MENOUFIA UNIVERSITY
FACULTY OF COMPUTERS AND INFORMATION
INFORMATION TECHNOLOGY
COMPUTER VISION

جامعة المنوفية
كلية الحاسبات والمعلومات
تكنولوجيا المعلومات
الرؤيا بالحاسب

Computer Vision: Correlation, Convolution, and Gradient

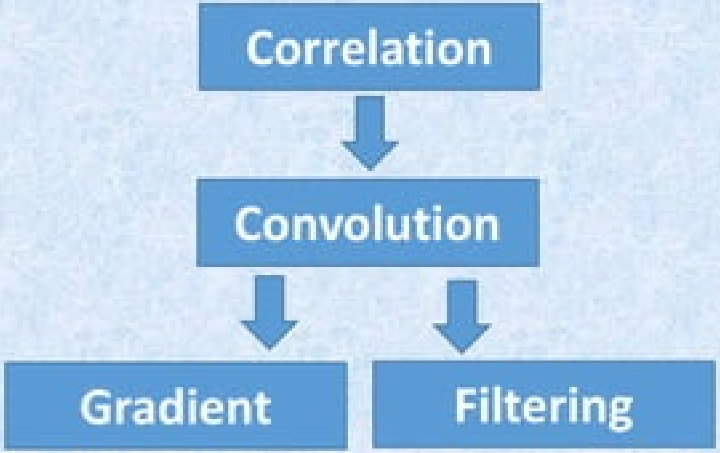
Ahmed Fawzy Gad

ahmed.fawzy@ci.menofia.edu.eg

Tuesday 3 October 2017

Index

- Correlation
  - Python Implementation
- Convolution
  - Python Implementation
- Gradient
  - Python Implementation



CORRELATION

Correlation for 2D Image

- Correlation is used to match a template to an image.
- Can you tell where the following template exactly located in the image?
- Using correlation we can find the image region that best matches the template.

How 2D Correlation Works?

- Given a template, using correlation the template will pass through each image part and a similarity check take place to find how similar the template and the current image part being processed.
- Starting by placing the template top-left corner on the top-left corner of the image, a similarity measure is calculated.

How 2D Correlation Works?

- Given a template, using correlation the template will pass through each image part and a similarity check take place to find how similar the template and the current image part being processed.
- Starting by placing the template top-left corner on the top-left corner of the image, a similarity measure is calculated.

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} h[u,v]F[i+u,j+v]$$

How 2D Correlation Works?

- Given a template, using correlation the template will pass through each image part and a similarity check take place to find how similar the template and the current image part being processed.
- Starting by placing the template top-left corner on the top-left corner of the image, a similarity measure is calculated.

Correlation for 2D Image

1	3	4	3
2	7	4	1
6	2	5	2
13	6	8	9

Template – 3x3

2	1	-4
3	2	5
-1	8	1

Correlation for 2D Image

1	3	4	3
2	7	4	1
6	2	5	2
13	6	8	9

Template – 3x3

2	1	-4
3	2	5
-1	8	1

Correlation for 2D Image

1	3	4	3
2	**7**	4	1
6	2	5	2
13	6	8	9

Template – 3x3

$$2*1+1*3-4*4+3*2+2*7+5*4$$
$$-1*6+8*2+1*5$$
$$=\mathbf{44}$$

Correlation for 2D Image

	44		

Template – 3x3

1	3	4	3
2	7	4	1
6	2	5	2
13	6	8	9

Did you get why template sizes are odd?

Even Template Size

58	**3**	**213**	81	78
185	**87**	**32**	27	11
70	**66**	[redacted]	[redacted]	[redacted]
61	91	[redacted]	[redacted]	[redacted]
14	7	[redacted]	[redacted]	[redacted]

???

84

Even template sizes has no center.
E.g. 2x3, 2x2, 5x6, …

Correlation for 2D Image

	44		

Template – 3x3

2	1	-4
3	2	5
-1	8	1

1	3	4	3
2	7	4	1
6	2	5	2
13	6	8	9

Correlation for 2D Image

	44		

Template – 3x3

1	3	4	3
2	7	4	1
6	2	5	2
13	6	8	9

$$2 * 3 + 1 * 4 - 4 * 3 + 3 * 7 + 2 * 4 + 5 * 1$$
$$- 1 * 2 + 8 * 5 + 1 * 2$$
$$= \mathbf{72}$$

Correlation for 2D Image

	44	72	

Template – 3x3

1	3	4	3
2	7	4	1
6	2	5	2
13	6	8	9

Based on the current step, what is the region that best matches the template?

It is the one corresponding to the highest score in the result matrix.

Continue Correlating the Template

	44	72	

1	3	4	3
2	7	4	1
6	2	5	2
13	6	8	9

Template – 3x3

2	1	-4
3	2	5
-1	8	1

Continue Correlating the Template

	44	72	

Template – 3x3

1	3	2	3	-4
2	7	4	2	5
6	2	-1	8	1
13	6	8	9	

This will cause the program performing correlation to fall into index out of bounds exception.

Padding by Zeros

	44	72	

Template – 3x3

1	3	4	3	**0**
2	7	4	1	**0**
6	2	5	2	**0**
13	6	8	9	**0**

Why Zero?

It doesn`t affect multiplication.

Continue Correlating the Template

	44	72	

Template – 3x3

1	3	2	3	0
2	7	3	2	0
6	2	1	8	0
13	6	8	9	0

Continue Correlating the Template

	44	72	

Template – 3x3

1	3	4	3	**0**
2	7	4	**1**	**0**
6	2	5	2	**0**
13	6	8	9	**0**

$$2 * 4 + 1 * 3 - 4 * 0 + 3 * 4 + 2 * 1 + 5 * 0$$
$$- 1 * 5 + 8 * 2 + 1 * 0$$
$$= \mathbf{36}$$

Continue Correlating the Template

	44	72	36

Template – 3x3

1	3	4	3	**0**
2	7	4	**1**	**0**
6	2	5	2	**0**
13	6	8	9	**0**

$$2 * 4 + 1 * 3 - 4 * 0 + 3 * 4 + 2 * 1 + 5 * 0$$
$$- 1 * 5 + 8 * 2 + 1 * 0$$
$$= \mathbf{36}$$

Continue Correlating the Template

	44	72	36

Template – 3x3

2	1	-4
3	2	5
-1	8	1

1	3	4	3
2	7	4	1
6	2	5	2
13	6	8	9

Continue Correlating the Template

	44	72	36

Template – 3x3

1	3	4	3
2	7	4	1
6	2	5	-2
13	8	8	9
	-1	8	1

Pad by Zeros

	44	72	36

Template – 3x3

1	3	4	3
2	7	4	1
6	2	5	2
13	6	8	9
0	**0**	**0**	**0**

Continue Correlating the Template

	44	72	36

Template – 3x3

1	3	4	3
2	7	4	1
6	2	5	-4
13	8	8	9
0	-1	0	0

Continue Correlating the Template

	44	72	36

Template – 3x3

1	3	4	3
2	7	4	1
6	2	5	2
13	6	**8**	9
0	**0**	**0**	**0**

$$2 * 2 + 1 * 5 - 4 * 2 + 3 * 6 + 2 * 8 + 5 * 9$$
$$- 1 * 0 + 8 * 0 + 1 * 0$$
$$= \mathbf{80}$$

Continue Correlating the Template

	44	72	36
		80	

Template – 3x3

1	3	4	3
2	7	4	1
6	2	5	2
13	6	8	9

Continue till end.

How much Padding Required?

- It depends on both the template size.
- For 3x3 template, pad two columns and two rows. One column to the left and another to the right. One row at the top and one row at the bottom.
- For 5x5? Pad 4 rows (two left & two right) and 5 columns (two top & two bottom).
- Generally, for N*N template pad (N-1)/2 columns and rows at each side.

0	0	0	0	0	0
0	1	3	4	3	0
0	2	7	4	1	0
0	6	2	5	2	0
0	13	6	8	9	0
0	0	0	0	0	0

How to do this Programmatically.?

Implementing Correlation in Python – 3x3 Template

```python
import skimage
import numpy
import matplotlib

img = skimage.io.imread("fruits.png")
img = skimage.color.rgb2gray(img)
template = numpy.array([[1, -1, 0], [.1, .6, .8], [0, .9, .3]])

new_img = numpy.zeros((img.shape[0]+2, img.shape[1]+2))
new_img[1:new_img.shape[0]-1, 1:new_img.shape[1]-1] = img
result = numpy.zeros((new_img.shape))

for r in numpy.arange(1, new_img.shape[0]-1):
    for c in numpy.arange(1, new_img.shape[1]-1):
        curr_region = new_img[r-1:r+2, c-1:c+2]
        curr_result = curr_region * template
        score = numpy.sum(curr_result)
        result[r, c] = score

final_img = result[1:result.shape[0]-1, 1:result.shape[1]-1]
matplotlib.pyplot.imshow(final_img).set_cmap("gray")
```

Implementing Correlation in Python – 141x141 Template

```
import skimage.io
import numpy
import matplotlib

img = skimage.io.imread("fruits.png")
img = skimage.color.rgb2gray(img)
template = skimage.io.imread("template22.png")
template = skimage.color.rgb2gray(template)

#141x141 Template
new_img = numpy.zeros((img.shape[0]+140, img.shape[1]+140))
new_img[70:img.shape[0]+70, 70:img.shape[1]+70] = img
result = numpy.zeros((new_img.shape))
```

Implementing Correlation in Python – 141x141 Template

```python
for r in numpy.arange(70, img.shape[0]+70):
    for c in numpy.arange(70, img.shape[1]+70):
        curr_region = new_img[r-70:r+71, c-70:c+71]
        curr_result = curr_region * template
        score = numpy.sum(curr_result)
        result[r, c] = score

result_img = result[70:result.shape[0]-70,
                    70:result.shape[1]-70]

idx = numpy.where(result_img == numpy.max(result_img))

ROI_scores = result_img[idx[0][0]-70:idx[0][0]+70,
                        idx[1][0]-70:idx[1][0]+70]
ROI_img = img[idx[0][0]-70:idx[0][0]+71,
              idx[1][0]-70:idx[1][0]+71]
```

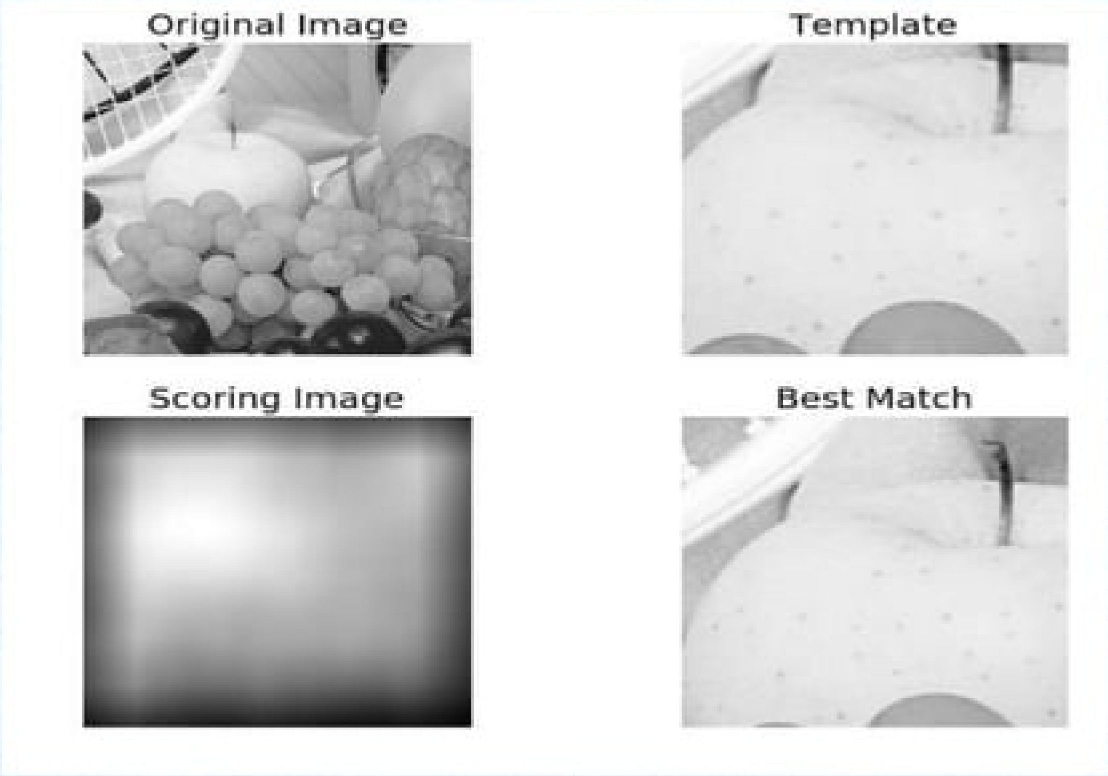
Implementing Correlation in Python – 141x141 Template

```python
fig, ax = matplotlib.pyplot.subplots(nrows=2, ncols=2)
ax[0, 0].imshow(img).set_cmap("gray")
ax[0, 0].set_title("Original Image")
ax[0, 0].axis("off")
ax[0, 1].imshow(template).set_cmap("gray")
ax[0, 1].set_title("Template")
ax[0, 1].axis("off")
ax[1, 0].imshow(result_img).set_cmap("gray")
ax[1, 0].set_title("Scoring Image")
ax[1, 0].axis("off")
ax[1, 1].imshow(ROI_img).set_cmap("gray")
ax[1, 1].set_title("Best Match")
ax[1, 1].axis("off")
matplotlib.pyplot.savefig("res2.png", bbox_inches="tight")
```

Implementing Correlation in Python – 141x141 Template

Is this is the optimal results?

No. But WHY?



Matching using Correlation

- Matching depends on just multiplication.
- The highest results of multiplication is the one selected as best match.
- But the highest results of multiplication not always refers to best match.

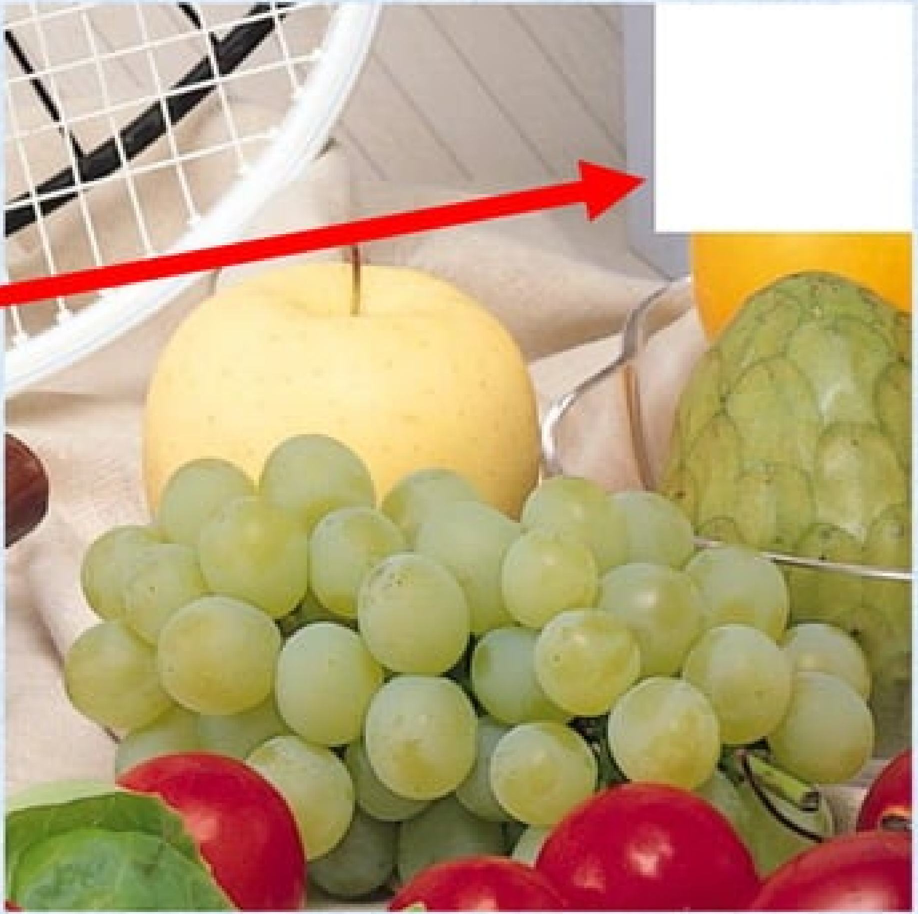
Template

Result

Matching using Correlation

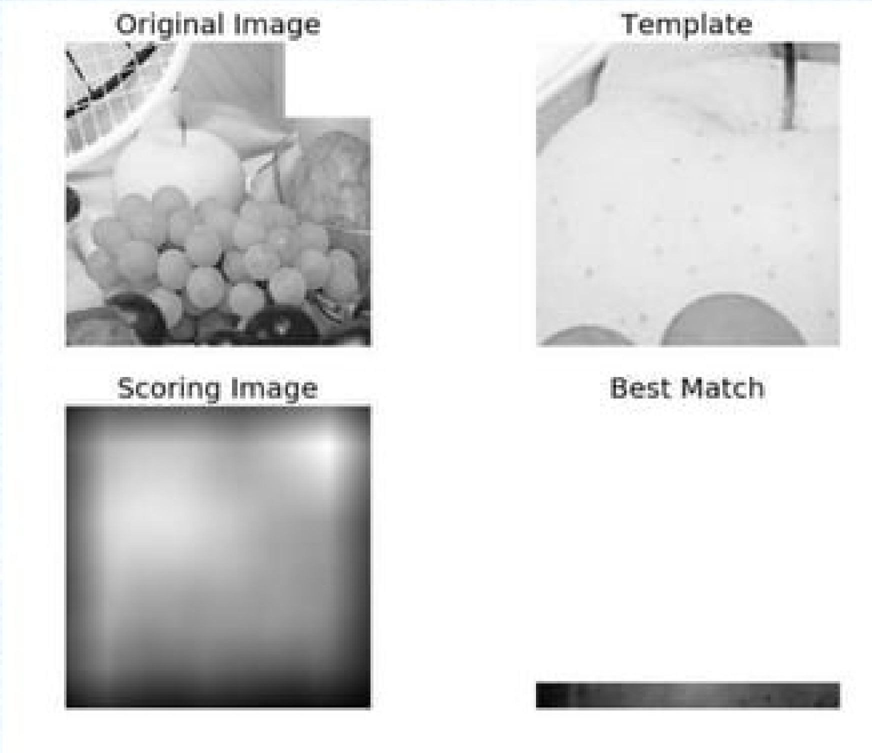
- Make a simple edit on the image by adding a pure white rectangular area.
- Use the previous template.
- Apply the algorithm.

Template



Matching using Correlation

- The best matched region is the white region.
- The reason is that correlation depends on multiplication.
- What gives highest multiplication results is the best match.
- Getting high results corresponds to multiplying by higher numbers.
- The highest pixel value for gray images is 255 for white.



Implementing Correlation in Python – Generic Code – Squared Odd Sized Templates

```
import skimage.io
import numpy
import matplotlib

img = skimage.io.imread("fruits.png")
img = skimage.color.rgb2gray(img)
template = skimage.io.imread("template22.png")
template = skimage.color.rgb2gray(template)
ts = template.shape[0]

new_img = numpy.zeros((img.shape[0]+ts-1,
                       img.shape[1]+ts-1))
new_img[(ts-1)/2:img.shape[0]+(ts-1)/2,
        (ts-1)/2:img.shape[1]+(ts-1)/2] = img
result = numpy.zeros((new_img.shape))
```

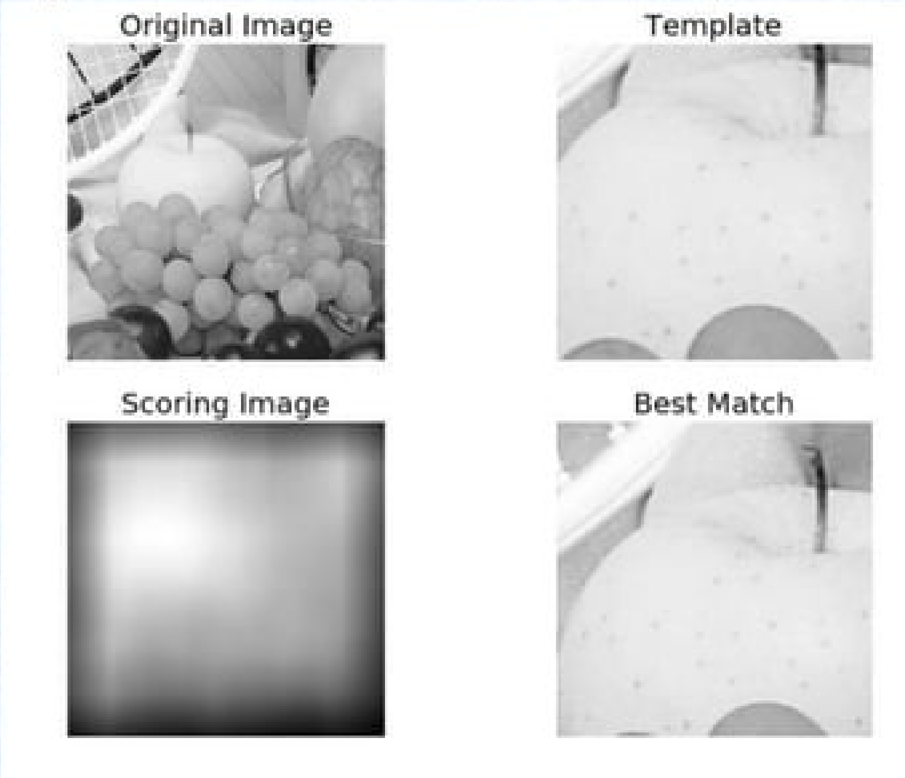
Implementing Correlation in Python – Generic Code – Squared Odd Sized Templates

```
for r in numpy.arange((ts-1)/2, img.shape[0]+(ts-1)/2):
    for c in numpy.arange((ts-1)/2,
                          img.shape[1]+(ts-1)/2):
        curr_region = new_img[r-(ts-1)/2:r+(ts-1)/2+1,
                              c-(ts-1)/2:c+(ts-1)/2+1]
        curr_result = curr_region * template
        score = numpy.sum(curr_result)
        result[r, c] = score

result_img = result[(ts-1)/2:result.shape[0]-(ts-1)/2,
                    (ts-1)/2:result.shape[1]-(ts-1)/2]
```

Implementing Correlation in Python – Generic Code – Squared Odd Sized Templates

```
idx = numpy.where(result_img == numpy.max(result_img))

ROI_scores = result_img[idx[0][0]-(ts-1)/2:idx[0][0]+(ts-1)/2,
                        idx[1][0]-(ts-1)/2:idx[1][0]+(ts-1)/2]
ROI_img = img[idx[0][0]-(ts-1)/2:idx[0][0]+(ts-1)/2+1,
              idx[1][0]-(ts-1)/2:idx[1][0]+(ts-1)/2+1]

fig, ax = matplotlib.pyplot.subplots(nrows=2, ncols=2)
ax[0, 0].imshow(img).set_cmap("gray")
ax[0, 0].set_title("Original Image")
ax[0, 0].axis("off")
ax[0, 1].imshow(template).set_cmap("gray")
ax[0, 1].set_title("Template")
ax[0, 1].axis("off")
ax[1, 0].imshow(result_img).set_cmap("gray")
ax[1, 0].set_title("Scoring Image")
ax[1, 0].axis("off")
ax[1, 1].imshow(ROI_img).set_cmap("gray")
ax[1, 1].set_title("Best Match")
ax[1, 1].axis("off")
matplotlib.pyplot.savefig("res.png", bbox_inches="tight")
```

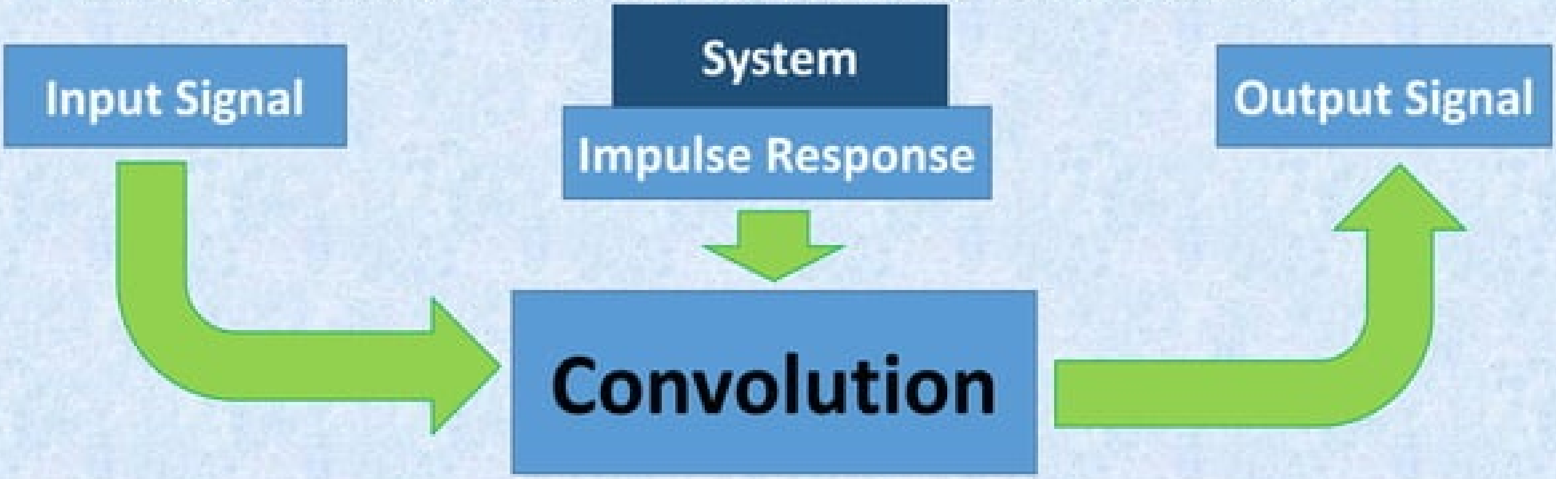
Implementing Correlation in Python – Generic Code – Squared Odd Sized Templates



CONVOLUTION

What is Convolution?

- The primary objective of mathematical convolution is combining two signals to generate a third signal. Convolution is not limited on digital image processing and it is a broad term that works on signals.



Convolution Formula for 2D Digital Signal

- Convolution is applied similarly to correlation.
- Note how similar the formulas for correlation and convolution. The only difference is that in the signs of $\boldsymbol{u}$ and $\boldsymbol{v}$ variables.
- Correlation formula has positive signs for $\boldsymbol{u}$ and $\boldsymbol{v}$ while correlation has negative signs for them.

Is just changing signs make sense?

$$\boldsymbol{G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} h[u,v]F[i - u, j - v]}$$

Correlation Vs. Convolution

Correlation

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} h[u,v]F[i+u, j+v]$$

Convolution

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} h[u,v]F[i-u, j-v]$$

.1	.4	.2
.3	.8	.5
.7	.5	.9

4	20	13
47	80	45
73	93	0

.1	.4	.2
.3	.8	.5
.7	.5	.9

Correlation Vs. Convolution

Correlation

$$G[i,j] = \sum_{u=-k}^{k}\sum_{v=-k}^{k} h[u,v]F[i+u,j+v]$$

Convolution

$$G[i,j] = \sum_{u=-k}^{k}\sum_{v=-k}^{k} h[u,v]F[i-u,j-v]$$

.1	.4	.2
.3	.8	.5
.7	.5	.9

4	20	13
47	80	45
73	93	0

.1	.4	.2
.3	.8	.5
.7	.5	.9

Correlation Vs. Convolution

Correlation

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} h[u,v] F[i+u, j+v]$$

Convolution

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} h[u,v] F[i-u, j-v]$$

.1	.4	.2
.3	.8	.5
.7	.5	.9

4	20	13
47	80	45
73	93	0

.1	.4	.2
.3	.8	.5
.7	.5	.9

Correlation Vs. Convolution

Correlation

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} h[u,v]F[i+u,j+v]$$

Convolution

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} h[u,v]F[i-u,j-v]$$

.1	.4	.2
.3	.8	.5
.7	.5	.9

4	20	13
47	80	45
73	93	0

.1	.4	.2
.3	.8	.5
.7	.5	.9

Correlation Vs. Convolution

Correlation

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} h[u,v]F[i+u, j+v]$$

Convolution

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} h[u,v]F[i-u, j-v]$$

.1	.4	.2
.3	.8	.5
.7	.5	.9

4	20	13
47	80	45
73	93	0

.1	.4	.2
.3	.8	.5
.7	.5	.9

Correlation Vs. Convolution

Correlation

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} h[u,v]F[i+u,j+v]$$

Convolution

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} h[u,v]F[i-u,j-v]$$

.1	.4	.2
.3	.8	.5
.7	.5	.9

4	20	13
47	80	45
73	93	0

.1	.4	.2
.3	.8	.5
.7	.5	.9

Correlation Vs. Convolution

Correlation

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} h[u,v] F[i+u, j+v]$$

Convolution

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} h[u,v] F[i-u, j-v]$$

.1	.4	.2
.3	.8	.5
.7	.5	.9

4	20	13
47	80	45
73	93	0

.1	.4	.2
.3	.8	.5
.7	.5	.9

Correlation Vs. Convolution

Correlation

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} h[u,v]F[i+u,j+v]$$

Convolution

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} h[u,v]F[i-u,j-v]$$

.1	.4	.2
.3	.8	.5
.7	.5	.9

4	20	13
47	80	45
73	93	0

.1	.4	.2
.3	.8	.5
.7	.5	.9

Correlation Vs. Convolution

Correlation

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} h[u,v] F[i+u, j+v]$$

Convolution

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} h[u,v] F[i-u, j-v]$$

.1	.4	.2
.3	.8	.5
.7	.5	.9

4	20	13
47	80	45
73	93	0

.1	.4	.2
.3	.8	.5
.7	.5	.9

Correlation Vs. Convolution

Correlation

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} h[u,v]F[i+u, j+v]$$

Convolution

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} h[u,v]F[i-u, j-v]$$

.1	.4	.2
.3	.8	.5
.7	.5	.9

-1, -1	-1, 0	-1, 1
0, -1	0, 0	0, 1
1, -1	1, 0	1, 1

4	20	13
47	80	45
73	93	0

How to simplify convolution?

.1	.4	.2
.3	.8	.5
.7	.5	.9

1, 1	1, 0	1, -1
0, 1	0, 0	0, -1
-1, 1	-1, 0	-1, -1

Simplifying Convolution

4	20	13
47	80	45
73	93	0

*

.1	.4	.2
.3	.8	.5
.7	.5	.9

1, 1	1, 0	1, -1
0, 1	0, 0	0, -1
-1, 1	-1, 0	-1, -1

=

4	20	13
47	80	45
73	93	0

*

.9	.5	.7
.5	.8	.3
.2	.4	.1

-1, -1	-1, 0	-1, 1
0, -1	0, 0	0, 1
1, -1	1, 0	1, 1

Just rotate the template with 180 degrees before multiplying by the image.

Applying convolution is similar to correlation except for flipping the template before multiplication.

Two Many Multiplications & Additions

- Assume to remove noise from an image, it should be applied to two filters (filter1 and filter2). Each filter has size of 7x7 pixels and the image is of size 1024x1024 pixels.
- A simple approach is to apply filter1 to the image then apply filter2 to the output of filter1.

1024x1024 Image * Filter1 → Temp. Result * Filter2 → Result

- Too many multiplications and additions due to applying two filters on the image. **1,024 x 1,024 x 49 multiplication** and **1,000 x 1,000 x 49** addition for the single filter.

Convolution Associative Property

- Using the associative property of convolution, we can reduce the number of multiplications and additions.
- To apply two filters f1 and f2 over an image, we can merge the two filters together then apply the convolution between the resultant new filter and the image.

$$(\mathrm{Im} * f_1) * f_2 = \mathrm{Im} * (f_1 * f_2)$$

For $f_1 * f_2 = f_3$

Result is $\mathrm{Im} * f_3$

- Finally, there is only 1,024 x 1024 x 49 multiplication and addition.

Convolution Applications

- Convolution is used to merge signals.
- It is used to apply operations like smoothing and filtering images where the primary task is selecting the appropriate filter template or mask.
- Convolution can also be used to find gradients of the image.

Implementing Convolution in Python

- The implementation of convolution is identical to correlation except for the new command that rotates the template.

```
import skimage
import numpy
import matplotlib
import scipy.ndimage

img = skimage.io.imread("fruits.png")
img = skimage.color.rgb2gray(img)
template = numpy.array([[30, -10, 10], [0, 0, .8], [0, 0, 70]])
template = scipy.ndimage.interpolation.rotate(template, 180)
```

GRADIENT

Image Gradients

- Image gradients can be defined as change of intensity in some direction.
- Based on the way gradients were calculated and specifically based on the template/kernel used, gradients can be horizontal (X direction) or vertical (Y direction) in direction.

Horizontal

-1	-1	-1
0	0	0
1	1	1

Vertical

-1	0	1
-1	0	1
-1	0	1

$$\theta = tan^{-1}[\frac{G_y}{G_x}]$$

$$G = \sqrt{G_x + G_y}$$

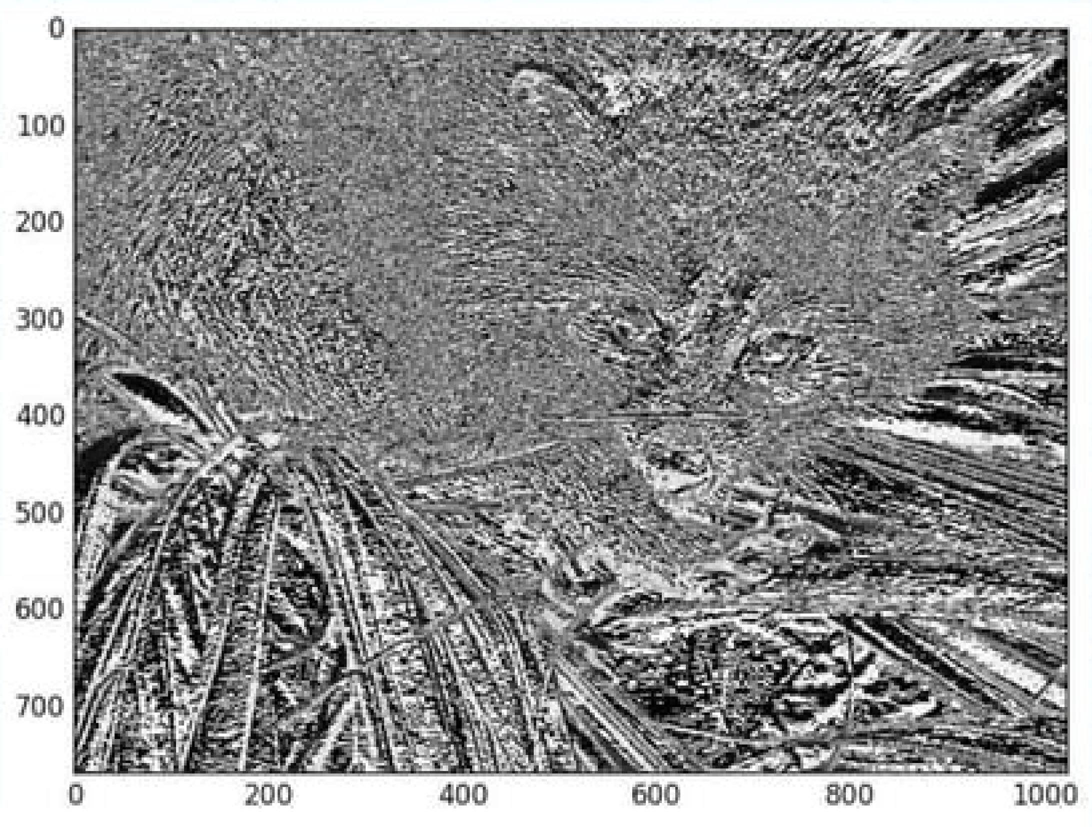
Image Gradient Calculation Steps

- Calculate the image gradient in X direction at each pixel.
- Calculate the image gradient in Y direction at each pixel.
- Calculate the overall gradient for the pixel.

Implementing Horizontal Gradient in Python

```python
import numpy
import matplotlib
import scipy.ndimage
import scipy.misc

img = scipy.misc.face(gray=True)
template = numpy.array([[-1, -1, -1], [0, 0, 0], [1, 1, 1]])

new_img = numpy.zeros((img.shape[0]+2, img.shape[1]+2))
new_img[1:new_img.shape[0]-1, 1:new_img.shape[1]-1] = img
result = numpy.zeros((new_img.shape))

for r in numpy.arange(1, new_img.shape[0]-1):
    for c in numpy.arange(1, new_img.shape[1]-1):
        curr_region = new_img[r-1:r+2, c-1:c+2]
        curr_result = curr_region * template
        score = numpy.sum(curr_result)
        result[r, c] = score

final_img = result[1:result.shape[0]-1, 1:result.shape[1]-1]
final_img = final_img.astype(numpy.uint8)
matplotlib.pyplot.imshow(final_img).set_cmap("gray")
matplotlib.pyplot.savefig("horizontal_gradient.png", bbox_inches="tight"
```

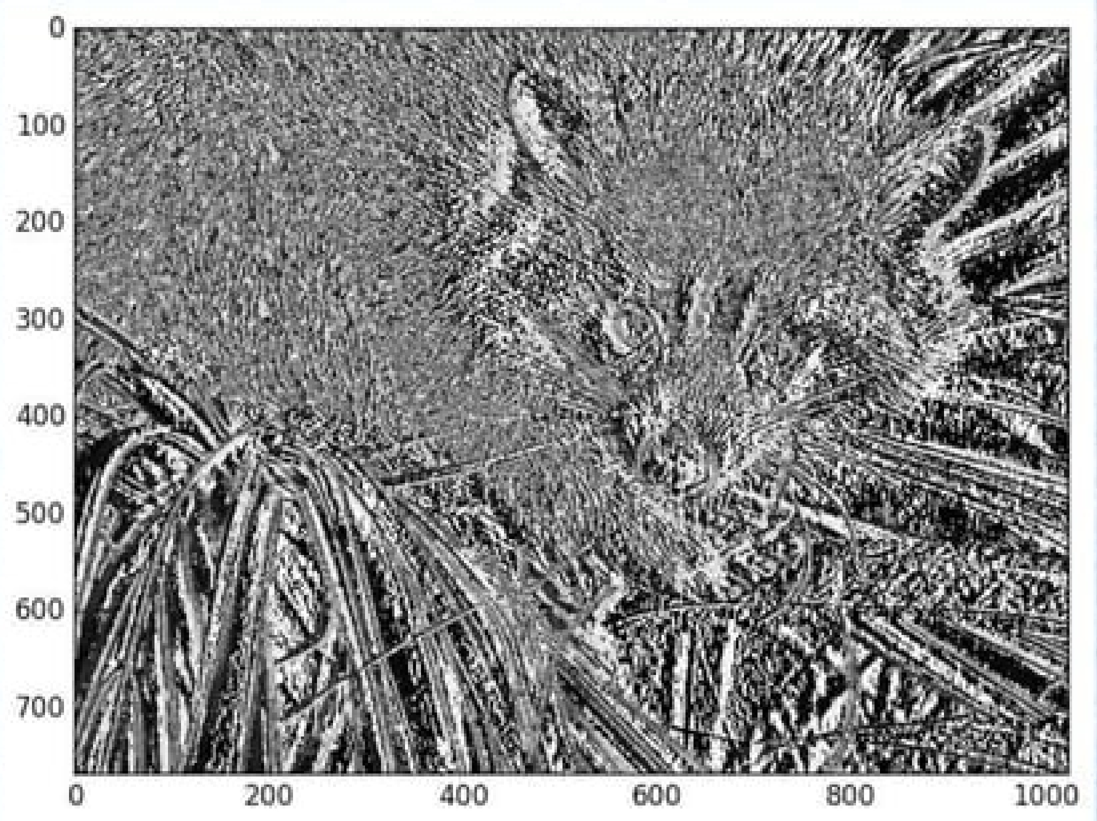
Implementing Horizontal Gradient in Python



Implementing Vertical Gradient in Python

```
import numpy
import matplotlib
import scipy.ndimage
import scipy.misc

img = scipy.misc.face(gray=True)
template = numpy.array([[-1, 0, 1], [-1, 0, 1], [-1, 0, 1]])

new_img = numpy.zeros((img.shape[0]+2, img.shape[1]+2))
new_img[1:new_img.shape[0]-1, 1:new_img.shape[1]-1] = img
result = numpy.zeros((new_img.shape))

for r in numpy.arange(1, new_img.shape[0]-1):
    for c in numpy.arange(1, new_img.shape[1]-1):
        curr_region = new_img[r-1:r+2, c-1:c+2]
        curr_result = curr_region * template
        score = numpy.sum(curr_result)
        result[r, c] = score

final_img = result[1:result.shape[0]-1, 1:result.shape[1]-1]
final_img = final_img.astype(numpy.uint8)
matplotlib.pyplot.imshow(final_img).set_cmap("gray")
matplotlib.pyplot.savefig("vertical_gradient.png", bbox_inches="tight")
```

Implementing Vertical Gradient in Python



FILTERING

Example – Mean Filter

Implement in Python (Deadline next week starting @ 7 Oct. 2017).

1. Mean Smoothing Filter
2. Gaussian Smoothing Filter
3. Sharpening Filter
4. Median Filter

- **Mean Filter Kernel.**

$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$

58	**3**	**213**	81	78
185	**87**	**32**	27	11
70	**66**	**60**	2	19
61	91	129	89	38
14	7	58	14	42

$$\frac{1}{9} * 58 + \frac{1}{9} * 3 + \frac{1}{9} * 213 + \frac{1}{9} * 185 + \frac{1}{9} * 87 + \frac{1}{9} * 32$$

$$+ \frac{1}{9} * 70 + \frac{1}{9} * 66 + \frac{1}{9} * 60$$

= **85.67 ~ 86**