COMP 204

Introduction to image analysis with scikit-image (part one)

Yue Li
based on slides from Mathieu Blanchette, Christopher J.F. Cameron and Carlos G. Oliver

Image processing and analysis in Python

Goal: Process and analyze digital images.

- Very useful for processing microscopy images, medical imaging, etc.
- Closely linked with machine learning for image analysis

scikit-image module or (skimage)

- image processing module in Python
- holds a wide library of image processing algorithms: filters, transforms, point detection
- API
  - `http://scikit-image.org/docs/dev/api/api.html`

Our mascot for today

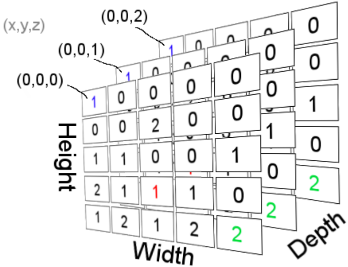
What's an image in Python?

An image is stored as a NumPy ndarray (n-dimensional array).

- ndarrays are easier and more efficient than using 2-dimensional lists as we've seen before.

A color image with R rows and C columns is

- represented as a 3-dimensional ndarray of dimensions $R \times C \times 3$
- element at position (i, j) of the array corresponds to the RGB value at row i and column j
- each pixel is represented by 3 numbers, each between 0 and 255: Red, Green, Blue



Reading an image into memory

We'll start with an example image using the **io module**

- basic I/O submodule of scikit-image
- API:

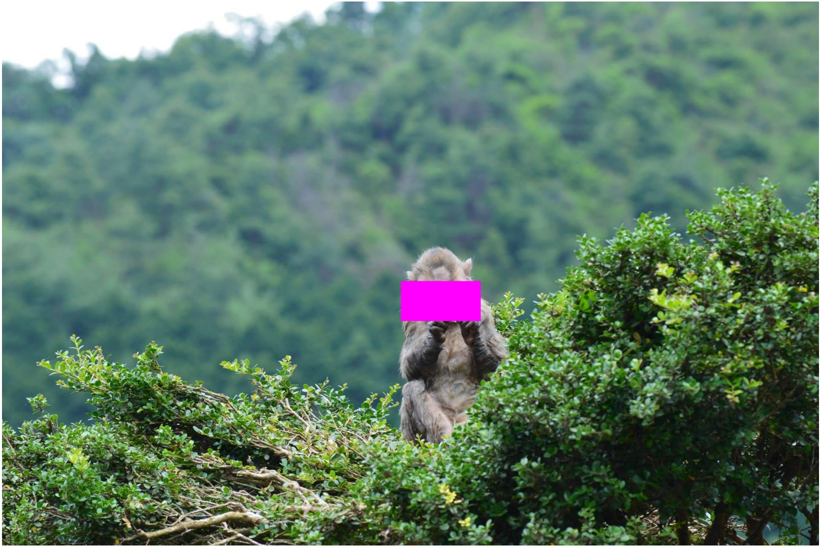
http://scikit-image.org/docs/dev/api/skimage.io.html

```python
import skimage.io as io
import matplotlib.pyplot as plt

# read image into memory
image = io.imread("monkey.jpg")

plt.imshow(image)
plt.show()

# print top-left pixel RGB values
print(image[0,0]) # a white pixel in the monkey image
# prints: [255 255 255]
# write image to disk
io.imsave("monkey_copy.jpg",image)
```

Playing with an image

```python
image = io.imread("monkey.jpg")

n_row, n_col, n_colours = image.shape
print(n_row, n_col, n_colours)  # prints (1362, 2048, 3)

# print pixel value at row 156 and column 293
pixel = image[156,292]
print(pixel)  # prints [112 158 147] = green-bluish color

# Create a pink rectangle between rows 700-800
# and column 1000-1200
for i in range(700,800):
    for j in range(1000,1200):
        image[i,j] = (255,0,255)

# this is equivalent to the following code
# i.e., set every element in the array to (255,0,255)
#image[700:800,1000:1200] = (255,0,255)

plt.imshow(image)
plt.show()
io.imsave("monkey_bar.jpg",image)
```

A face-masked image

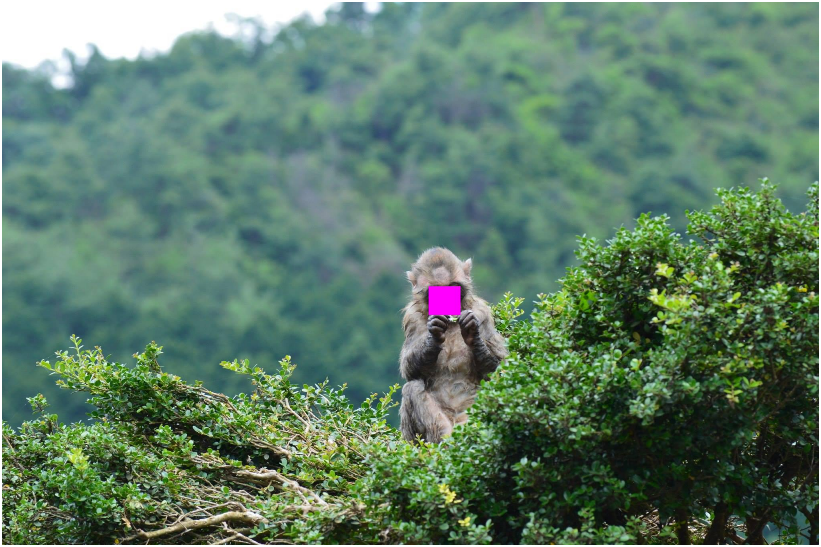


Let's try a more "refined" face mask by modifying the code

Code for a more refined mask

```python
import skimage.io as io
import matplotlib.pyplot as plt

# read image into memory
image = io.imread("monkey.jpg")

# find out where the monkey face is in the image
plt.imshow(image)
plt.show()

# a more "refined" face mask (713,785),(1070,1150)
image[713:785,1070:1150] = (255,0,255)

plt.imshow(image)
plt.show()
io.imsave("monkey_refined_mask.jpg",image)
```

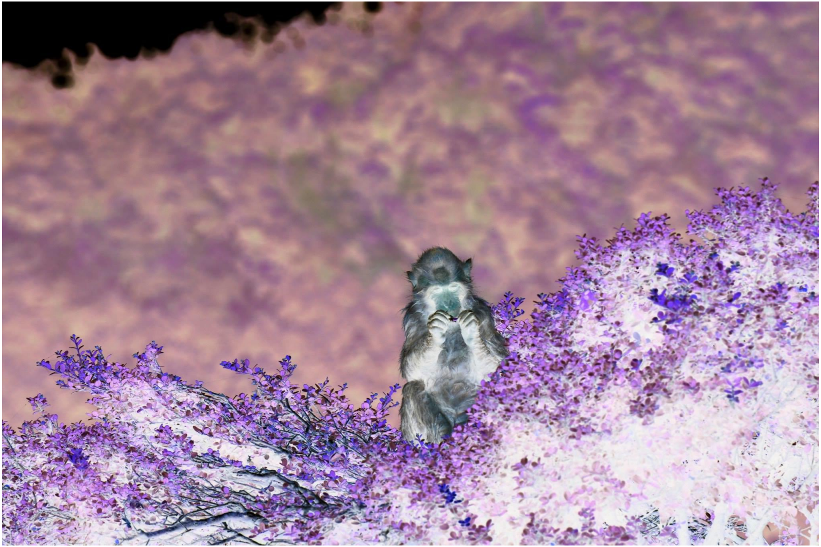
A more refined face-masked image



Creating the negative of an image

```python
import skimage.io as io
import skimage.color as color
import matplotlib.pyplot as plt

# read image into memory
image = io.imread("monkey.jpg")

# Create the negative of an image
for i in range(image.shape[0]):
    for j in range(image.shape[1]):
        for c in range(3):
            image[i,j,c] = 255-image[i,j,c]

# we could just have written:
#image = 255 - image

plt.imshow(image)
plt.show()
io.imsave("monkey_negative.jpg",image)
```

The “negative” image

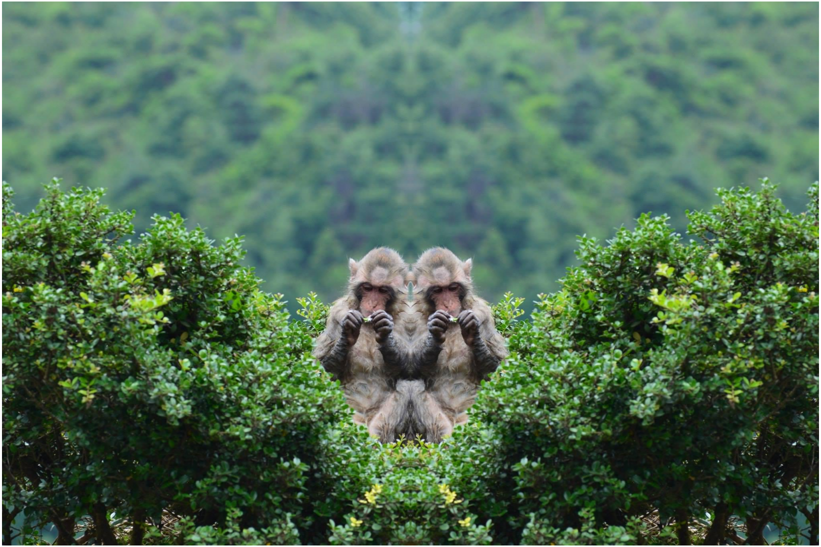


Flipping the image horizontally

There is a bug in this code. Can you find and fix it?

```python
import skimage.io as io
import skimage.color as color
import matplotlib.pyplot as plt

# read image into memory
image = io.imread("monkey.jpg")
n_row, n_col, colours = image.shape

# Flip the image horizontally
for i in range(0,n_row):
    for j in range(0,int(n_col/2)):
        image[i,j] = image[i, n_col-j-1]

plt.imshow(image)
plt.show()
io.imsave("monkey_flipped_wrong.jpg",image)
```

Incorrectly flipped image (a gemini monkey? just kidding)



Flipping the image horizontally (3 correct ways)

```python
import skimage.io as io
import skimage.color as color
import matplotlib.pyplot as plt
import numpy as np

# read image into memory
image = io.imread("monkey.jpg")
n_row, n_col, colours = image.shape

# Flip the image horizontally
for i in range(0,n_row):
    for j in range(0,int(n_col/2)):
        t = image[i,j].copy()
        image[i,j] = image[i, n_col-j-1]
        image[i, n_col-j-1] = t

# this is equivalent to:
#image = image[:,::-1]

# this is also equivalent to:
#image = np.flip(image, 1)

plt.imshow(image)
plt.show()
io.imsave("monkey_flipped_right.jpg",image)
```

Correctly flipped image

How to flip an image up side down?

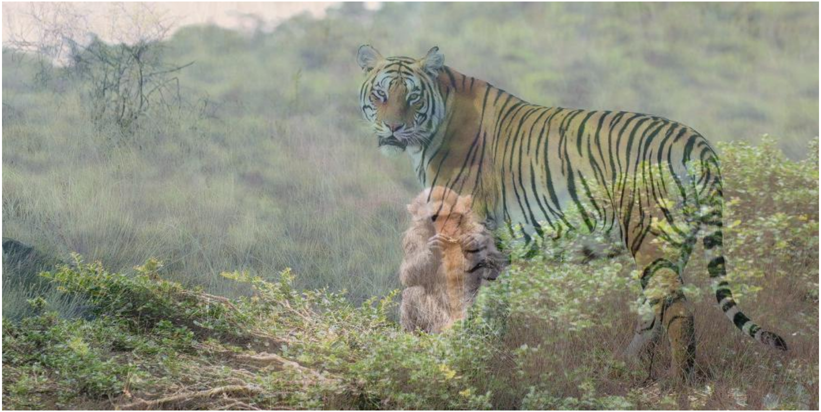
Combining images

Since images are just numerical arrays, we can easily combine them. Example:
Create an image that is the average of monkey and tiger.

Combining images

```python
import skimage.io as io
import matplotlib.pyplot as plt
import numpy as np
from skimage.transform import resize

monkey = io.imread("monkey.jpg")
tiger = io.imread("tiger.jpg")

#resize images to 500x1000 pixels
monkey_resized = resize(monkey, (500, 1000))
tiger_resized = resize(tiger, (500, 1000))

combined = np.zeros((500,1000,3))
for i in range(500):
    for j in range(1000):
        for c in range(3):
            combined[i,j,c]=monkey_resized[i,j,c]/2 + \
                tiger_resized[i,j,c]/2

# this is equivalent to:
#combined = monkey_resized/2 + tiger_resized/2

plt.imshow(combined)
plt.show()
io.imsave("combined.jpg",combined)
```

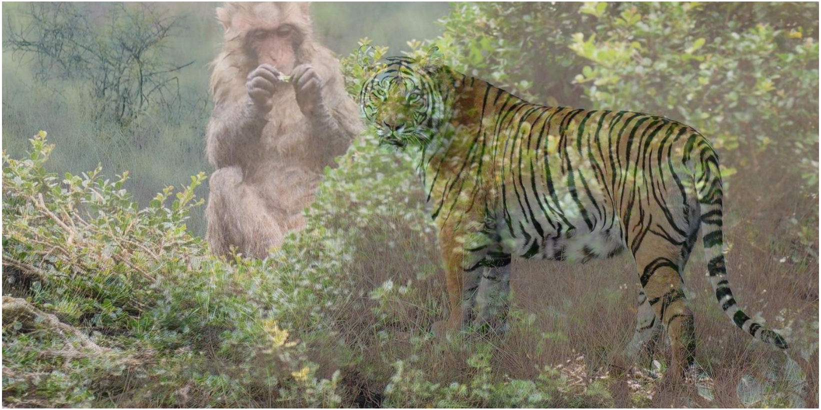
Combining images (monkey/2+tiger/2)



crop and combe images

```python
import skimage.io as io
import matplotlib.pyplot as plt
from skimage import img_as_float

monkey = io.imread("monkey.jpg")
tiger = io.imread("tiger.jpg")

print(monkey.shape) # (1362, 2048, 3)
print(tiger.shape) # (697, 1400, 3)

monkey_height, monkey_width, n_colours = monkey.shape
tiger_height, tiger_width, n_colours = tiger.shape

monkey_cropped = monkey[monkey_height-tiger_height:monkey_height,
        monkey_width-tiger_width:monkey_width]

print(monkey_cropped.shape) # (697, 1400, 3)
print(tiger.shape) # (697, 1400, 3)

combined = monkey_cropped/2 + tiger/2
combined = img_as_float(combined/255)

plt.imshow(combined)
plt.show()
io.imsave("cropped_and_combined.jpg",combined)
```

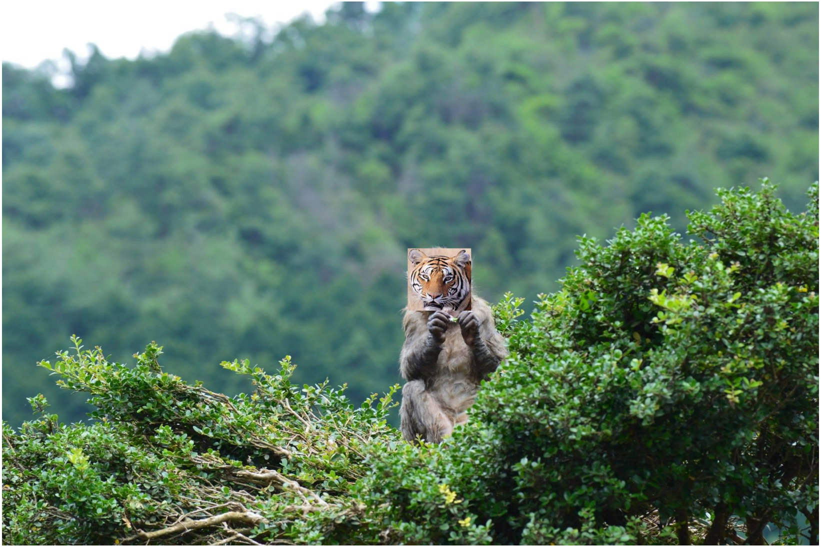
cropped and combined images



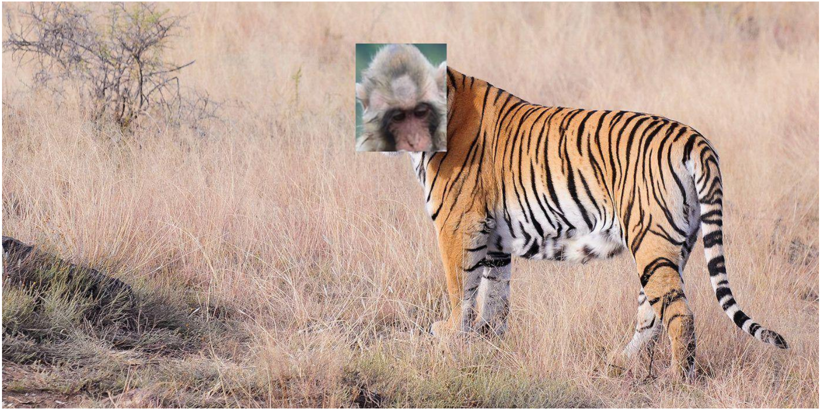
Switching patches between two images

```
#plt.imshow(monkey)
#plt.show()
print(monkey.shape)
monkey_head = monkey[618:774,1017:1176]
#plt.imshow(monkey_head)
#plt.show()

#plt.imshow(monkey)
#plt.show()
print(tiger.shape)
tiger_head = tiger[72:257,606:762]
#plt.imshow(tiger_head)
#plt.show()

print(monkey_head.shape)
print(tiger_head.shape)

tiger_head_resized = resize(tiger_head, (774-618,
↪  1176-1017))
monkey_head_resized = resize(monkey_head, (257-72,
↪  762-606))

monkey[618:774,1017:1176] = tiger_head_resized*255
tiger[72:257,606:762] = monkey_head_resized*255
```

A monkey with a tiger head

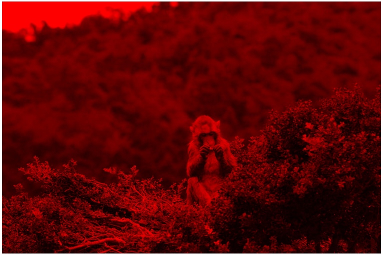


A tiger with a monkey head

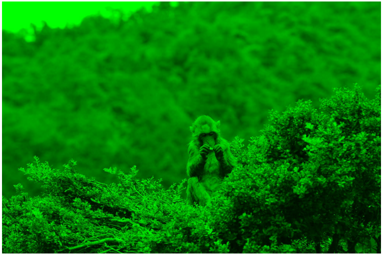


Color separation colors

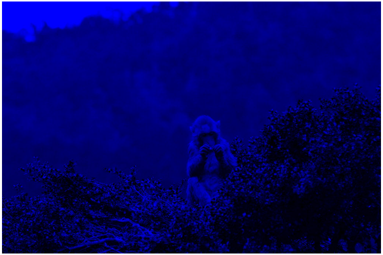
```python
import skimage.io as io

image = io.imread("monkey.jpg")

red, green, blue = image.copy(), image.copy(), image.copy()
red[:,:,(1,2)] = 0  # NumPy indexing
green[:,:,(0,2)] = 0
blue[:,:,(0,1)] = 0
io.imsave("monkey_red.jpg",red)
io.imsave("monkey_green.jpg",green)
io.imsave("monkey_blue.jpg",blue)
```



red intensity



green intensity

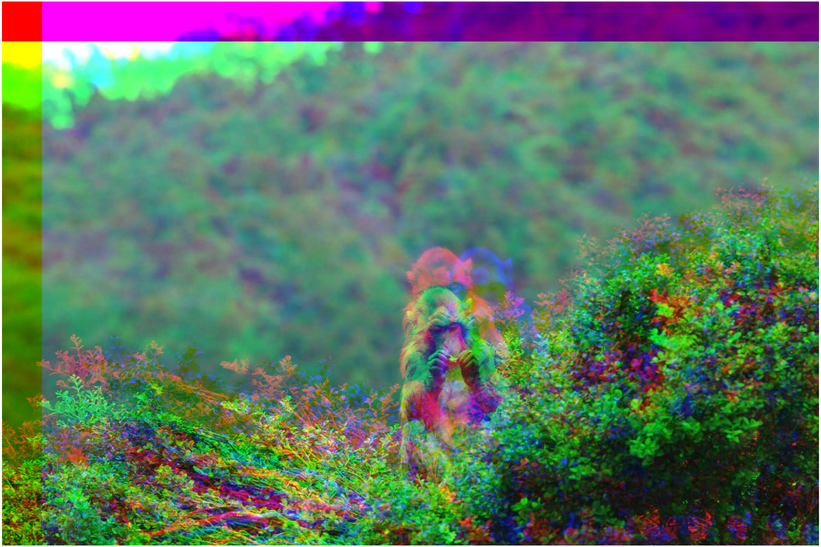


blue intensity

Shifting colors

```python
import skimage.io as io
import skimage.color as color
import matplotlib.pyplot as plt
import numpy as np

image = io.imread("monkey.jpg")
n_row, n_col, colors = image.shape

# create a blank image
new_image = np.zeros( (n_row, n_col, 3), dtype=np.uint8)

# assemble a new image made of shifted colors
# blue is shifted right by 100 pixels
# green is shifted up by 100 pixels
for i in range(n_row):
    for j in range(n_col):
        new_image[i,j,0] = image[i,j,0]  # keep red
        if i>=100:
            new_image[i,j,1]=image[i-100,j,1]  # move green
        if j>=100:
            new_image[i,j,2]=image[i,j-100,2]  # move blue

plt.imshow(new_image)
plt.show()
io.imsave("monkey_shifted.jpg",new_image)
```

Color shifted image



Grayscaling

Many image processing algorithms assume a 2D matrix

- not an image with a third dimension of color

To bring the image into two dimensions

- we need to summarize the three colors into a single value
- this process is more commonly know as **grayscaling**
- where the resulting image only holds intensities of gray
    - with values between 0 and 1

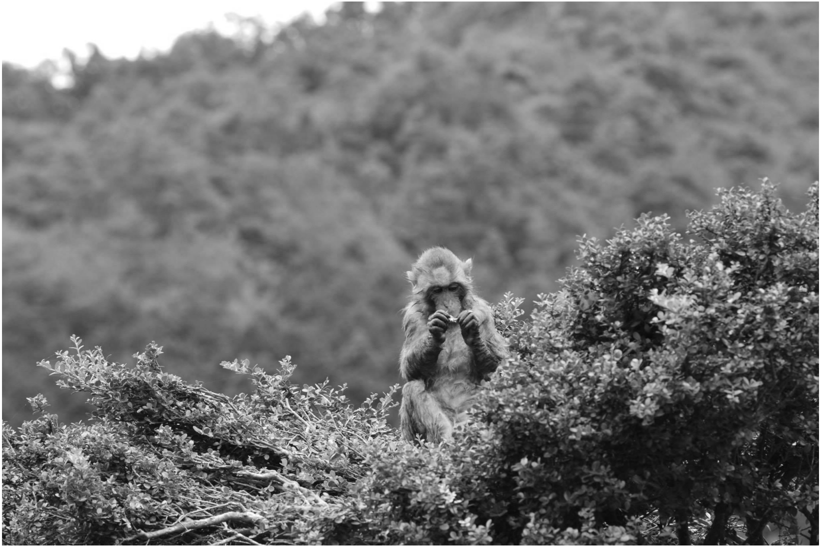
skimage submodule **color** has useful functions for this task

- API: `http://scikit-image.org/docs/dev/api/skimage.color.html`

Grayscaling images

```python
from skimage.color import rgb2gray
import skimage.io as io
import matplotlib.pyplot as plt

# read image into memory
image = io.imread("monkey.jpg")
# convert to grayscale
gray_image = rgb2gray(image)
plt.imshow(gray_image, cmap='gray')
plt.show()
io.imsave("monkey_grayscale.jpg",gray_image)

print(image[0,0]) # a white pixel in RGB
# prints: [255 255 255]
print(gray_image[0,0]) # a white pixel in grayscale
# prints: 1.0
```

Grayscale image



Binarizing images

```python
import skimage.io as io
import matplotlib.pyplot as plt
from skimage.color import rgb2gray
import numpy as np

image = io.imread("monkey.jpg")
gray_image = rgb2gray(image)
#print(gray_image[0,0])
# prints: 1.0
binary_image = np.where(gray_image > np.mean(gray_image),1.0,0.0)

plt.imshow(binary_image, cmap='gray')
plt.show()
io.imsave("monkey_binary.jpg",binary_image)
print(binary_image[0,0]) # a white pixel in the binary image
# prints: 1.0
```

`numpy.where(condition[, x, y])`: Return elements chosen from x or y depending on the condition.

A binary image