Template Matching with Python and Open CV

- Goal: With a ‘patch’ or section of image, search a larger image or set of data and find the closest match.

- Foundation of recognition.

- Two types to consider:
  - Normalized Correlation
  - Sum of Squared Differences

Sum of Squared Differences

SSD is defined by:

$$S_{tx}(r,\ c) = \sum_{j=0}^{tplRows-1} \sum_{i=0}^{tplCols-1} \left[t(j,\ i) - x\left(r + j - \frac{tplRows}{2},\ c + i - \frac{tplCols}{2}\right)\right]^2$$

Note: This formula is based on “center matching” where the image *x* is searched by template *t* around the center point of the search.

SSD Example: (Corner Matching)

img

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

patch

6	4
4	6

SSD Example

img

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

patch

6	4
4	6

1-6	3-4
2-4	4-6

-5	-1
-2	-2

25	1
4	4

34

34					

SSD Example – Non Normalized

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

34	13	25	42	63	
25	16	20	54	59	
18	0	25	25	23	
33	9	27	27	47	
16	21	26	21	26	

6	4
4	6

1-6	3-4
2-4	4-6

-5	-1
-2	-2

25	1
4	4

34

SSD Example – Normalized

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

.6	.15	.21	.27	.35	
.23	.15	.14	.31	.35	
.17	0	.19	.32	.16	
.39	.08	.18	.20	.40	
.13	.19	.19	.14	.18	

6	4
4	6

1-6	3-4
2-4	4-6

-5	-1
-2	-2

25	1
4	4

34 / 63

Note that the filters find the Upper Left Corner of the Match

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

6	4
4	6

.6	.15	.21	.27	.35	
.23	.15	.14	.31	.35	
.17	0	.19	.32	.16	
.39	.08	.18	.20	.40	
.13	.19	.19	.14	.18	

What about an in-exact match?

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

6	3
4	6

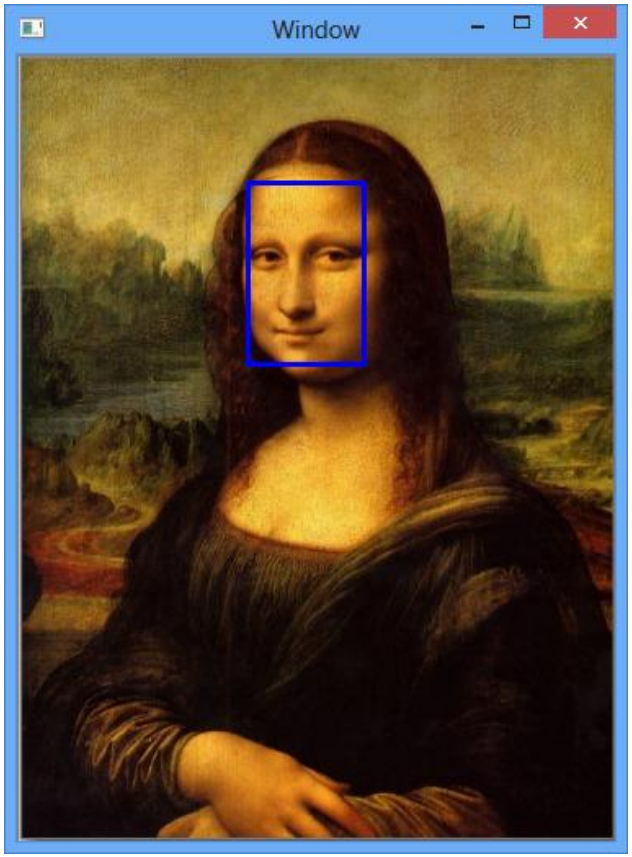
.61	.12	.25	.33	.42	
.25	.20	.21	.38	.42	
.23	.01	.27	.41	.20	
.42	.13	.24	.19	.50	
.14	.26	.27	.18	.18	

Still Works!
Closet Match found

Example with an Image

Source: lisa.png

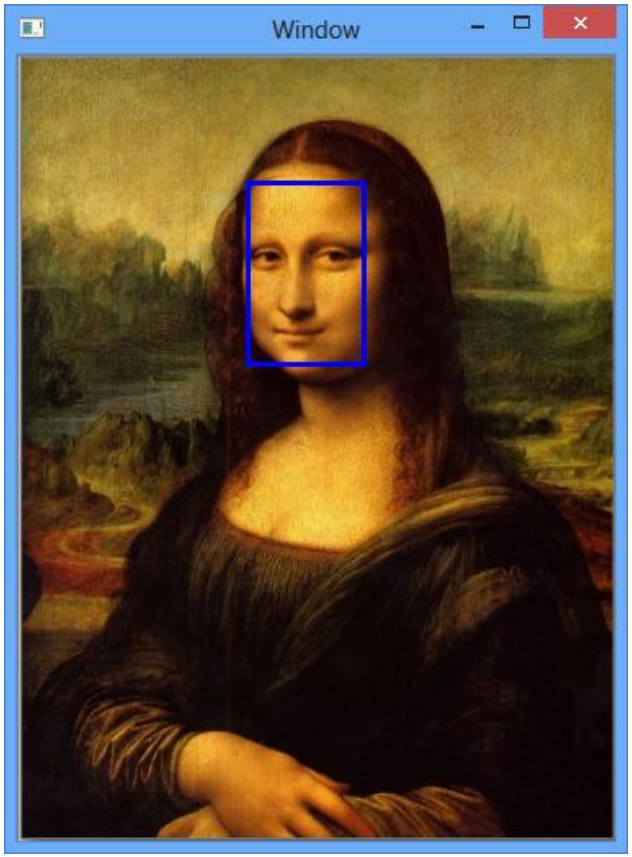
patch



img

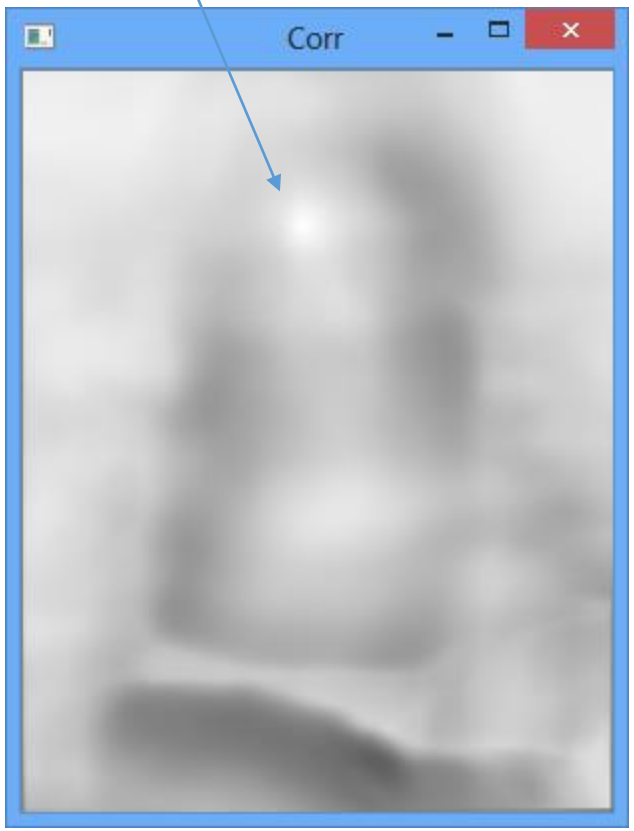
Correlation Map

Brightest area is region of best match.



img

patch

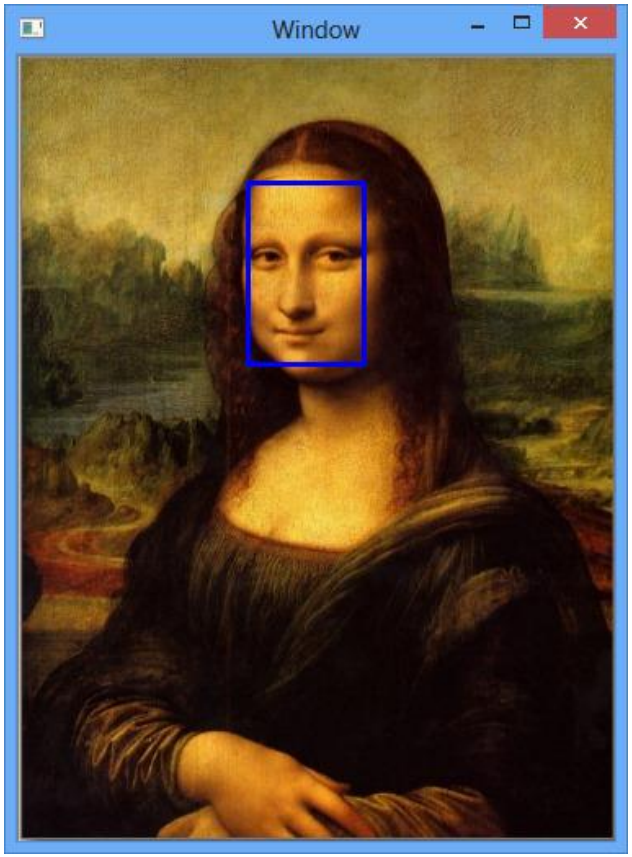


f

(Correlation Map)

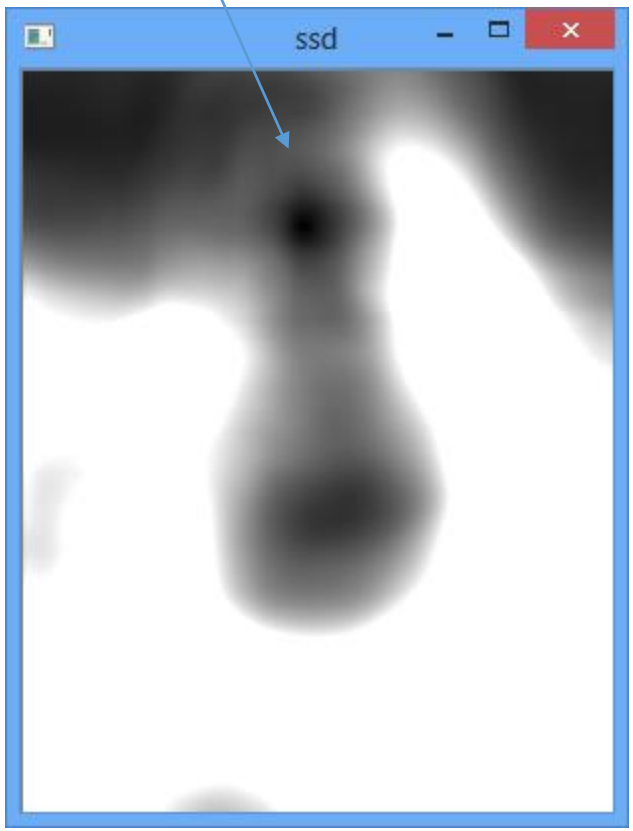
Difference Map

Darkest area is region of best match.



img

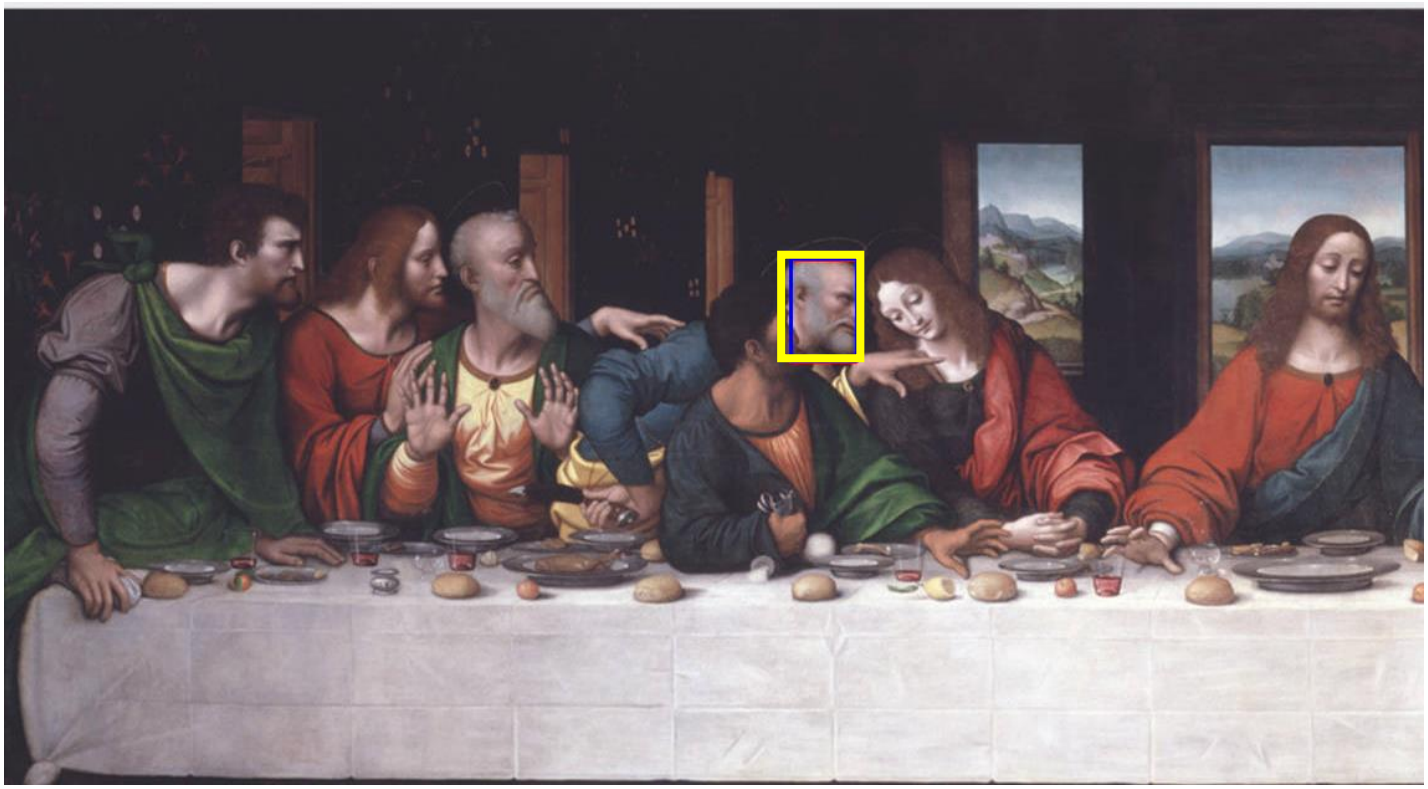
patch

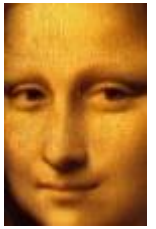


f

(Correlation Map)

Not Exact Match: Last Supper





Note that it did find a face. Correlations and SSD are not perfect, but they provide a starting point for Computer vision

Normalized Correlation

$$R_{tx}(r,\ c) = \sum_{j=0}^{tplRows-1} \sum_{i=0}^{tplCols-1} t(j,\ i) \cdot x\left(r+j-\frac{tplRows}{2},\ c+i-\frac{tplCols}{2}\right)$$

This result is then normalized:

$$\rho_{tx}(r,\ c) = \frac{R_{tx}(r,\ c)}{\sqrt{R_{xx}(r,\ c) R_{tt}\left(\frac{tplRows}{2},\ \frac{tplCols}{2}\right)}}$$

Algorithm: Normalized Correlation

1. Convert image and patch to grayscale
2. Filter image with patch -> f
3. Locate row and column in f with highest value -> point
4. Identify x and y from point object
5. Identify width and height from patch
6. Draw Rectangle at x, y with width and height

Python, CV2, and Numpy commands for Template matching

`f = cv2.matchTemplate(image, patch, MatchType)`

`matchTemplate()` returns a correlation map using the MatchType constant.

- `patch` refers to the search template
- `image`: Image array to be searched or filtered
- (Examples of MatchType)
  - `TM_CCORR_NORMED`
  - `TM_SQDIFF_NORMED`
  - `TM_SQDIFF`

Example:

```
f = cv2.matchTemplate(img, patch, cv2. TM_CCORR_NORMED)
```

Python, CV2, and Numpy commands for Template matching

```
np.where(f == value)
```

np.where() returns a list of points from an array based on the search parameters

```
point = np.where(f == f.max())
y = point[0][0]
x = point[1][0]
```

Code Samples: Sum of Squared Differences

```
# img is the image to search
# patch is the template

# Make Grayscale
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# Create Summed of Squared Differences with patch
ssd = cv2.matchTemplate(gray, patch, cv2.TM_SQDIFF_NORMED)

# find min for match with SQDIFF_NORMED
point = np.where(ssd == ssd.min())
y = point[0][0]
x = point[1][0]
w = len(patch[0])
l = len(patch)

# Draw Rectangle
cv2.rectangle(img, (x, y), (x+w, y+h), (255, 0, 0), 2)
```

Code Samples: Normalized Correlation

```
# img is the image to search
# patch is the template

# Make Grayscale
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# Create Normalized Correlation with Patch
f = cv2.matchTemplate(gray, patch, cv2.TM_CCORR_NORMED)

# Find max for match with CCORR_NORMED
point = np.where(f == f.max())
y = point[0][0]
x = point[1][0]
w = len(patch[0])
h = len(patch)

#Draw Rectangle
cv2.rectangle(img, (x, y), (x+w, y+h), (0, 0, 255), 2)
```