image processing

1. Manipulating Images
   - images are matrices
   - changing color intensities
2. Filtering Images
   - applying convolve of ndimage
   - edge detection with Sobel filters
3. Python Imaging Library and JuliaImages
   - the package Pillow
   - working with images in Julia

MCS 507 Lecture 27
Mathematical, Statistical and Scientific Software
Jan Verschelde, 23 October 2023

image processing

1. Manipulating Images
   - images are matrices
   - changing color intensities
2. Filtering Images
   - applying convolve of ndimage
   - edge detection with Sobel filters
3. Python Imaging Library and JuliaImages
   - the package Pillow
   - working with images in Julia

imageio

The `imread` in the scipy package is deprecated.

Imageio is a Python library that provides an easy interface
to read and write a wide range of image data, including
animated images, video, volumetric data, and scientific formats.
It is cross-platform, runs on Python 2.7 and 3.4+, and is easy to install.

Imageio started out as component of the scikit-image project.

web site: `https://imageio.github.io/`

a familiar image

a first script

We bring an image into a Python session and look at the type, size, and shape:

```
from imageio import imread
from matplotlib import pyplot as plt

seo = imread('buildingseo.jpg')
print(type(seo))
print(seo.size)
print(seo.shape)
plt.imshow(seo)
plt.show()
```

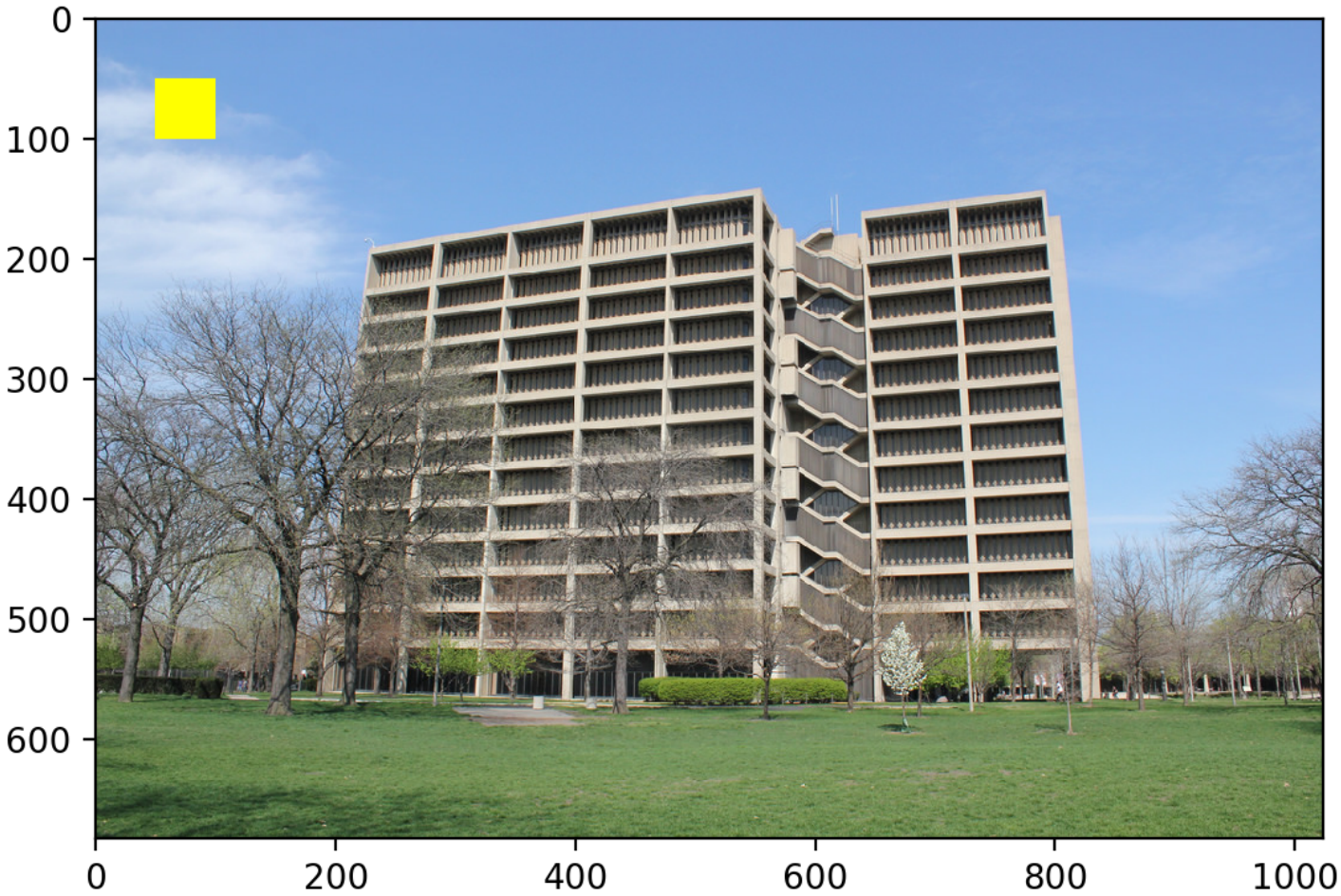
output of the script

The script shows a figure with the image and prints

```
$ python3 buildingseo.py
<class 'imageio.core.util.Array'>
2098176
(683, 1024, 3)
```

- The image is a 3-dimensional matrix of dimensions 683, 1024, and 3, of 683 rows and 1024 columns.
- The third dimension holds the color intensities as red, green, blue numbers in the range between 0 and 255.
- The origin (0, 0) is at the top left of the image.

adding a yellow square



slicing a numpy matrix

To calculate with images we convert to a numpy matrix.
We place a yellow square in the top left corner of the picture.

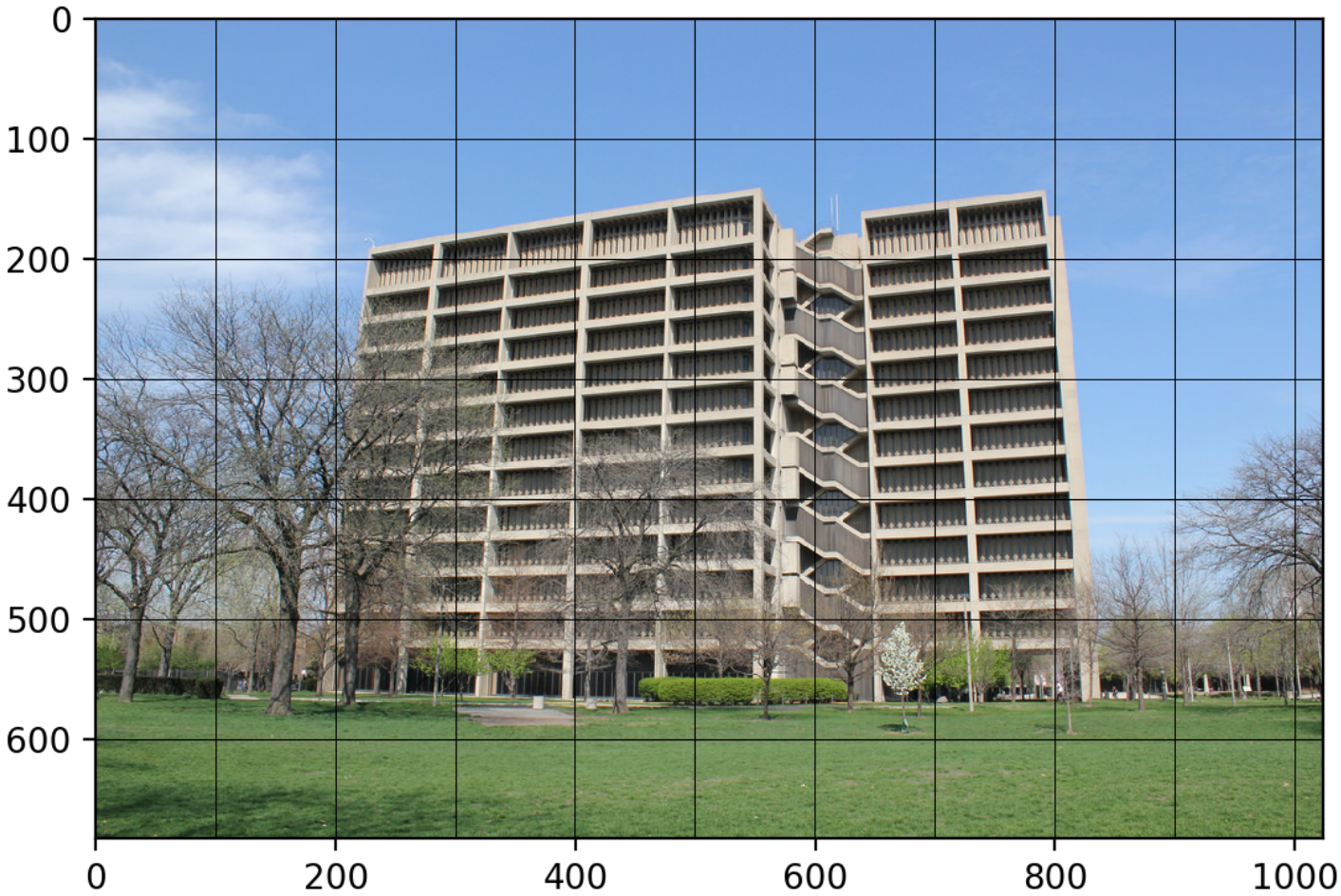
```
from imageio import imread
from matplotlib import pyplot as plt
import numpy as np

seo = imread('buildingseo.jpg')
seomat = np.copy(seo)
seomat[50:100, 50:100] = [255, 255, 0]

plt.imshow(seomat)
plt.show()
```

Yellow is the sum of red and green intensities.

placing a grid



script to place a grid

We place a grid of black lines, every 100 rows and columns.

```
from imageio import imread
from matplotlib import pyplot as plt
import numpy as np

seo = imread('buildingseo.jpg')
seomat = np.copy(seo)

idxgrid = 100
seomat[idxgrid:-1:idxgrid, :] = [0, 0, 0]
seomat[:, idxgrid:-1:idxgrid] = [0, 0, 0]

plt.imshow(seomat)
plt.show()
```

indexing numpy arrays

```
import numpy as np
A = np.array(range(21))
print(A)
for k in range(2,11):
    print('multiples of', k, ':', A[0::k])
for k in range(2,11):
    print('multiples of', k, ':', A[k::k])
```

output of the script

```
[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20]
multiples of 2 : [ 0  2  4  6  8 10 12 14 16 18 20]
multiples of 3 : [ 0  3  6  9 12 15 18]
multiples of 4 : [ 0  4  8 12 16 20]
multiples of 5 : [ 0  5 10 15 20]
multiples of 6 : [ 0  6 12 18]
multiples of 7 : [ 0  7 14]
multiples of 8 : [ 0  8 16]
multiples of 9 : [ 0  9 18]
multiples of 10 : [ 0 10 20]
multiples of 2 : [ 2  4  6  8 10 12 14 16 18 20]
multiples of 3 : [ 3  6  9 12 15 18]
multiples of 4 : [ 4  8 12 16 20]
multiples of 5 : [ 5 10 15 20]
multiples of 6 : [ 6 12 18]
multiples of 7 : [ 7 14]
multiples of 8 : [ 8 16]
multiples of 9 : [ 9 18]
multiples of 10 : [10 20]
```

image processing

1. Manipulating Images
   - images are matrices
   - changing color intensities
2. Filtering Images
   - applying convolve of ndimage
   - edge detection with Sobel filters
3. Python Imaging Library and JuliaImages
   - the package Pillow
   - working with images in Julia

removing the blue out of a picture

```
from imageio import imread, imwrite
from matplotlib import pyplot as plt
import numpy as np
seo = imread('buildingseo.jpg')
seomat = np.copy(seo)
nbrows = np.size(seomat,0)
nbcols = np.size(seomat,1)

seomatnoblue = np.copy(seomat)
seomatnoblue[:,:,2] = np.zeros((nbrows, nbcols), \
    dtype=int)
imwrite('buildingseonoblue.jpg', seomatnoblue)
```

without blue

script continues

We remove also the green intensities, assigning a zero matrix at position 1 of the third dimension:

```
seomatnoblue[:,:,1] = np.zeros((nbrows, nbcols), \
    dtype=int)
imwrite('buildingseoredonly.jpg', seomatnoblue)
```

and then set the same intensity as in red

```
seomatnoblue[:,:,1] = seomatnoblue[:,:,0]
seomatnoblue[:,:,2] = seomatnoblue[:,:,0]
imwrite('buildingsoeredsame.jpg', seomatnoblue)
```

without blue and without green

all in the same color intensity as red

script continues

We increase the red intensities:

```
seomatred = np.copy(seomat)
seomatred[:,:,0] = seomatred[:,:,0]*1.5
imwrite('buildingseotoored.jpg', seomatred)
```

with increased red intensity

script continues

We convert the picture into grayscale:

```
seomatgray = np.copy(seo)
gray = 0.2989*seomatgray[:,:,0] \
     + 0.5870*seomatgray[:,:,1] \
     + 0.1140*seomatgray[:,:,2]
seomatgray[:,:,0] = gray
seomatgray[:,:,1] = gray
seomatgray[:,:,2] = gray
imwrite('buildingseoingray.jpg', seomatgray)
```

in grayscale

image processing

1. Manipulating Images
   - images are matrices
   - changing color intensities
2. Filtering Images
   - applying convolve of ndimage
   - edge detection with Sobel filters
3. Python Imaging Library and JuliaImages
   - the package Pillow
   - working with images in Julia

multi-dimensional image processing with ndimage

```
>>> from scipy import ndimage
>>> help(ndimage)
```

The package ndimage contains various functions
for multi-dimensional image processing.

We follow chapter 3 of *Elegant SciPy* by Juan Nunez-Iglesias, Harriet Dashnow, and Stéfan van der Walt, published by O'Reilly Media, 2017.

convolution of a signal

The script below

1. defines a signal in the 30 to 60 range,
2. applies the convolve of ndimage.

```
import numpy as np
from scipy import ndimage as ndi
sig = np.zeros(100, np.float)
sig[30:60] = 1
kernel = np.array([1, 0, -1])
filtered = ndi.convolve(sig, kernel)
```

The `convolve` with kernel `[1, 0, -1]`
replaces `sig[i]` with `sig[i+1]` + 0 × `sig[i]` − `sig[i-1]`.

the plotting instructions

```
from matplotlib import pyplot as plt

fig = plt.figure()

ax = fig.add_subplot(211)
ax.plot(sig);
ax.set_ylim(-0.1, 1.1);
ax = fig.add_subplot(212)
ax.plot(filtered);

plt.show()
```

the signal and its convolution

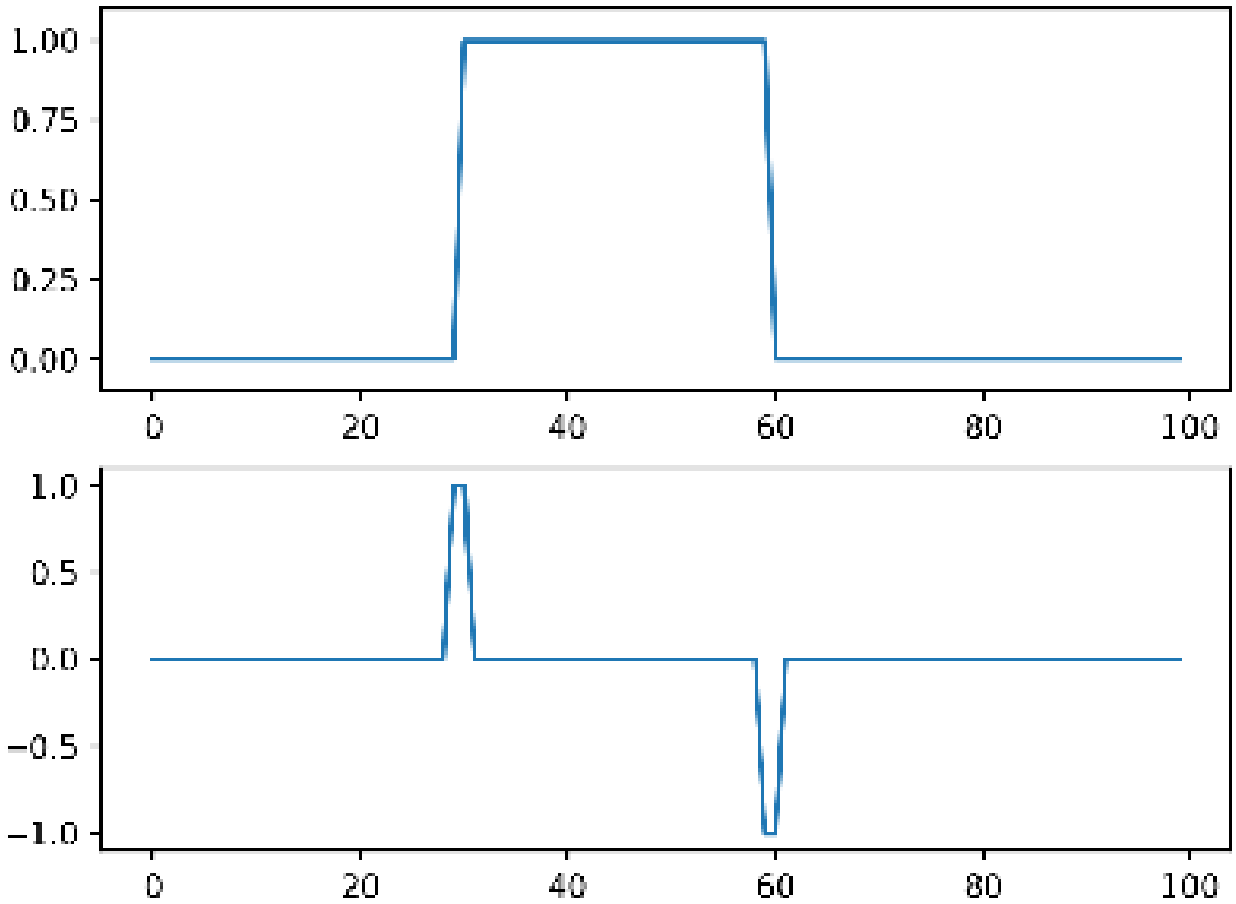


image processing

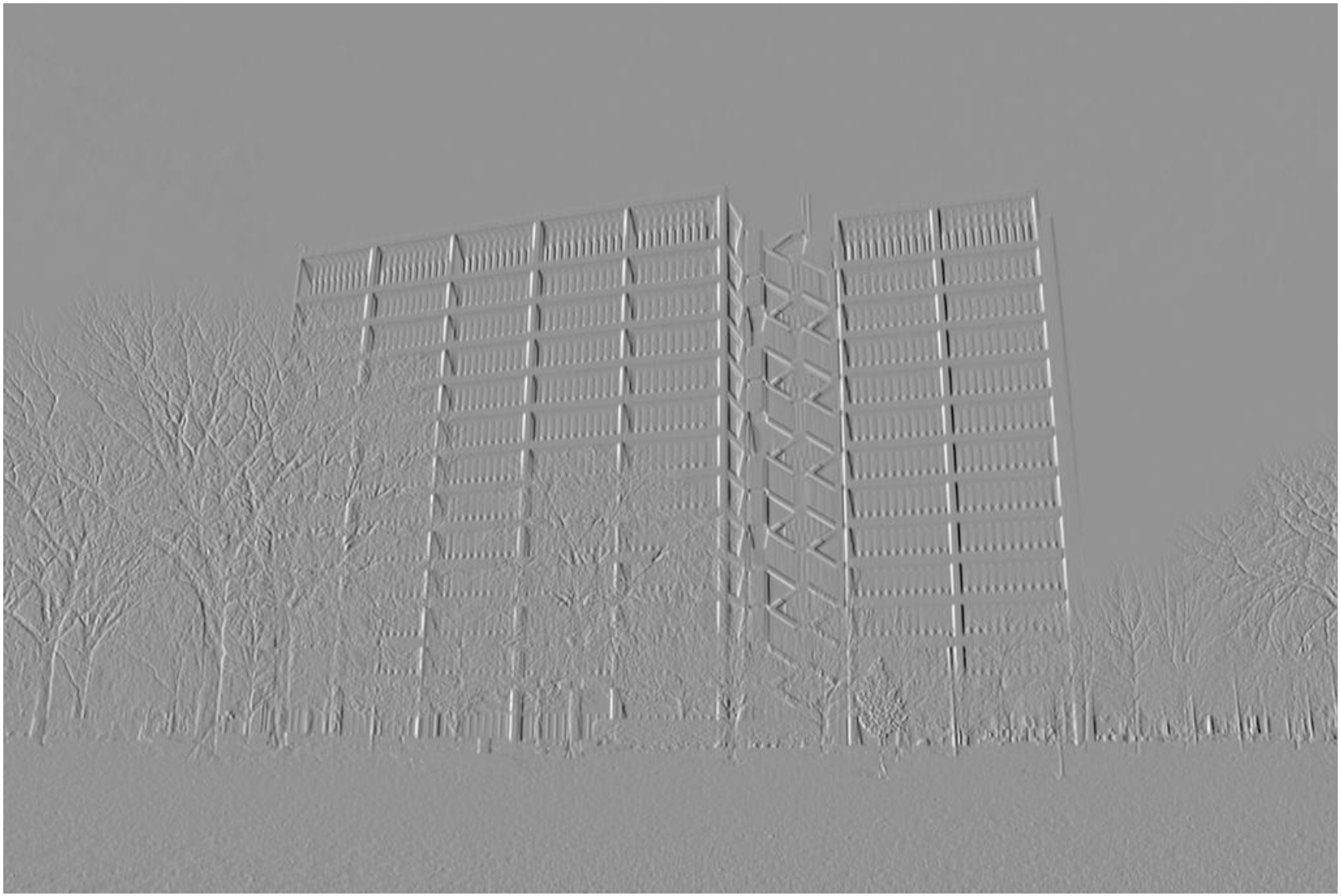
1. Manipulating Images
   - images are matrices
   - changing color intensities
2. Filtering Images
   - applying convolve of ndimage
   - edge detection with Sobel filters
3. Python Imaging Library and JuliaImages
   - the package Pillow
   - working with images in Julia

applying Sobel filters

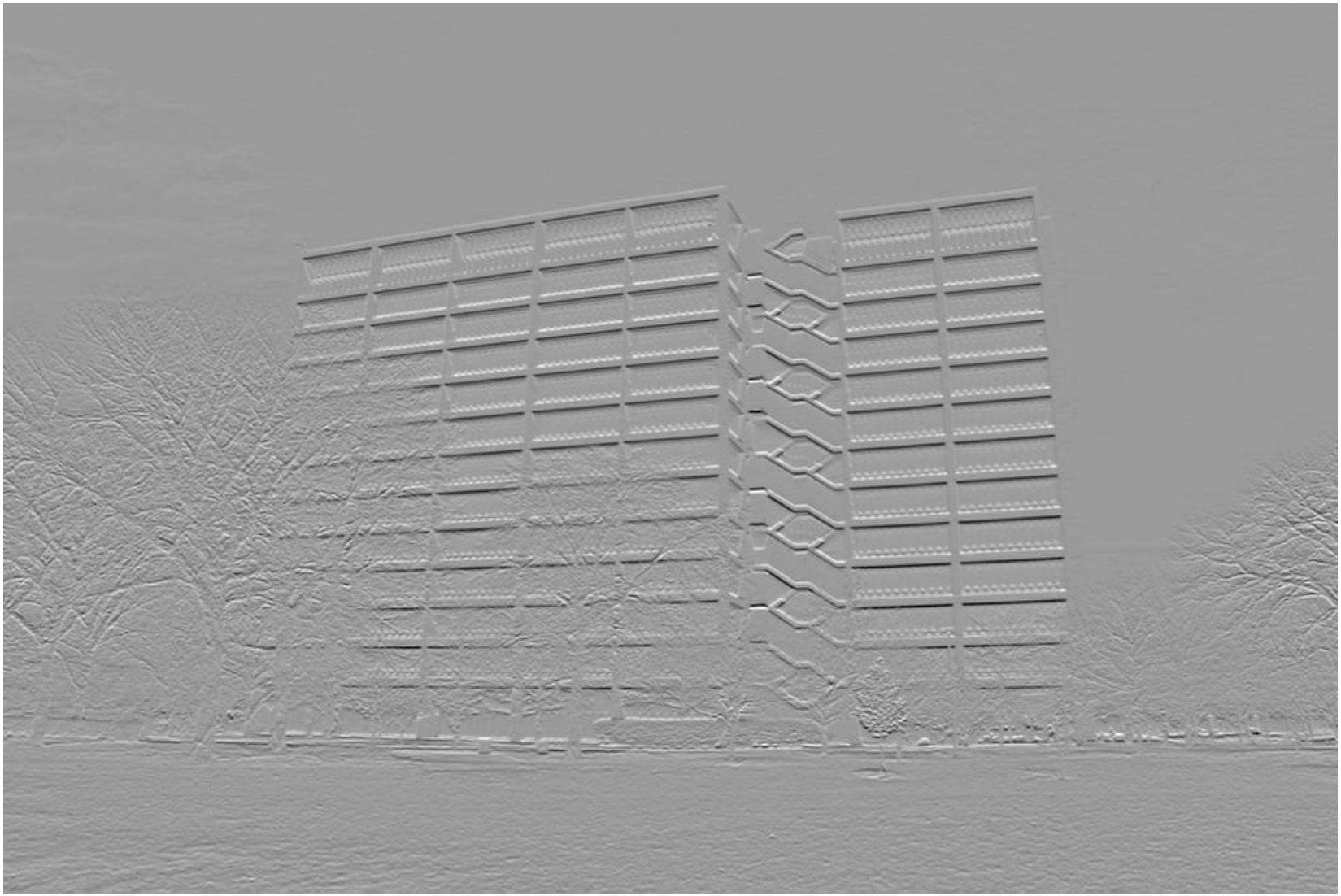
Starting from the grayscale image,
to prevent overflow we divide by 255.

```
seomat = seomat.astype(float)/255
hsobel = np.array([[ 1,  2,  1],
                   [ 0,  0,  0],
                   [-1, -2, -1]])
vsobel = hsobel.T
seo_vertical = ndi.convolve(seomat[:,:,0], vsobel)
seo_horizontal = ndi.convolve(seomat[:,:,0], hsobel)
imwrite('buildingseovertical.jpg', seo_vertical)
imwrite('buildingseohorizontal.jpg', seo_horizontal)
```

vertical edges



horizontal edges



edge detection continued

```
seomat = seomat.astype(float)/255
hsobel = np.array([[ 1,  2,  1],
                   [ 0,  0,  0],
                   [-1, -2, -1]])
vsobel = hsobel.T
seo_vertical = ndi.convolve(seomat[:,:,0], vsobel)
seo_horizontal = ndi.convolve(seomat[:,:,0], hsobel)
seo_sobel = np.sqrt(seo_horizontal**2 + seo_vertical**2)
plt.imshow(seo_sobel, cmap='inferno')
plt.show()
```

the edges of the seo building

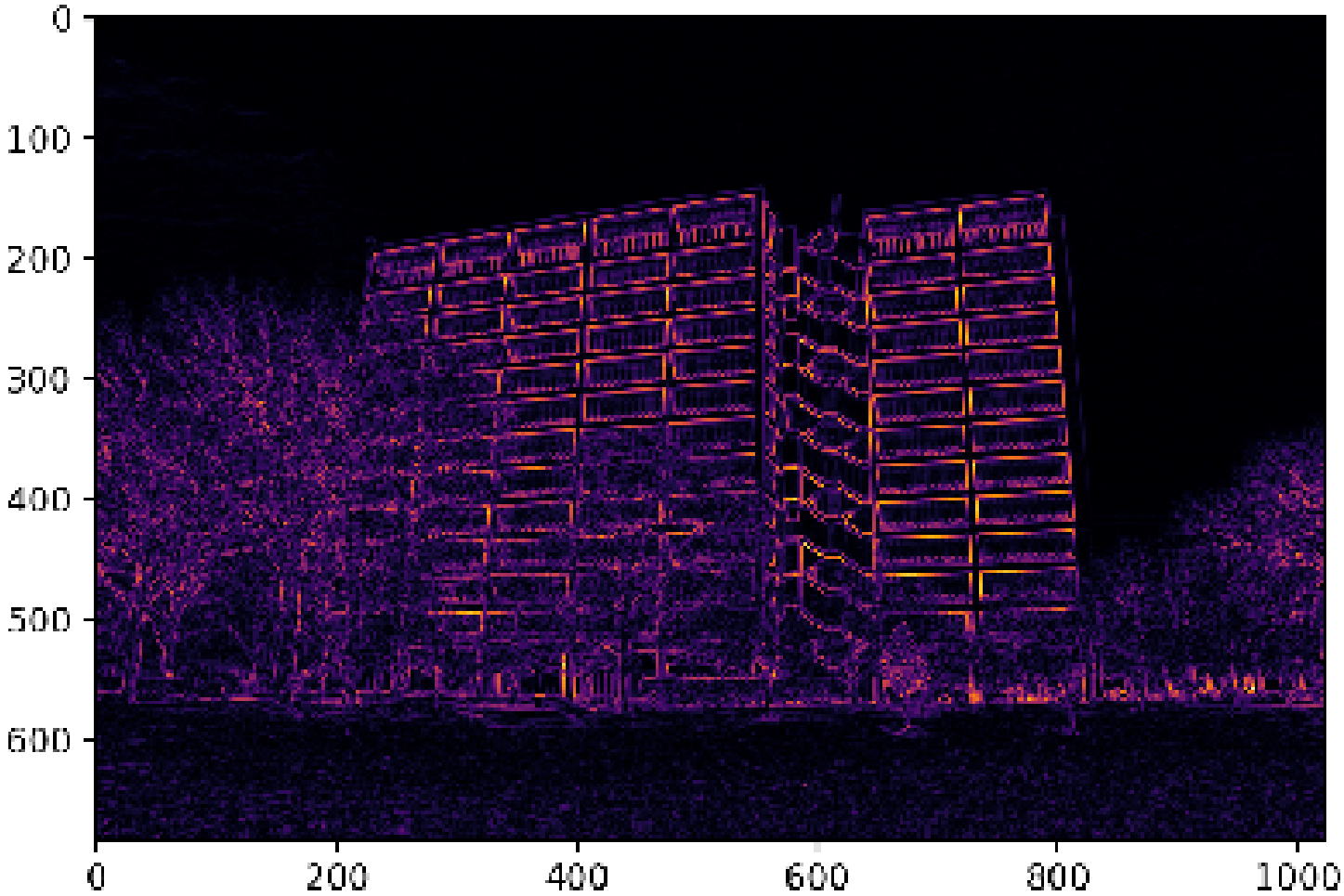


image processing

1. Manipulating Images
   - images are matrices
   - changing color intensities
2. Filtering Images
   - applying convolve of ndimage
   - edge detection with Sobel filters
3. Python Imaging Library and JuliaImages
   - the package Pillow
   - working with images in Julia

Python Imaging Library

The Python Imaging Library (PIL) is a free library developed by Secret Labs AB, first released in 1995.

A successor project to PIL is the fork Pillow.

Main features:

- support for many file formats;
- many standard image manipulation procedures.

grayscale and image enhancement

Converting an image into grayscale is straightforward:

```
from PIL import Image
img = Image.open('buildingseo.jpg').convert('L')
img.save('buildingseograyscale.png')
```

The grass looks greener with more contrast:

```
from PIL import Image, ImageEnhance
img = Image.open('buildingseo.jpg')
img.show()
enh = ImageEnhance.Contrast(img)
enh.enhance(1.3).show("30% more contrast")
```

image processing

1. Manipulating Images
   - images are matrices
   - changing color intensities
2. Filtering Images
   - applying convolve of ndimage
   - edge detection with Sobel filters
3. Python Imaging Library and JuliaImages
   - the package Pillow
   - working with images in Julia

working with images in Julia

The `JuliaImages` hosts the packages for image processing and machine vision in Julia.

Two packages:

- The package `ImageTransformation.jl` provides tools for resizing, rotating, and other coordinate transformations.
- The package `ImageSegmentation.jl` provides image segmentation algorithms.

exercises

1. A color image can be converted to a grayscale by averaging the color intensities. Convert the image of the SEO building to grayscale and reduce the number of different gray scales to 10, via a reduction of the gray intensity to the first digit.
2. Define a sequence of grayscale images, starting at the image of the previous exercise and gradually adding the digits of the gray intensity, increasing its precision.
3. Apply the edge detection on the SEO building picture in Julia.