SciComp with Py

CVIP

Basic Image Processing with OpenCV

Part 02

Vladimir Kulyukin
Department of Computer Science
Utah State University

Review

OpenCV

- OpenCV (opencv.org) is a open source CV library
- Free for both commercial and academic use under a BSD license
- OpenCV has C++, Java, & Python interfaces
- Available on Linux, Mac OS, Android, iOS, Windows

Image Formats

- Image formats are different standards of storing digital images
- Broadly speaking, there are three kinds of data storage: compressed, uncompressed, and vector
- There are two types of compression: lossless and lossy
- Common formats: JPEG (compressed, lossy), BMP (uncompressed, lossless), PNG (compressed, lossless), SVG (scalable vector graphics)
- OpenCV supports all common formats

Checking OpenCV Installation

```
$ python
Python 2.7.6 (default, Jun 22 2015, 17:58:13)
[GCC 4.8.2] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import cv2
>>> cv2.__version__
'3.0.0'
>>>
```

Problem

Write a program that loads a user specified image, displays it in a window, and waits for the user to press a key before closing the window.

Sample Call

```
$ python   load_image.py  -i truck.jpg
```

Loading Images

load_image.py

```
import argparse
import cv2

ap = argparse.ArgumentParser()
ap.add_argument('-i', '--image', required = True, help = 'Path to image')
args = vars(ap.parse_args())

image = cv2.imread(args['image'])

cv2.imshow('Image', image)
cv2.waitKey(0)
```

Parse user args

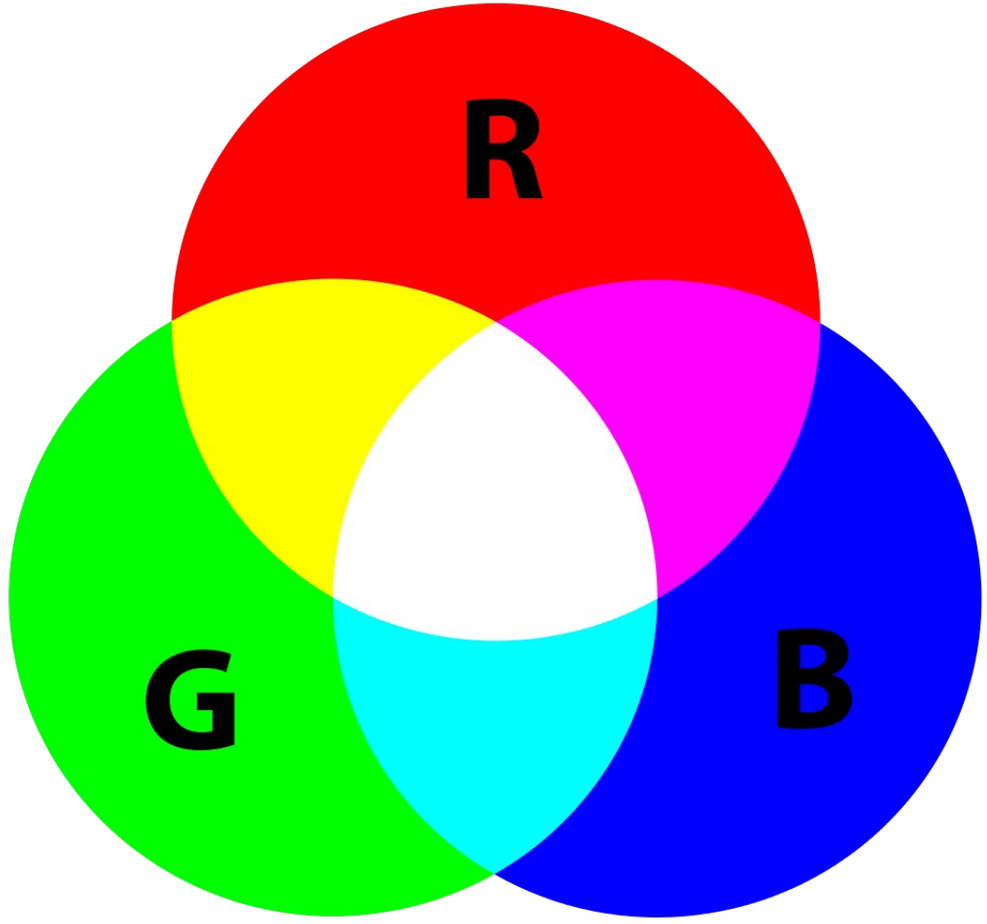
Load image from user-specified file

Show Image

Wait for user to press a key

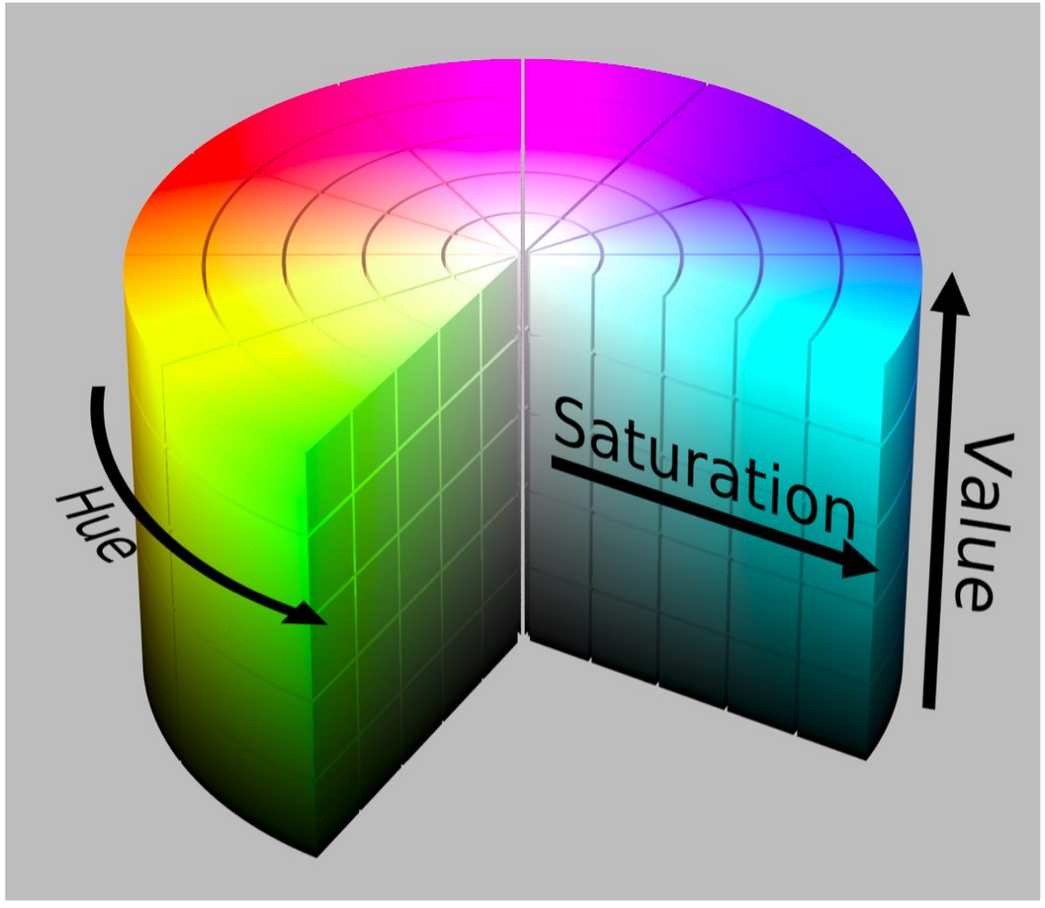
RGB and HSV: Two Basic Color Spaces

RGB Color Space



In OpenCV, the standard pixel representation is B, G, R, e.g., [10, 234, 50]. The values of B, G, R are in [0, 255]. [0, 0, 0] is black; [255, 255, 255] is white.

HSV Color Space



- Hue is color value [0, 179]
- Saturation is color vibrancy [0, 255]; at lower saturation in the center everything is white
- Value is brightness/intensity of color [0, 255]; it goes from dark (below) to bright (above)

RGB & HSV Spaces

It is possible to map an RGB point to a HSV point and vice versa; most image processing applications have this option

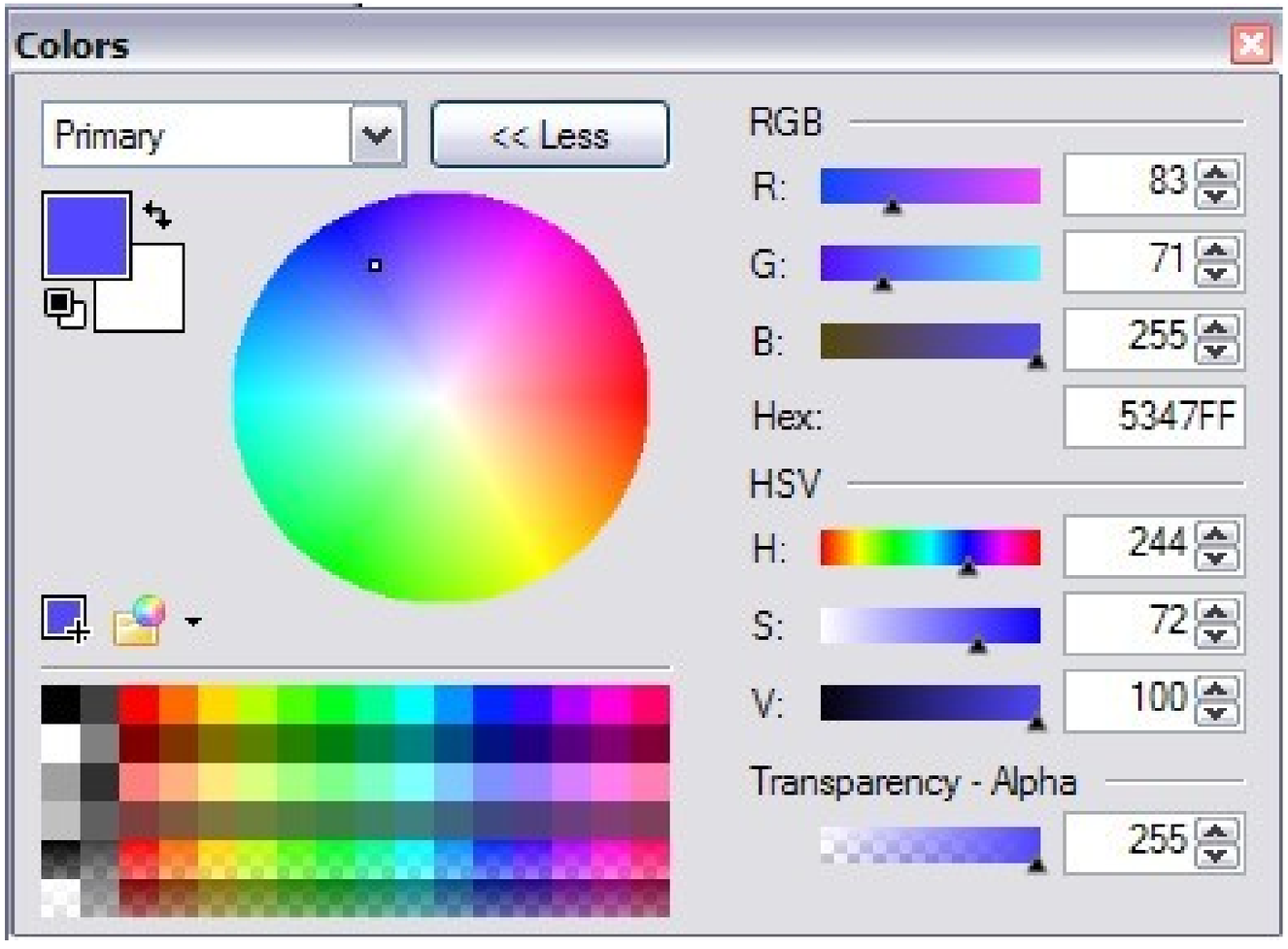


Image Shapes

Problem

Write a program that loads a user specified image, prints out the shape of the image (height, width, number of channels), displays the image in a window, and waits for the user to press a key before closing the window.

Sample Call

```
$ python   image_shape.py  -i truck.jpg
```

Getting Image's Shape (Height, Width, Num of Channels)

image_shape.py

```
import argparse
import cv2

ap = argparse.ArgumentParser()
ap.add_argument('-i', '--image', required = True, help = 'Path to image')
args = vars(ap.parse_args())

image = cv2.imread(args['image'])
(h, w, num_channels) = image.shape
print 'h=' + str(h) + '; ' + 'w=' + str(w) + '; ' + 'c=' + str(num_channels)

cv2.imshow('Loaded Image', image)
cv2.waitKey(0)
```

Load image from user-specified file

Get & print image's shape

Show image

Image Grayscaling

Grayscaling

Grayscaling is an operation of converting RGB pixels to grayscale intensity pixels. Here is a commonly used conversion formula:

```
def luminosity(rgb, rcoeff=0.2126, gcoeff=0.7152, bcoeff=0.0722):
    return rcoeff*rgb[0]+gcoeff*rgb[1]+bcoeff*rgb[2]
```

Problem

Write a program that loads a user specified image, converts it into grayscale, displays the original image and the grayscale image in two windows, and waits for the user to press any key before closing the windows.

Sample Call

```
$ python   grayscale_image.py  -i truck.jpg
```

Grayscaling

grayscale_image.py

```
import cv2
import argparse

ap = argparse.ArgumentParser()
ap.add_argument('-i', '--image', required = True, help = 'Path to image')
args = vars(ap.parse_args())

image = cv2.imread(args['image'])

cv2.imshow('Original', image)
cv2.waitKey()

gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
cv2.imshow('Grayscaled', gray_image)
cv2.waitKey()

cv2.destroyAllWindows()
```

Grayscale image

Grayscaling

grayscale_image2.py

```
import cv2
import argparse

ap = argparse.ArgumentParser()
ap.add_argument('-i', '--image', required = True, help = 'Path to image')
args = vars(ap.parse_args())

image = cv2.imread(args['image'], 0)
cv2.imshow('Grayscaled', image)
cv2.waitKey()

cv2.destroyAllWindows()
```

Grayscale image: the image is grayscaled, because the 2nd parameter is 0.

References

- www.opencv.org
- www.python.org