情報とコンピュータ

http://www.cs.miyazaki-u.ac.jp/~date/lectures/2013ic/

伊達 章

宮崎大学 工学部 情報システム工学科

2014 年 1 月 23 日

情報とコンピュータ

- 担当教員： 伊達 章，A-333
  date@cs.miyazaki-u.ac.jp
- 成績の評価方法： レポート

わけがわからない事，多すぎ！
かもしれないが，とにかく，さわって理解する

講義のスケジュール（案）

10/31 プログラミング環境の整備・インストール
Python, PyCharm, NumPy, SciPy, matplotlib

11/7 確率と情報
コンピュータで乱数を発生させてシミュレーション

1/16 パターン情報処理（音，画像，言語）

http://www.cs.miyazaki-u.ac.jp/~date/lectures/2013ic/comments20140116.html

1/23 パターン情報処理（音，画像，言語）

1/30 脳の仕組み：脳とコンピュータ
Deep Learning

- 提出課題に対する回答

パターン情報処理： 画像

- 画像を表示する.
- 「画像は数字の羅列」を確認
- 画像データを加工する

- 顔認識, 文字認識： なぜ区別できる？
- 顔認識技術の現在

「画像は数字の羅列」を確認

```
$ ipython
```

PIL（Python Imaging Library）がインストールされていない場合は... ちょっと待っててください

「画像は数字の羅列」を確認

```
$ ipython
In [1]:from PIL import Image
```

PIL（Python Imaging Library）がインストールされていない場合は... ちょっと待っててください

「画像は数字の羅列」を確認

```
$ ipython
In [1]:from PIL import Image
In [2]:im = Image.open('ics001.jpg')  # ファイル名
```

PIL（Python Imaging Library）がインストールされていない場合は... ちょっと待っててください

「画像は数字の羅列」を確認

```
$ ipython
In [1]:from PIL import Image
In [2]:im = Image.open('ics001.jpg')  # ファイル名
In [3]:im.show() # 表示
```

PIL（Python Imaging Library）がインストールされていない場合は... ちょっと待っててください

「画像は数字の羅列」を確認

```
$ ipython
In [1]:from PIL import Image
In [2]:im = Image.open('ics001.jpg')  # ファイル名
In [3]:im.show() # 表示
In [4]:x = im.size # 画像の大きさは？
```

PIL（Python Imaging Library）がインストールされていない場合は... ちょっと待っててください

「画像は数字の羅列」を確認

```
$ ipython
In [1]:from PIL import Image
In [2]:im = Image.open('ics001.jpg')  # ファイル名
In [3]:im.show() # 表示
In [4]:x = im.size # 画像の大きさは？
In [5]:x
```

PIL（Python Imaging Library）がインストールされていない場合は... ちょっと待っててください

「画像は数字の羅列」を確認

```
$ ipython
In [1]:from PIL import Image
In [2]:im = Image.open('ics001.jpg')  # ファイル名
In [3]:im.show() # 表示
In [4]:x = im.size # 画像の大きさは？
In [5]:x
Out[5]:(800, 649)
```

PIL（Python Imaging Library）がインストールされていない場合は... ちょっと待っててください

「画像は数字の羅列」を確認

```
$ ipython
In [1]:from PIL import Image
In [2]:im = Image.open('ics001.jpg')  # ファイル名
In [3]:im.show() # 表示
In [4]:x = im.size # 画像の大きさは？
In [5]:x
Out[5]:(800, 649)
In [6]:x[0]
```

PIL（Python Imaging Library）がインストールされていない場合は... ちょっと待っててください

「画像は数字の羅列」を確認

```
$ ipython
In [1]:from PIL import Image
In [2]:im = Image.open('ics001.jpg')  # ファイル名
In [3]:im.show() # 表示
In [4]:x = im.size # 画像の大きさは？
In [5]:x
Out[5]:(800, 649)
In [6]:x[0]
Out[6]:800 # 1個目の要素だけ取り出す
```

PIL（Python Imaging Library）がインストールされていない場合は... ちょっと待っててください

「画像は数字の羅列」を確認

```
$ ipython
In [1]:from PIL import Image
In [2]:im = Image.open('ics001.jpg')  # ファイル名
In [3]:im.show() # 表示
In [4]:x = im.size # 画像の大きさは？
In [5]:x
Out[5]:(800, 649)
In [6]:x[0]
Out[6]:800 # 1個目の要素だけ取り出す
In [7]:p = im.getdata() # 画素データを取り出す
```

PIL（Python Imaging Library）がインストールされていない場合は... ちょっと待っててください

「画像は数字の羅列」を確認

```
$ ipython
In [1]:from PIL import Image
In [2]:im = Image.open('ics001.jpg')  # ファイル名
In [3]:im.show() # 表示
In [4]:x = im.size # 画像の大きさは？
In [5]:x
Out[5]:(800, 649)
In [6]:x[0]
Out[6]:800 # 1個目の要素だけ取り出す
In [7]:p = im.getdata() # 画素データを取り出す
In [8]:p
```

PIL（Python Imaging Library）がインストールされていない場合は... ちょっと待っててください

「画像は数字の羅列」を確認

```
$ ipython
In [1]:from PIL import Image
In [2]:im = Image.open('ics001.jpg')  # ファイル名
In [3]:im.show() # 表示
In [4]:x = im.size # 画像の大きさは？
In [5]:x
Out[5]:(800, 649)
In [6]:x[0]
Out[6]:800 # 1個目の要素だけ取り出す
In [7]:p = im.getdata() # 画素データを取り出す
In [8]:p
Out[8]:<ImagingCore at 0xb72d75a0>
```

PIL（Python Imaging Library）がインストールされていない場合は... ちょっと待っててください

「画像は数字の羅列」を確認

```
$ ipython
In [1]:from PIL import Image
In [2]:im = Image.open('ics001.jpg')  # ファイル名
In [3]:im.show() # 表示
In [4]:x = im.size # 画像の大きさは？
In [5]:x
Out[5]:(800, 649)
In [6]:x[0]
Out[6]:800 # 1個目の要素だけ取り出す
In [7]:p = im.getdata() # 画素データを取り出す
In [8]:p
Out[8]:<ImagingCore at 0xb72d75a0>
In [9]:p[3]
```

PIL（Python Imaging Library）がインストールされていない場合は... ちょっと待っててください

「画像は数字の羅列」を確認

```
$ ipython
In [1]:from PIL import Image
In [2]:im = Image.open('ics001.jpg')  # ファイル名
In [3]:im.show() # 表示
In [4]:x = im.size # 画像の大きさは？
In [5]:x
Out[5]:(800, 649)
In [6]:x[0]
Out[6]:800 # 1個目の要素だけ取り出す
In [7]:p = im.getdata() # 画素データを取り出す
In [8]:p
Out[8]:<ImagingCore at 0xb72d75a0>
In [9]:p[3]
Out[9]:(34, 81, 159) # 第3画素のRGB成分（のはず）
```

PIL（Python Imaging Library）がインストールされていない場合は... ちょっと待っててください

「画像は数字の羅列」を確認

```
$ ipython
In [1]:from PIL import Image
In [2]:im = Image.open('ics001.jpg')  # ファイル名
In [3]:im.show() # 表示
In [4]:x = im.size # 画像の大きさは？
In [5]:x
Out[5]:(800, 649)
In [6]:x[0]
Out[6]:800 # 1個目の要素だけ取り出す
In [7]:p = im.getdata() # 画素データを取り出す
In [8]:p
Out[8]:<ImagingCore at 0xb72d75a0>
In [9]:p[3]
Out[9]:(34, 81, 159) # 第3画素のRGB成分（のはず）
In [10]:p[3][1]
```

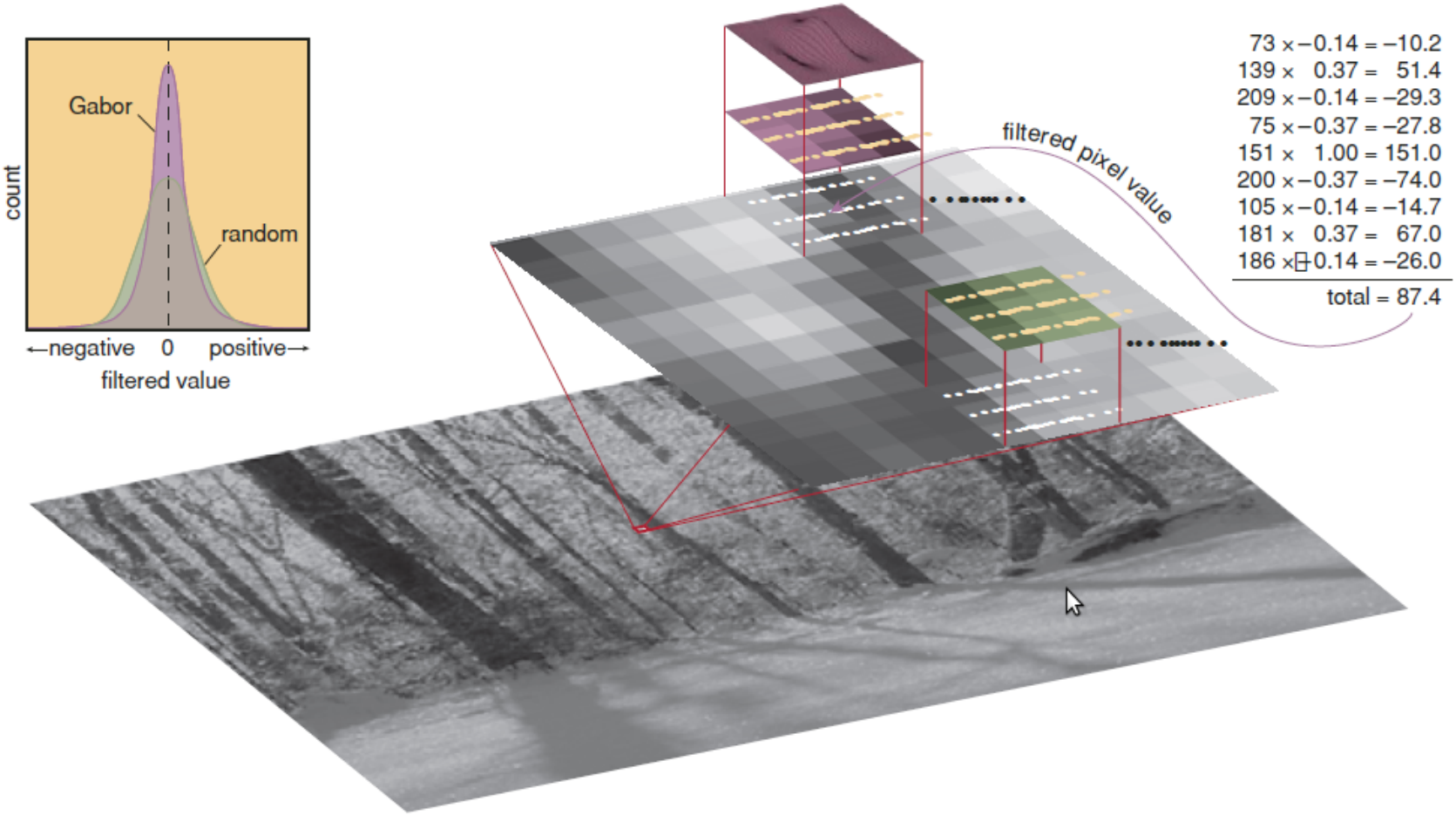
PIL（Python Imaging Library）がインストールされていない場合は... ちょっと待っててください

「画像は数字の羅列」を確認

```
$ ipython
In [1]:from PIL import Image
In [2]:im = Image.open('ics001.jpg')  # ファイル名
In [3]:im.show() # 表示
In [4]:x = im.size # 画像の大きさは？
In [5]:x
Out[5]:(800, 649)
In [6]:x[0]
Out[6]:800 # 1個目の要素だけ取り出す
In [7]:p = im.getdata() # 画素データを取り出す
In [8]:p
Out[8]:<ImagingCore at 0xb72d75a0>
In [9]:p[3]
Out[9]:(34, 81, 159) # 第3画素のRGB成分（のはず）
In [10]:p[3][1]
Out[10]:81
```

PIL（Python Imaging Library）がインストールされていない場合は... ちょっと待っててください

画素値（0〜255）



(B.Olshausen & D.Field, American Scientist, 2000)

im. + tab キーで...

```
$ ipython
```

```
im.app                    im.getpalette          im.putdata
im.applist                im.getpixel            im.putpalette
im.bits                   im.getprojection       im.putpixel
im.category               im.histogram           im.quantization
im.convert                im.huffman_ac          im.quantize
im.copy                   im.huffman_dc          im.readonly
im.crop                   im.icclist             im.resize
im.decoderconfig          im.im                  im.rotate
im.decodermaxblock        im.info                im.save
im.draft                  im.layer               im.seek
im.filename               im.layers              im.show
im.filter                 im.load                im.size
im.format                 im.load_djpeg          im.split
im.format_description     im.load_end            im.tell
im.fp                     im.load_prepare        im.thumbnail
im.fromstring             im.map                 im.tile
im.getbands               im.mode                im.tobitmap
im.getbbox                im.offset              im.tostring
im.getcolors              im.palette             im.transform
im.getdata                im.paste               im.transpose
im.getextrema             im.point               im.verify
im.getim                  im.putalpha
```

im. + tab キーで...

```
$ ipython
In [1]:im.
im.app                      im.getpalette          im.putdata
im.applist                  im.getpixel            im.putpalette
im.bits                     im.getprojection       im.putpixel
im.category                 im.histogram           im.quantization
im.convert                  im.huffman_ac          im.quantize
im.copy                     im.huffman_dc          im.readonly
im.crop                     im.icclist             im.resize
im.decoderconfig            im.im                  im.rotate
im.decodermaxblock          im.info                im.save
im.draft                    im.layer               im.seek
im.filename                 im.layers              im.show
im.filter                   im.load                im.size
im.format                   im.load_djpeg          im.split
im.format_description       im.load_end            im.tell
im.fp                       im.load_prepare        im.thumbnail
im.fromstring               im.map                 im.tile
im.getbands                 im.mode                im.tobitmap
im.getbbox                  im.offset              im.tostring
im.getcolors                im.palette             im.transform
im.getdata                  im.paste               im.transpose
im.getextrema               im.point               im.verify
im.getim                    im.putalpha
```

p. + tab キーで...

```
$ ipython
```

```
p.chop_add                p.effect_spread        p.pixel_access
p.chop_add_modulo         p.expand               p.point
p.chop_and                p.fillband             p.point_transform
p.chop_darker             p.filter               p.putband
p.chop_difference         p.gaussian_blur        p.putdata
p.chop_invert             p.getband              p.putpalette
p.chop_lighter            p.getbbox              p.putpalettealpha
p.chop_multiply           p.getcolors            p.putpixel
p.chop_or                 p.getextrema           p.quantize
p.chop_screen             p.getpalette           p.rankfilter
p.chop_subtract           p.getpixel             p.resize
p.chop_subtract_modulo    p.getprojection        p.rotate
p.chop_xor                p.histogram            p.save_ppm
p.convert                 p.isblock              p.setmode
p.convert2                p.modefilter           p.stretch
p.convert_matrix          p.new_array            p.transform2
p.copy                    p.new_block            p.transpose
p.copy2                   p.offset               p.unsharp_mask
p.crop                    p.paste
```

p. + tab キーで...

```
$ ipython
In [1]:p.
```

```
p.chop_add                p.effect_spread          p.pixel_access
p.chop_add_modulo         p.expand                 p.point
p.chop_and                p.fillband               p.point_transform
p.chop_darker             p.filter                 p.putband
p.chop_difference         p.gaussian_blur          p.putdata
p.chop_invert             p.getband                p.putpalette
p.chop_lighter            p.getbbox                p.putpalettealpha
p.chop_multiply           p.getcolors              p.putpixel
p.chop_or                 p.getextrema             p.quantize
p.chop_screen             p.getpalette             p.rankfilter
p.chop_subtract           p.getpixel               p.resize
p.chop_subtract_modulo    p.getprojection          p.rotate
p.chop_xor                p.histogram              p.save_ppm
p.convert                 p.isblock                p.setmode
p.convert2                p.modefilter             p.stretch
p.convert_matrix          p.new_array              p.transform2
p.copy                    p.new_block              p.transpose
p.copy2                   p.offset                 p.unsharp_mask
p.crop                    p.paste
```

001_grayscale_image.py

```
 1 # -*- coding: utf-8 -*-
 2 import Image
 3
 4 def grayscale(im):
 5     x,y = im.size
 6     for col in range(y):
 7         for raw in range(x):
 8             r,g,b = im.getpixel((raw, col))
 9             m = int(r*0.30 + g*0.59 + b*0.11)
10             im.putpixel((raw,col), (m,m,m))
11     return
12
13 if __name__ == '__main__':
14     im = Image.open("ics001.jpg")
15     im = im.convert("RGB")
16     grayscale(im)
17     im.save("ics001gray.jpg")
18     im.show()
```

「画像は数字の羅列」を確認

scipy, matplotlib だけ使用

```
$ ipython
```

「画像は数字の羅列」を確認

scipy, matplotlib だけ使用

```
$ ipython
In [1]:from scipy import misc # 準備
```

「画像は数字の羅列」を確認

scipy, matplotlib だけ使用

```
$ ipython
In [1]:from scipy import misc # 準備
In [2]:import matplotlib.pyplot as plt
```

「画像は数字の羅列」を確認

scipy, matplotlib だけ使用

```
$ ipython
In [1]:from scipy import misc # 準備
In [2]:import matplotlib.pyplot as plt

In [3]:im = misc.imread('ics001.jpg') #ファイルの指定
```

「画像は数字の羅列」を確認

scipy, matplotlib だけ使用

```
$ ipython
In [1]:from scipy import misc # 準備
In [2]:import matplotlib.pyplot as plt

In [3]:im = misc.imread('ics001.jpg') #ファイルの指定
In [4]:plt.imshow(im)
```

「画像は数字の羅列」を確認

scipy, matplotlib だけ使用

```
$ ipython
In [1]:from scipy import misc # 準備
In [2]:import matplotlib.pyplot as plt

In [3]:im = misc.imread('ics001.jpg') #ファイルの指定
In [4]:plt.imshow(im)
In [5]:plt.show() # 画面に表示
```

「画像は数字の羅列」を確認

scipy, matplotlib だけ使用

```
$ ipython
In [1]:from scipy import misc # 準備
In [2]:import matplotlib.pyplot as plt

In [3]:im = misc.imread('ics001.jpg') #ファイルの指定
In [4]:plt.imshow(im)
In [5]:plt.show() # 画面に表示

In [6]:im.shape
```

「画像は数字の羅列」を確認

scipy, matplotlib だけ使用

```
$ ipython
In [1]:from scipy import misc # 準備
In [2]:import matplotlib.pyplot as plt

In [3]:im = misc.imread('ics001.jpg') #ファイルの指定
In [4]:plt.imshow(im)
In [5]:plt.show() # 画面に表示

In [6]:im.shape
Out[6]:(649, 800, 3)
```

「画像は数字の羅列」を確認

scipy, matplotlib だけ使用

```
$ ipython
In [1]:from scipy import misc # 準備
In [2]:import matplotlib.pyplot as plt

In [3]:im = misc.imread('ics001.jpg') #ファイルの指定
In [4]:plt.imshow(im)
In [5]:plt.show() # 画面に表示

In [6]:im.shape
Out[6]:(649, 800, 3)
In [7]:type(im)
```

「画像は数字の羅列」を確認

scipy, matplotlib だけ使用

```
$ ipython
In [1]:from scipy import misc # 準備
In [2]:import matplotlib.pyplot as plt

In [3]:im = misc.imread('ics001.jpg') #ファイルの指定
In [4]:plt.imshow(im)
In [5]:plt.show() # 画面に表示

In [6]:im.shape
Out[6]:(649, 800, 3)
In [7]:type(im)
Out[7]:numpy.ndarray
```

「画像は数字の羅列」を確認

scipy, matplotlib だけ使用

```
$ ipython
In [1]:from scipy import misc # 準備
In [2]:import matplotlib.pyplot as plt

In [3]:im = misc.imread('ics001.jpg') #ファイルの指定
In [4]:plt.imshow(im)
In [5]:plt.show() # 画面に表示

In [6]:im.shape
Out[6]:(649, 800, 3)
In [7]:type(im)
Out[7]:numpy.ndarray
In [8]:im.dtype
```

「画像は数字の羅列」を確認

scipy, matplotlib だけ使用

```
$ ipython
In [1]:from scipy import misc # 準備
In [2]:import matplotlib.pyplot as plt

In [3]:im = misc.imread('ics001.jpg') #ファイルの指定
In [4]:plt.imshow(im)
In [5]:plt.show() # 画面に表示

In [6]:im.shape
Out[6]:(649, 800, 3)
In [7]:type(im)
Out[7]:numpy.ndarray
In [8]:im.dtype
Out[8]:dtype('uint8')
```

「画像は数字の羅列」を確認

scipy, matplotlib だけ使用

```
$ ipython




# im[300:350:1, 400:450:1,:]=[123,245,82] # 緑化
# im[300:310] = 255 # im[300:310:1, :, :] と同じ
# im[300:310:1, 0:800:1, 0:3:1] と同じ
# im[300:350:1, 400:450:1, 0:3:1] = 255 を試す
```

「画像は数字の羅列」を確認

scipy, matplotlib だけ使用

```
$ ipython
In [1]:im




# im[300:350:1, 400:450:1,:]=[123,245,82] # 緑化
# im[300:310] = 255 # im[300:310:1, :, :] と同じ
# im[300:310:1, 0:800:1, 0:3:1] と同じ
# im[300:350:1, 400:450:1, 0:3:1] = 255 を試す
```

「画像は数字の羅列」を確認

scipy, matplotlib だけ使用

```
$ ipython
In [1]:im
Out[1]:array([[[ 36, 30, 68], [56, 68, 126], [41, 77, 155],



# im[300:350:1, 400:450:1,:]=[123,245,82] # 緑化
# im[300:310] = 255 # im[300:310:1, :, :] と同じ
# im[300:310:1, 0:800:1, 0:3:1] と同じ
# im[300:350:1, 400:450:1, 0:3:1] = 255 を試す
```

「画像は数字の羅列」を確認

scipy, matplotlib だけ使用

```
$ ipython
In [1]:im
Out[1]:array([[[ 36, 30, 68], [56, 68, 126], [41, 77, 155],

In [2]:im[300:350:1, 400:450:1,::]=255 # ある領域だけ白く

# im[300:350:1, 400:450:1,::]=[123,245,82] # 緑化
# im[300:310] = 255 # im[300:310:1, ::, ::] と同じ
# im[300:310:1, 0:800:1, 0:3:1] と同じ
# im[300:350:1, 400:450:1, 0:3:1] = 255 を試す
```

「画像は数字の羅列」を確認

scipy, matplotlib だけ使用

```
$ ipython
In [1]:im
Out[1]:array([[[ 36, 30, 68], [56, 68, 126], [41, 77, 155],

In [2]:im[300:350:1, 400:450:1,::]=255 # ある領域だけ白く

# im[300:350:1, 400:450:1,::]=[123,245,82] # 緑化
# im[300:310] = 255 # im[300:310:1, ::, ::] と同じ
# im[300:310:1, 0:800:1, 0:3:1] と同じ
# im[300:350:1, 400:450:1, 0:3:1] = 255 を試す

In [3]:plt.imshow(im)
```

「画像は数字の羅列」を確認

scipy, matplotlib だけ使用

```
$ ipython
In [1]:im
Out[1]:array([[[ 36, 30, 68], [56, 68, 126], [41, 77, 155],

In [2]:im[300:350:1, 400:450:1,:]=255 # ある領域だけ白く

# im[300:350:1, 400:450:1,:]=[123,245,82] # 緑化
# im[300:310] = 255 # im[300:310:1, :, :] と同じ
# im[300:310:1, 0:800:1, 0:3:1] と同じ
# im[300:350:1, 400:450:1, 0:3:1] = 255 を試す

In [3]:plt.imshow(im)
In [4]:plt.show()
```

「画像は数字の羅列」を確認

scipy, matplotlib だけ使用

```
$ ipython
In [1]:im
Out[1]:array([[[ 36, 30, 68], [56, 68, 126], [41, 77, 155],

In [2]:im[300:350:1, 400:450:1,:]=255 # ある領域だけ白く

# im[300:350:1, 400:450:1,:]=[123,245,82] # 緑化
# im[300:310] = 255 # im[300:310:1, :, :] と同じ
# im[300:310:1, 0:800:1, 0:3:1] と同じ
# im[300:350:1, 400:450:1, 0:3:1] = 255 を試す

In [3]:plt.imshow(im)
In [4]:plt.show()
In [5]:im[330:350] = [100,123,211]
```

「画像は数字の羅列」を確認

scipy, matplotlib だけ使用

```
$ ipython
In [1]:im
Out[1]:array([[[ 36, 30, 68], [56, 68, 126], [41, 77, 155],

In [2]:im[300:350:1, 400:450:1,:]=255 # ある領域だけ白く

# im[300:350:1, 400:450:1,:]=[123,245,82] # 緑化
# im[300:310] = 255 # im[300:310:1, :, :] と同じ
# im[300:310:1, 0:800:1, 0:3:1] と同じ
# im[300:350:1, 400:450:1, 0:3:1] = 255 を試す

In [3]:plt.imshow(im)
In [4]:plt.show()
In [5]:im[330:350] = [100,123,211]
In [6]:plt.imshow(im)
```

「画像は数字の羅列」を確認

scipy, matplotlib だけ使用

```
$ ipython
In [1]:im
Out[1]:array([[[ 36, 30, 68], [56, 68, 126], [41, 77, 155],

In [2]:im[300:350:1, 400:450:1,:]=255 # ある領域だけ白く

# im[300:350:1, 400:450:1,:]=[123,245,82] # 緑化
# im[300:310] = 255 # im[300:310:1, :, :] と同じ
# im[300:310:1, 0:800:1, 0:3:1] と同じ
# im[300:350:1, 400:450:1, 0:3:1] = 255 を試す

In [3]:plt.imshow(im)
In [4]:plt.show()
In [5]:im[330:350] = [100,123,211]
In [6]:plt.imshow(im)
In [7]:plt.show()
```

「画像は数字の羅列」を確認

scipy, matplotlib だけ使用

```
$ ipython
```

「画像は数字の羅列」を確認

scipy, matplotlib だけ使用

```
$ ipython
In [1]:im[0]
```

「画像は数字の羅列」を確認

scipy, matplotlib だけ使用

```
$ ipython
In [1]:im[0]
Out[1]:array([[ 36,  30,  68],....
```

「画像は数字の羅列」を確認

scipy, matplotlib だけ使用

```
$ ipython
In [1]:im[0]
Out[1]:array([[ 36,  30,  68],....

In [2]:im[0][0]
```

「画像は数字の羅列」を確認

scipy, matplotlib だけ使用

```
$ ipython
In [1]:im[0]
Out[1]:array([[ 36,  30,  68],....

In [2]:im[0][0]
Out[2]:array([36, 30, 68], dtype=uint8)
```

「画像は数字の羅列」を確認

scipy, matplotlib だけ使用

```
$ ipython
In [1]:im[0]
Out[1]:array([[ 36,  30,  68],....

In [2]:im[0][0]
Out[2]:array([36, 30, 68], dtype=uint8)
In [3]:im[0][0][0]
```

「画像は数字の羅列」を確認

scipy, matplotlib だけ使用

```
$ ipython
In [1]:im[0]
Out[1]:array([[ 36,  30,  68],....

In [2]:im[0][0]
Out[2]:array([36, 30, 68], dtype=uint8)
In [3]:im[0][0][0]
Out[3]:36
```

「画像は数字の羅列」を確認

scipy, matplotlib だけ使用

```
$ ipython
In [1]:im[0]
Out[1]:array([[ 36,  30,  68],....

In [2]:im[0][0]
Out[2]:array([36, 30, 68], dtype=uint8)
In [3]:im[0][0][0]
Out[3]:36

In [4]:im.shape[0]
```

「画像は数字の羅列」を確認

scipy, matplotlib だけ使用

```
$ ipython
In [1]:im[0]
Out[1]:array([[ 36,  30,  68],....

In [2]:im[0][0]
Out[2]:array([36, 30, 68], dtype=uint8)
In [3]:im[0][0][0]
Out[3]:36

In [4]:im.shape[0]
Out[4]:649 # 縦方向
```

im. + tab キーで...

```
$ ipython
```

```
im.T                im.cumsum           im.min              im.shape
im.all              im.data             im.nbytes           im.size
im.any              im.diagonal         im.ndim             im.sort
im.argmax           im.dot              im.newbyteorder     im.squeeze
im.argmin           im.dtype            im.nonzero          im.std
im.argpartition     im.dump             im.partition        im.strides
im.argsort          im.dumps            im.prod             im.sum
im.astype           im.fill             im.ptp              im.swapaxes
im.base             im.flags            im.put              im.take
im.byteswap         im.flat             im.ravel            im.tofile
im.choose           im.flatten          im.real             im.tolist
im.clip             im.getfield         im.repeat           im.tostring
im.compress         im.imag             im.reshape          im.trace
im.conj             im.item             im.resize           im.transpose
im.conjugate        im.itemset          im.round            im.var
im.copy             im.itemsize         im.searchsorted     im.view
im.ctypes           im.max              im.setfield
im.cumprod          im.mean             im.setflags
```

im. + tab キーで...

```
$ ipython
In [1]:im.
im.T                im.cumsum           im.min              im.shape
im.all              im.data             im.nbytes           im.size
im.any              im.diagonal         im.ndim             im.sort
im.argmax           im.dot              im.newbyteorder     im.squeeze
im.argmin           im.dtype            im.nonzero          im.std
im.argpartition     im.dump             im.partition        im.strides
im.argsort          im.dumps            im.prod             im.sum
im.astype           im.fill             im.ptp              im.swapaxes
im.base             im.flags            im.put              im.take
im.byteswap         im.flat             im.ravel            im.tofile
im.choose           im.flatten          im.real             im.tolist
im.clip             im.getfield         im.repeat           im.tostring
im.compress         im.imag             im.reshape          im.trace
im.conj             im.item             im.resize           im.transpose
im.conjugate        im.itemset          im.round            im.var
im.copy             im.itemsize         im.searchsorted     im.view
im.ctypes           im.max              im.setfield
im.cumprod          im.mean             im.setflags
```

002_grayscale_image.py

```
 1 # -*- coding: utf-8 -*-
 2 from scipy import misc
 3 import matplotlib.pyplot as plt
 4
 5 def grayscale(im):
 6     n = im.shape
 7     for col in range(n[0]):
 8         for raw in range(n[1]):
 9             r,g,b = im[col][raw]
10             m = int(r*0.30 + g*0.59 + b*0.11)
11             im[col][raw] = m #  =[m,m,m] と同じ
12     return
13
14 if __name__ == '__main__':
15     im = misc.imread('ics001.jpg')
16     grayscale(im)
17     misc.imsave("ics001gray.jpg", im)
18     plt.imshow(im)
19     plt.show()
```

002f_grayscale_image.py

```
1  # -*- coding: utf-8 -*-
2  from scipy import misc
3  import matplotlib.pyplot as plt
4
5  def grayscale(im):
6      tmp = im[:,:,:1]*0.30 + im[:,:,1:2]*0.59 +
           im[:,:,2:3]*0.11
7      im[:,:,:3] = tmp[:,:,:1]
8
9      return
10
11 if __name__ == '__main__':
12     im = misc.imread('ics001.jpg')
13     grayscale(im)
14     misc.imsave("ics001gray.jpg", im)
15     plt.imshow(im)
16     plt.show()
```

速い！

「画像」のグレースケール化

まとめ

```
$ ipython
```

「画像」のグレースケール化

まとめ

```
$ ipython
In [1]:from scipy import misc
```

「画像」のグレースケール化

まとめ

```
$ ipython
In [1]:from scipy import misc
In [2]:import matplotlib.pyplot as plt
```

「画像」のグレースケール化

まとめ

```
$ ipython
In [1]:from scipy import misc
In [2]:import matplotlib.pyplot as plt

In [3]:im = misc.imread('ics001.jpg')
```

「画像」のグレースケール化

まとめ

```
$ ipython
In [1]:from scipy import misc
In [2]:import matplotlib.pyplot as plt

In [3]:im = misc.imread('ics001.jpg')

In [4]: tmp=im[:,:,:1]*0.30+im[:,:,1:2]*0.59+im[:,:,2:3]*0.11
```

「画像」のグレースケール化

まとめ

```
$ ipython
In [1]:from scipy import misc
In [2]:import matplotlib.pyplot as plt

In [3]:im = misc.imread('ics001.jpg')

In [4]: tmp=im[:,:,:1]*0.30+im[:,:,1:2]*0.59+im[:,:,2:3]*0.11

In [5]:im[:,:,:3] = tmp[:,:,:1]
```

「画像」のグレースケール化
まとめ

```
$ ipython
In [1]:from scipy import misc
In [2]:import matplotlib.pyplot as plt

In [3]:im = misc.imread('ics001.jpg')

In [4]: tmp=im[:,:,:1]*0.30+im[:,:,1:2]*0.59+im[:,:,2:3]*0.11

In [5]:im[:,:,:3] = tmp[:,:,:1]

In [6]:plt.imshow(im)
```

「画像」のグレースケール化

まとめ

```
$ ipython
In [1]:from scipy import misc
In [2]:import matplotlib.pyplot as plt

In [3]:im = misc.imread('ics001.jpg')

In [4]: tmp=im[:,:,:1]*0.30+im[:,:,1:2]*0.59+im[:,:,2:3]*0.11

In [5]:im[:,:,:3] = tmp[:,:,:1]

In [6]:plt.imshow(im)
Out[6]:<matplotlib.image.AxesImage at 0x9589c6c>
```

「画像」のグレースケール化
まとめ

```
$ ipython
In [1]:from scipy import misc
In [2]:import matplotlib.pyplot as plt

In [3]:im = misc.imread('ics001.jpg')

In [4]: tmp=im[:,:,:1]*0.30+im[:,:,1:2]*0.59+im[:,:,2:3]*0.11

In [5]:im[:,:,:3] = tmp[:,:,:1]

In [6]:plt.imshow(im)
Out[6]:<matplotlib.image.AxesImage at 0x9589c6c>

In [7]:plt.show()
```

python のスライス操作

```
$ ipython
```

```
array([[[ 0,  1,  2],
        [ 3,  4,  5]],

       [[ 6,  7,  8],
        [ 9, 10, 11]],

       [[12, 13, 14],
        [15, 16, 17]],

       [[18, 19, 20],
        [21, 22, 23]]])
```

```
array([[[ 0,  1,  2],
        [ 3,  4,  5]],

       [[ 6,  7,  8],
        [ 9, 10, 11]]])
```

いろいろ試して確認する. （最初）:（最後ー１）:Δ

python のスライス操作

```
$ ipython
In [1]:from numpy import *
```

```
array([[[ 0,  1,  2],
        [ 3,  4,  5]],

       [[ 6,  7,  8],
        [ 9, 10, 11]],

       [[12, 13, 14],
        [15, 16, 17]],

       [[18, 19, 20],
        [21, 22, 23]]])
```

```
array([[[ 0,  1,  2],
        [ 3,  4,  5]],

       [[ 6,  7,  8],
        [ 9, 10, 11]]])
```

python のスライス操作

```
$ ipython
In [1]:from numpy import *
In [2]:A = array ([1 , 2, 3])
```

```
array([[[ 0,  1,  2],
        [ 3,  4,  5]],

       [[ 6,  7,  8],
        [ 9, 10, 11]],

       [[12, 13, 14],
        [15, 16, 17]],

       [[18, 19, 20],
        [21, 22, 23]]])


array([[[ 0,  1,  2],
        [ 3,  4,  5]],

       [[ 6,  7,  8],
        [ 9, 10, 11]]])
```

いろいろ試して確認する. （最初）:（最後ー１）:Δ

python のスライス操作

```
$ ipython
In [1]:from numpy import *
In [2]:A = array ([1 , 2, 3])
In [3]:A = array(range(24))
```

```
array([[[ 0,  1,  2],
        [ 3,  4,  5]],

       [[ 6,  7,  8],
        [ 9, 10, 11]],

       [[12, 13, 14],
        [15, 16, 17]],

       [[18, 19, 20],
        [21, 22, 23]]])
```

```
array([[[ 0,  1,  2],
        [ 3,  4,  5]],

       [[ 6,  7,  8],
        [ 9, 10, 11]]])
```

いろいろ試して確認する. (最初) : (最後ー１) : Δ

python のスライス操作

```
$ ipython
In [1]:from numpy import *
In [2]:A = array ([1 , 2, 3])
In [3]:A = array(range(24))
In [4]:A = A.reshape(8,3) # 長方形
```

```
array([[[ 0,  1,  2],
        [ 3,  4,  5]],

       [[ 6,  7,  8],
        [ 9, 10, 11]],

       [[12, 13, 14],
        [15, 16, 17]],

       [[18, 19, 20],
        [21, 22, 23]]])
```

```
array([[[ 0,  1,  2],
        [ 3,  4,  5]],

       [[ 6,  7,  8],
        [ 9, 10, 11]]])
```

いろいろ試して確認する. （最初）:（最後ー１）:Δ

python のスライス操作

```
$ ipython
In [1]:from numpy import *
In [2]:A = array ([1 , 2, 3])
In [3]:A = array(range(24))
In [4]:A = A.reshape(8,3) # 長方形
In [5]:A = A.reshape(4,2,3) # 直方体をイメージ！
array([[[ 0,  1,  2],
        [ 3,  4,  5]],

       [[ 6,  7,  8],
        [ 9, 10, 11]],

       [[12, 13, 14],
        [15, 16, 17]],

       [[18, 19, 20],
        [21, 22, 23]]])


array([[[ 0,  1,  2],
        [ 3,  4,  5]],

       [[ 6,  7,  8],
        [ 9, 10, 11]]])
```

いろいろ試して確認する. （最初）:（最後ー１）:Δ

python のスライス操作

```
$ ipython
In [1]:from numpy import *
In [2]:A = array ([1 , 2, 3])
In [3]:A = array(range(24))
In [4]:A = A.reshape(8,3) # 長方形
In [5]:A = A.reshape(4,2,3) # 直方体をイメージ！
array([[[ 0,  1,  2],
        [ 3,  4,  5]],

       [[ 6,  7,  8],
        [ 9, 10, 11]],

       [[12, 13, 14],
        [15, 16, 17]],

       [[18, 19, 20],
        [21, 22, 23]]])
In [6]:A[0:2] # A[0:2:1,0:2:1,0:4:1] と同じ
array([[[ 0,  1,  2],
        [ 3,  4,  5]],

       [[ 6,  7,  8],
        [ 9, 10, 11]]])
```

いろいろ試して確認する. （最初）:（最後ー１）:Δ

python のスライス操作

```
$ ipython
```

```
array([[[ 0,  1,  2],
        [ 3,  4,  5]],

       [[ 6,  7,  8],
        [ 9, 10, 11]],

       [[12, 13, 14],
        [15, 16, 17]],

       [[18, 19, 20],
        [21, 22, 23]]])
```

(1,7,13,19) と，一本だけ取り出すには？

```
array([[[ 1]],
       [[ 7]],
       [[13]],
       [[19]]])
```

いろいろ試して確認する．（最初）:（最後ー１）:Δ

python のスライス操作

```
$ ipython
In [1]:A = A.reshape(4,2,3) # 直方体をイメージ！
array([[[ 0,  1,  2],
        [ 3,  4,  5]],

       [[ 6,  7,  8],
        [ 9, 10, 11]],

       [[12, 13, 14],
        [15, 16, 17]],

       [[18, 19, 20],
        [21, 22, 23]]])
```

(1,7,13,19) と，一本だけ取り出すには？

```
array([[[ 1]],
       [[ 7]],
       [[13]],
       [[19]]])
```

いろいろ試して確認する．（最初）:（最後ー１）:Δ

python のスライス操作

```
$ ipython
In [1]:A = A.reshape(4,2,3) # 直方体をイメージ！
array([[[ 0,  1,  2],
        [ 3,  4,  5]],

       [[ 6,  7,  8],
        [ 9, 10, 11]],

       [[12, 13, 14],
        [15, 16, 17]],

       [[18, 19, 20],
        [21, 22, 23]]])
```

(1,7,13,19) と，一本だけ取り出すには？

```
In [2]:A[:,0:1,1:2]
array([[[ 1]],
       [[ 7]],
       [[13]],
       [[19]]])
```

いろいろ試して確認する．（最初）:（最後ー１）:Δ

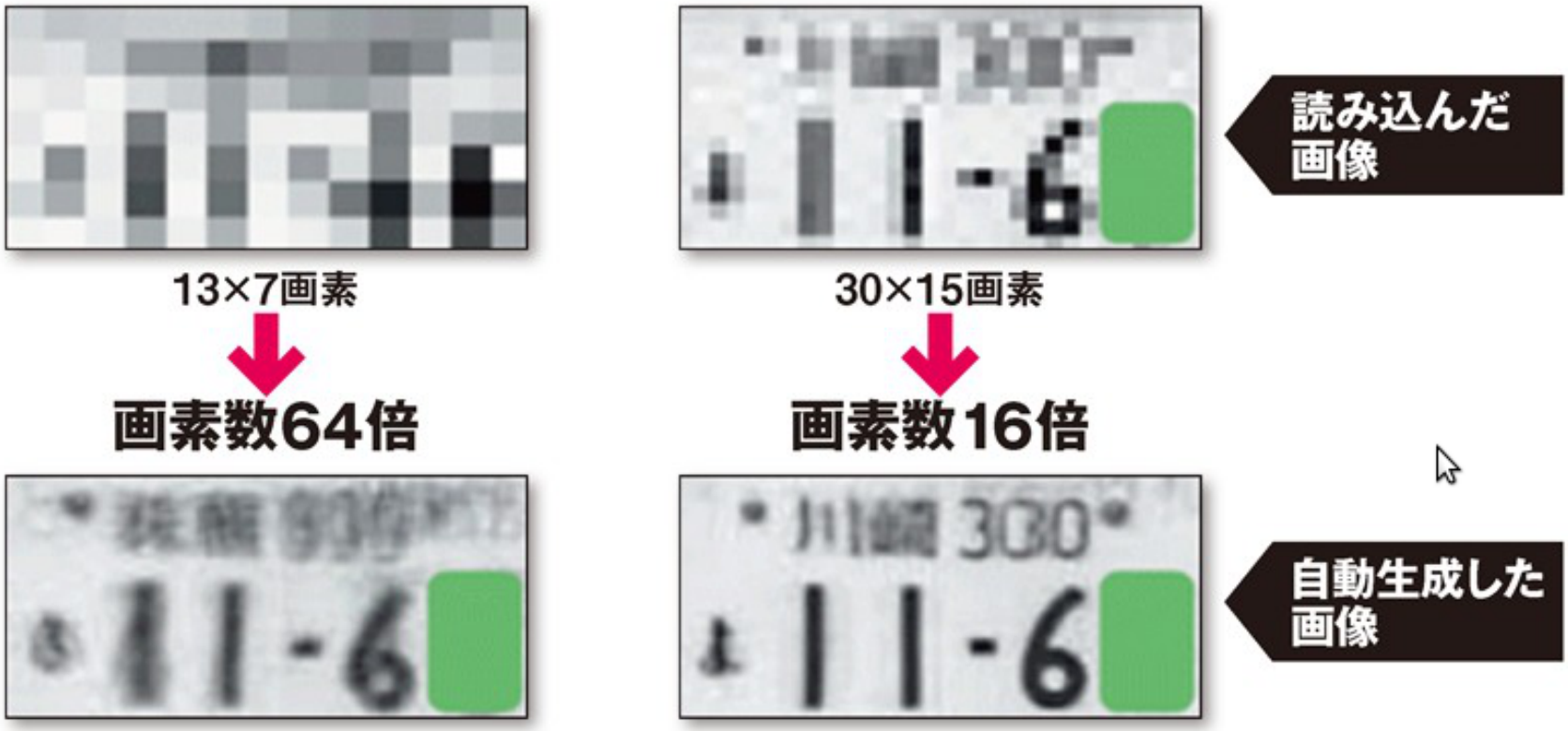
参考資料

1. Python Scientific Lecture Notes
   `http://scipy-lectures.github.io/index.html`

機械学習

低解像度の画像から高解像度の画像を生成

図6 NECが開発した「学習型超解像技術」



（日経コンピュータ 2014 年 1 月 9 日号）

本日の課題

- 感想・質問を 200 字程度で述べよ．できるだけ他人と違うコメントになることを意識して書くこと．字数が短いので，言いたいこと・要点だけ書くこと．

- メールで date@cs.miyazaki-u.ac.jp 宛に提出．テキストメールを送ること（Word ファイルを添付しない）．

- 本文の最初か最後に所属学科・氏名を記述

- Subject（メールの件名） は 20140123ic-0000000（学生証番号）

終