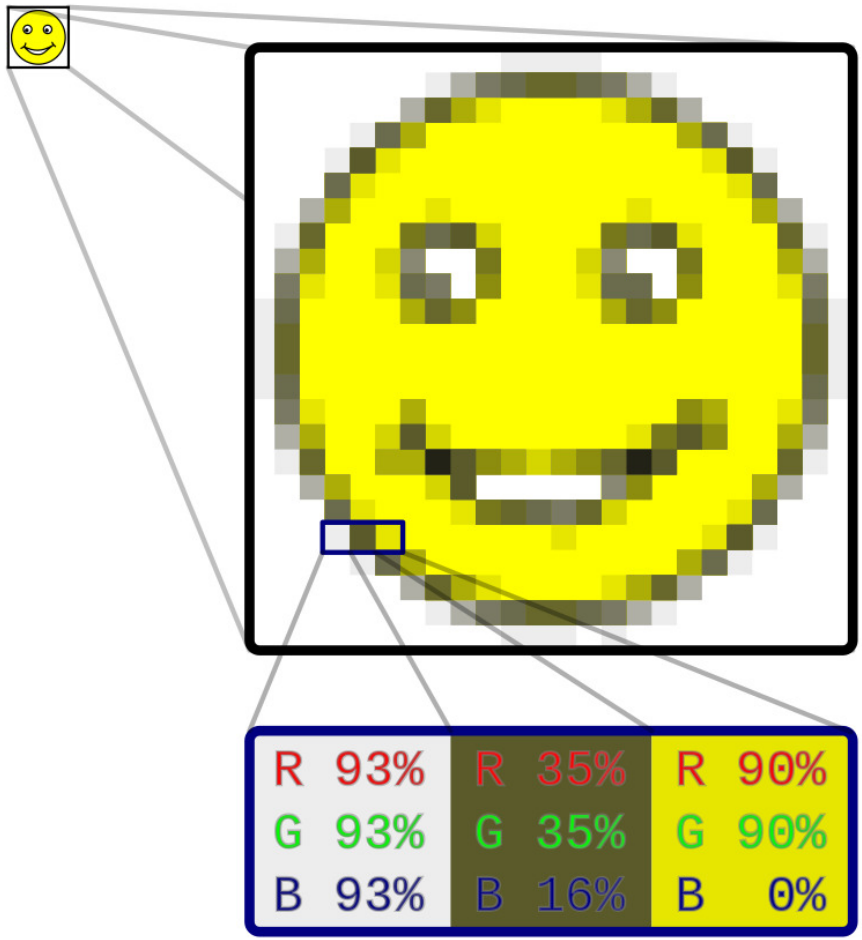
Lecture overview

1. key assignment information
2. images as 3D arrays
3. digital image formats
4. data types
5. structure of assignment
6. strategies for the assignment

1) Key assignment information

- released: Monday 10 May 2021
- **due date: 9:00am Monday 31 May 2021**
  - Monday of week 13
- weighting: 15% of final course grade
- assignment sheet in BB > assessment
  - ensure you always have latest version
- submission: upload `imageProcessing.py` file to BB as an assignment submission
- marking: during face-face labs in week 13
  - final exam is on Tuesday of "week 14" (8 June)
- marking guide released earlier today; details page 15 this lecture

2) Images as 3D arrays



https://en.wikipedia.org/wiki/Raster_graphics

CC0 1.0

Colour image as a 3D array

- think of 3D array as a 2D array where each entry is itself a length-3 array specifying the colour in RGB (red, green, blue) format
  - each [R,G,B] entry maps to one pixel

$$\begin{bmatrix} [R,\ G,\ B] & \cdots & [R,\ G,\ B] \\ [R,\ G,\ B] & \cdots & [R,\ G,\ B] \\ [R,\ G,\ B] & \cdots & [R,\ G,\ B] \end{bmatrix}$$

Examples:

red: [255, 0 ,0] green: [0, 255, 0]
blue: [0, 0, 255] purple: [65, 0, 125]

3) Digital image formats

- colourspaces
  - RGB: red–green–blue
  - HSL: hue–saturation–luminance
- RGB and HSL are two different ways of representing the *same* colour
  - key theme of assignment: RGB $\longleftrightarrow$ HSL
- intensity values stored as numbers with min/max values:
  - floats in range $[0, 1]$
  - integers in range $[0, 255]$
- https://www.w3schools.com/colors/colors_picker.asp

4) Data types

Assignment makes heavy use of numpy datatypes:

- `uint8`
  - integer 0 to 255
- `uint16`
  - integer 0 to 65535
- `float32`
  - single-precision float (occupies 32 bits)
- `float64`
  - double-precision float (occupies 64 bits)

5) Structure of assignment

- covers the basics of digital image manipulation
- you will learn how everyday tools such as mobile phone camera apps perform several common image processing tasks
- first 5 functions
  - "unconstrained" help from discord, demonstrators, fellow students is permitted
    (your assignment submission must be your own work)
  - no marks for these questions; required for later q's
- next 8 functions
  - where the marks are
  - **functions can be attempted in any order!**
  - implement all 8 functions, or fewer (for fewer marks)

Five "getting started" functions

- **`loadImage()`**
  - read image file into 3D numpy array
- **`saveImage()`**
  - save 3D numpy array as image file
- **`rgb2hsl()`**
  - convert image in RGB format to HSL format
- **`hsl2rgb()`**
  - convert image in HSL format to RGB format
- **`showImage()`**
  - display image in window

Eight functions in the assignment

Eight (8) functions to be graded in assignment

- **`brightness()`**
  - adjust image brightness
- **`contrast()`**
  - adjust image contrast
- **`saturation()`**
  - adjust image saturation
- **`toneMap()`**
  - adjust image by setting H and S channels of each pixel

Eight (8) functions to be graded in assignment (ctd.)

- **`crop()`**
  - crop image
- **`histogram()`**
  - plot histogram of image
- **`saturated()`**
  - compute percentage of pixels which have at least one RGB channel value which has undergone clipping saturation
- **`unsharpMask()`**
  - sharpen image

6) Strategies for the assignment

- lots of useful and relevant info in the assignment sheet
- *strongly* recommend completion of week 10 lab sheet before starting assignment
- start small, take tiny steps
- test RGB/HSL conversion against colour picker

Strategies

- submission to BB will be a single file `imageProcessing.py`
  - your uploaded file will contain definitions and code for 5+8 functions
  - implement < 8 assessable functions, for < 15 marks

- **strongly** encouraged to develop and test as follows:

  1. each function's behaviour in its own script (test it)
  2. define code into function in same file (test it again)
  3. copy/paste working function into `imageProcessing.py` (and test it again!)
     - test code will be made available to students
     - you can check *in advance* if your code works correctly!

Strategies for developing code

Step 1: in `square.py`

```
1 # square
2
3 x = 3
4 print('{} squared = {:.4f}'.format(x,x**2))
```

Strategies for developing code

Step 1: in `square.py`

```
1 # square
2
3 x = 3
4 print('{} squared = {:.4f}'.format(x,x**2))
```

Step 2: in `square_fn.py`

```
1 # square_fn
2 def f(x):
3     return x**2
4
5 x = 3
6 print('{} squared = {:.4f}'.format(x,f(x)))
```

Strategies for developing code

Step 1: in `square.py`

```
1 # square
2
3 x = 3
4 print('{} squared = {:.4f}'.format(x,x**2))
```

Step 2: in `square_fn.py`

```
1 # square_fn
2 def f(x):
3     return x**2
4
5 x = 3
6 print('{} squared = {:.4f}'.format(x,f(x)))
```

Step 3: in `imageProcessing.py`

```
1 def f(x):
2     return x**2
```

Code testing and grading

- 15 marks total, for 15% assignment
- 2 marks each for correct implementation of:
  - brightness()
  - contrast()
  - saturation()
  - toneMap()
  - crop()
  - saturated()
  - histogram()
- 1 mark for correct implementation of:
  - unsharpMask()

Python test script published on BB which allows you to judge the correctness of your implementations with easy-to-debug data