ENGG1003 - Thursday Week 10

Assignment 2: Image processing

Steve Weller

University of Newcastle

13 May 2021

Last compiled: May 13, 2021 1:31pm +10:00

Lecture overview

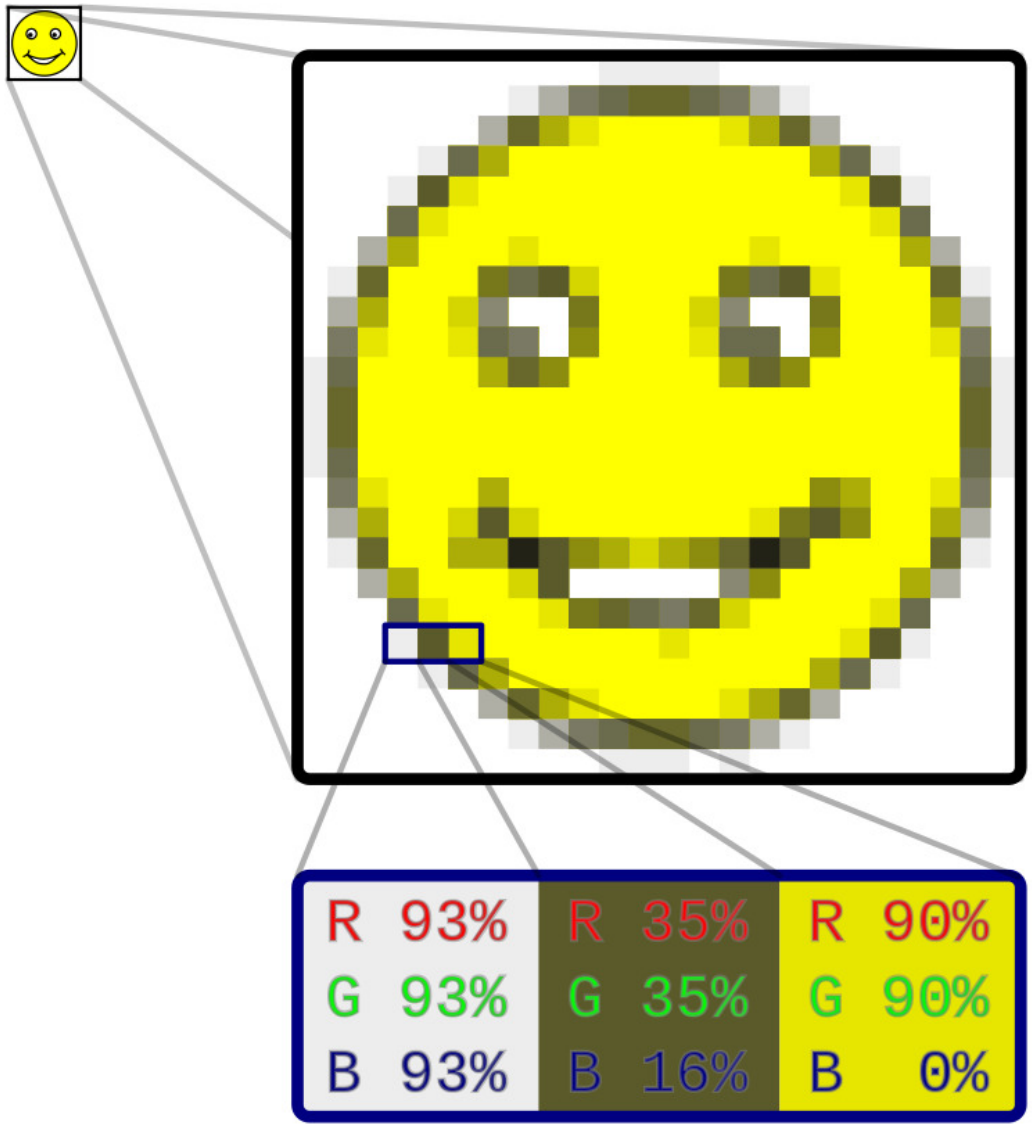
1. key assignment information
2. images as 3D arrays
3. digital image formats
4. data types
5. structure of assignment
6. strategies for the assignment

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 8 of "week 14" (8 June)
- marking guide released earlier today; details later in this lecture

2) Images as 3D arrays

- raster images
  - gif, jpeg, png
  - contrast with vector images: svg
- review of Sarah's material from Thursday week 7



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

3) Digital image formats

- colourspaces
  - RGB
  - HSL
- RGB and HSL are two different ways of representing the *same* colour
  - key theme of assignment: RGB <--> HSL
- $[0, 1]$ and $[0, 255]$
- use colour images and links to colour picker

4) Data types

- uint8
- uint16
- float32
- float64

- type conversions

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
  - "unlimited" 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

- rtfm
- recommend (assume) Lab sheet week 10 first
- 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 up to eight (8) functions
  - you may implement < 8 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
```

Getting started with the assignment

- XXX