Lecture Notes for

Machine Learning in Python

⟦👨‍🏫👨‍💻🐍👨‍🔬⟧

Professor Eric Larson

Dimensionality Reduction and Images

Class Logistics and Agenda

- **Logistics**:
  - Lab grading…
  - Do **quiz one**!!
  - Coldfront Allocation
  - **Next Time**: Flipped Module
    - Turn in one per team (HTML), please include team member names from canvas
- **Agenda**
  - Common Feature Extraction Methods for Images
  - Begin Town Hall, if time

Class Overview, by topic

Table Data Visualization
Numpy, Pandas, Seaborn
Overviews with some in-depth discussion

Dimension Reduction and Image Processing
Scikit-learn, Scikit Image,
Intuition only, Some mathematics

Linear and Logistic Regression
Numpy, Recreate API for Scikit-learn
Detailed mathematics for simple optimization
intuition for advanced optimization

Neural Networks and Back Prop.
Numpy
Detailed mathematics for NN operations

Wide and Deep Networks | **Convolutional Networks** | **Recurrent Networks**
Keras, Tensorflow
Intuition, Detailed implement.

Ethics in Language Models
ConceptNet
Case studies

Last time...

E_1	E_2
0.749	0.662
0.662	-0.749
λ=268.3	λ=1.57

covariance

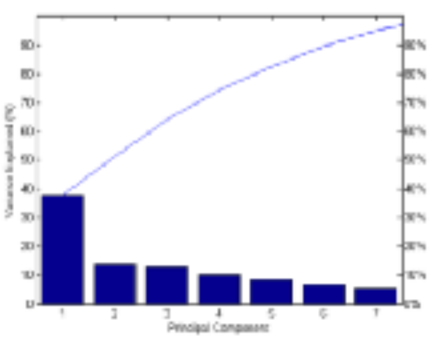
151.5	132.4
132.4	118.3

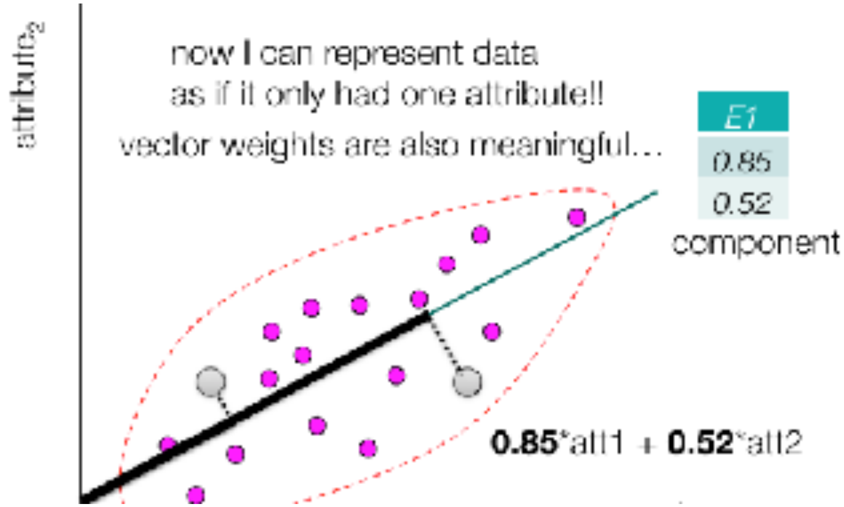
	A_1	A_2
1	14	12.6
2	26	26.6
3	36.3	33.3
4	2.5	3.6
5	15	17.4
6	8	11

	A'_1	A'_2
1	-2.96	-4.82
2	9.03	9.18
3	19.33	15.88
4	-14.46	-13.82
5	-1.96	-0.02
6	-8.96	-6.42

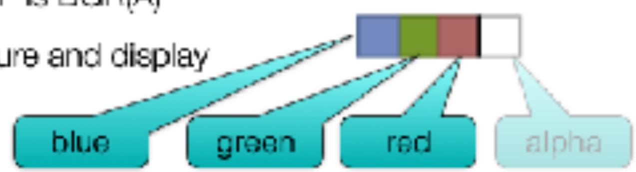
normalize: zero mean
optional: unit std

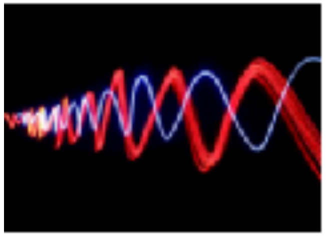
$$r_q = \frac{\sum_{j=1}^{q} \lambda_j}{\sum_{j=1}^{p} \lambda_j}$$

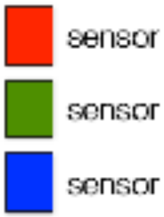


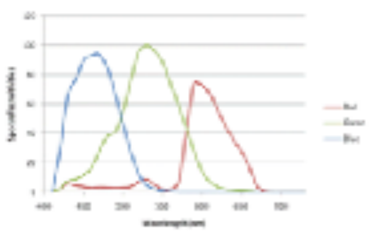


- an image can be represented in many ways
- most common format is a matrix of pixels
  - each "pixel" is BGR(A)
- used for capture and display









Review: Image Representation, Features

Problem: need to represent image as table data

- need a compact representation

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

Review: Image Representation, Features

Problem: need to represent image as table data

- need a compact representation

Solution: row concatenation (also, vectorizing)

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

...

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

Demo

"Refresher" Demo

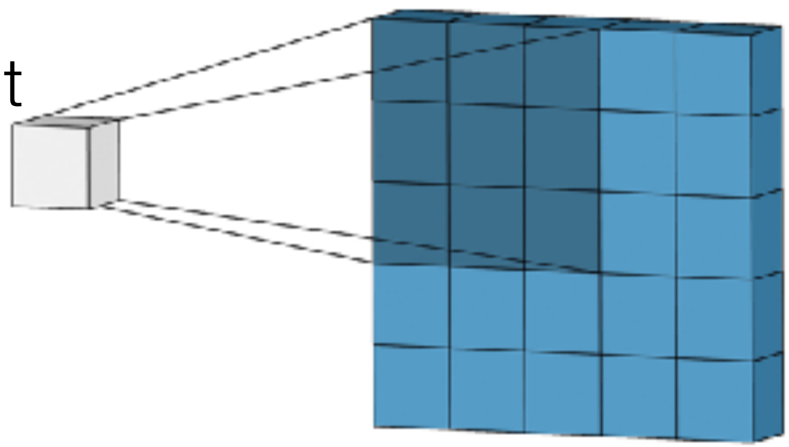
Images Representation in PCA and Randomized PCA

`04.Dimension Reduction and Images.ipynb`

Features of Images

Extracting Features: Convolution

- For images:
  - kernel (matrix of values)
  - slide kernel across image, pixel by pixel
  - multiply and accumulate



This Example:
3x3 Kernel (dark)
Ignoring edges of input
Input Image is 5x5
Output is then 3x3

https://towardsdatascience.com/intuitively-understanding-convolutions-for-deep-learning-1f6f42faee1

Convolution

$$\sum \left(\mathbf{I}\left[i \pm \frac{r}{2}, j \pm \frac{c}{2}\right] \odot \mathbf{k} \right) = \mathbf{O}[i,j]$$

output image at pixel i,j

input image slice centered in i,j with range r x c

kernel of size, r x c usually $r=c$

0	0	0	0	0	0	0	0	0
0	1	2	3	4	12	9	8	0
0	5	2	3	4	12	9	8	0
0	5	2	1	4	10	9	8	0
0	7	2	1	4	12	7	8	0
0	7	2	1	4	14	9	8	0
0	5	2	3	4	12	7	8	0
0	5	2	1	4	12	9	8	0
0	0	0	0	0	0	0	0	0

input image, $\mathbf{I}$

0	0	0
2	3	4
2	3	4

1	2	1
2	4	2
1	2	1

kernel filter, $\mathbf{k}$ 3x3 r x c

20	21	36	…	…	…	…
…	…	…	…	…	…	…
…	…	…	…	…	…	…
…	…	…	…	…	…	…
…	…	…	…	…	…	…
…	…	…	…	…	…	…
…	…	…	…	…	…	…

output image, $\mathbf{O}$

Convolution Examples

0	0	0
1	0	0
0	0	0

Self test:

What does this do?

A. move left pixel to center

B. move right to center

C. blur

Blur

1	1	1
1	1	1
1	1	1

Vertical Edges

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

Sharpen

0	-1	0
-1	5	-1
0	-1	0

Convolution is linear, with $\alpha = 5$

$$\alpha \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} + \left(\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix} \right) = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

α Image + (Image - Blur) = Sharp

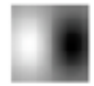
https://setosa.io/ev/image-kernels/

Common operations

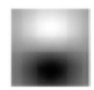
- the gradient (2D derivative)

I

filter

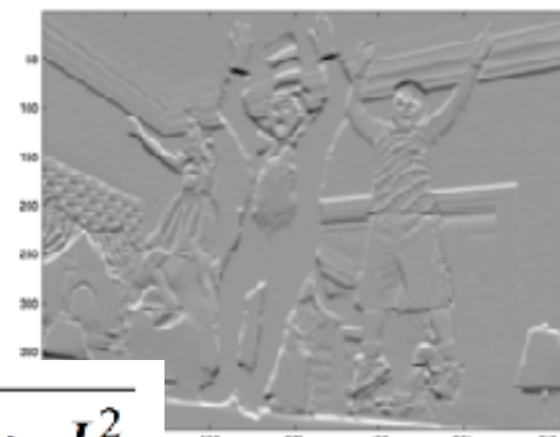


filter



J_x

J_y



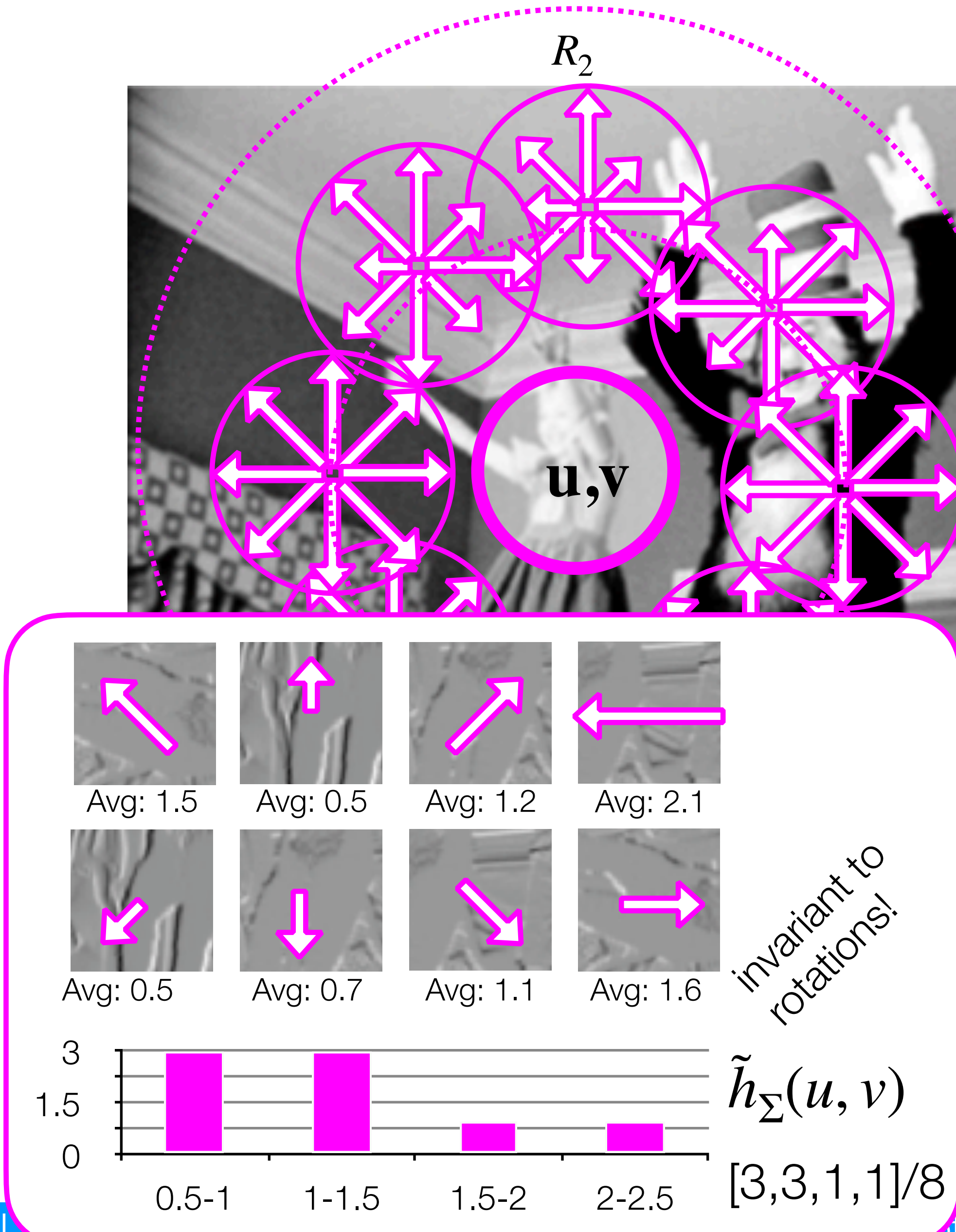
$$\|\nabla J\| = \sqrt{J_x^2 + J_y^2}$$

$$O = \tan^{-1}(J_x / J_y)$$

J_y

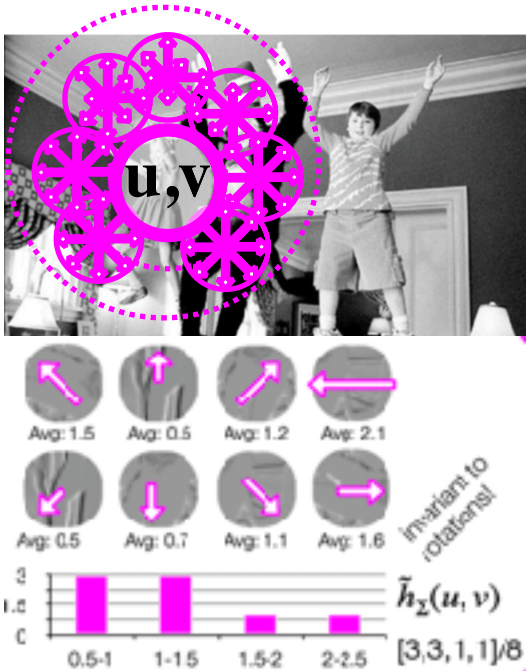
J_x

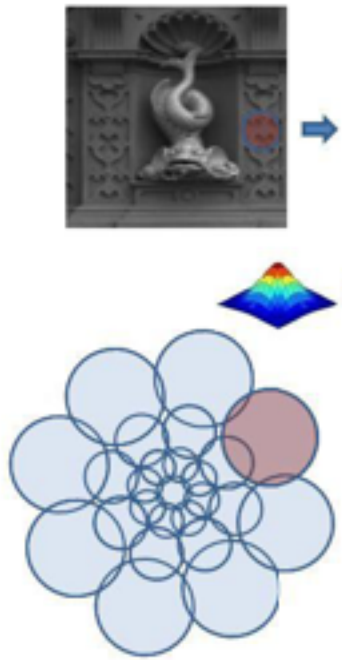
DAISY: same features, regardless of orientation



1. Select u,v pixel location in image and radius
2. Take histogram of average gradient magnitudes in circle for each orientation $\tilde{h}_\Sigma(u, v)$
3. Select circles in a ring, R_1
4. For each circle on the ring, take another histogram $\tilde{h}_\Sigma(\mathbf{l}_O(u, v, R_1))$
5. Repeat for more rings: R_2, R_3
6. Save all histograms as "descriptors" $[\tilde{h}_\Sigma(\,\cdot\,), \tilde{h}_\Sigma(\mathbf{l}_1(\,\cdot\,, R_1)), \tilde{h}_\Sigma(\mathbf{l}_2(\,\cdot\,, R_1)) \ldots \tilde{h}_\Sigma(\mathbf{l}_7(\,\cdot\,, R_2)),\ \tilde{h}_\Sigma(\mathbf{l}_8(\,\cdot\,, R_2))]$
7. Concatenate as "feature" vector at that pixel location

Efficient DAISY, Orient x Circle Radius convolutions





one convolution per orientation

one convolve per ring size

take histogram of convolved images at points u,v

Daisy Operator at u_0,v_0 is Concatenated ||Histograms||

take **normalized** histogram of magnitudes

$$\mathscr{D}(u_0, v_0) =$$

$$[\ \tilde{h}_\Sigma(u_0, v_0), \tilde{h}_\Sigma(\mathbf{l}_1(u_0, v_0, R_1)), \tilde{h}_\Sigma(\mathbf{l}_2(u_0, v_0, R_1)) \ldots$$

$$\tilde{h}_\Sigma(\mathbf{l}_7(u_0, v_0, R_2)),\ \tilde{h}_\Sigma(\mathbf{l}_8(u_0, v_0, R_2))\ ]$$

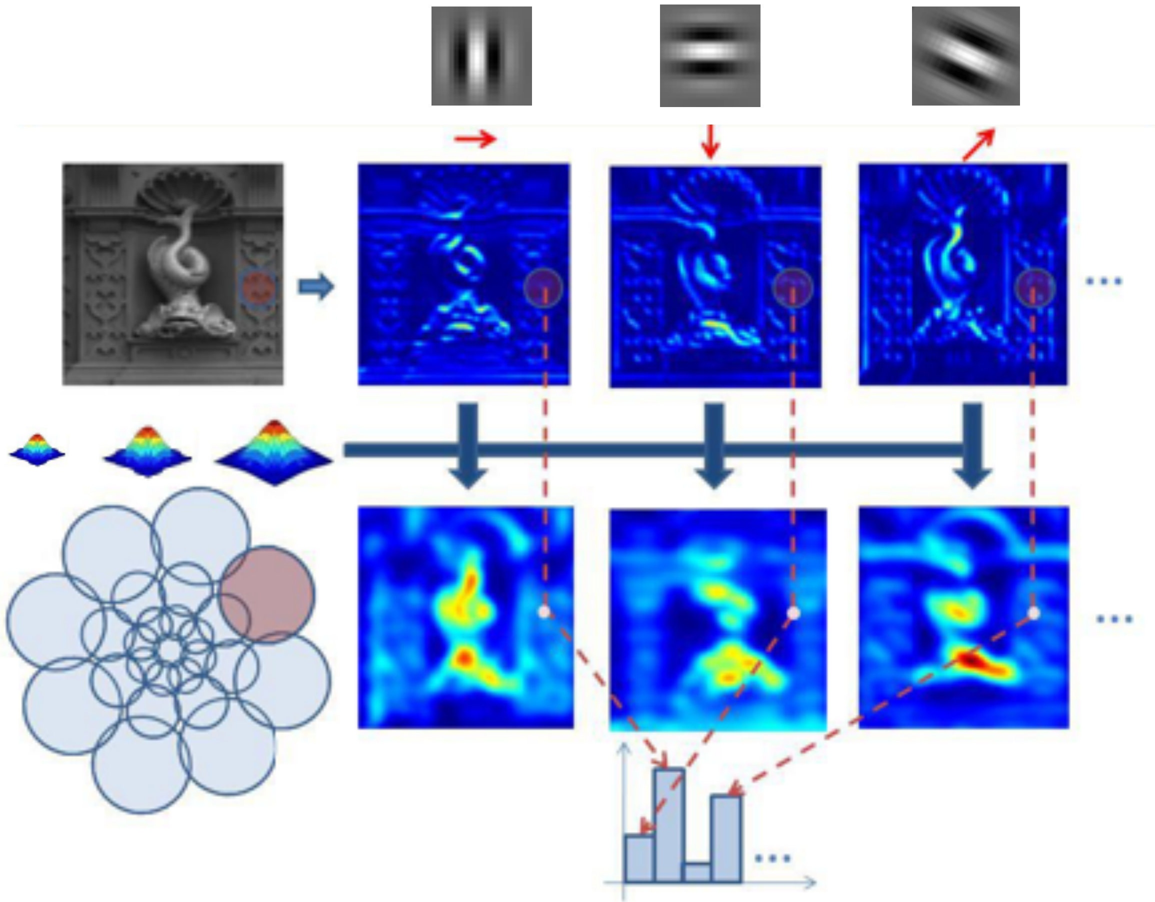
Tola et al. "*Daisy: An efficient dense descriptor applied to wide- baseline stereo*." Pattern Analysis and Machine Intelligence, IEEE

An intuition for the future: DAISY workflow

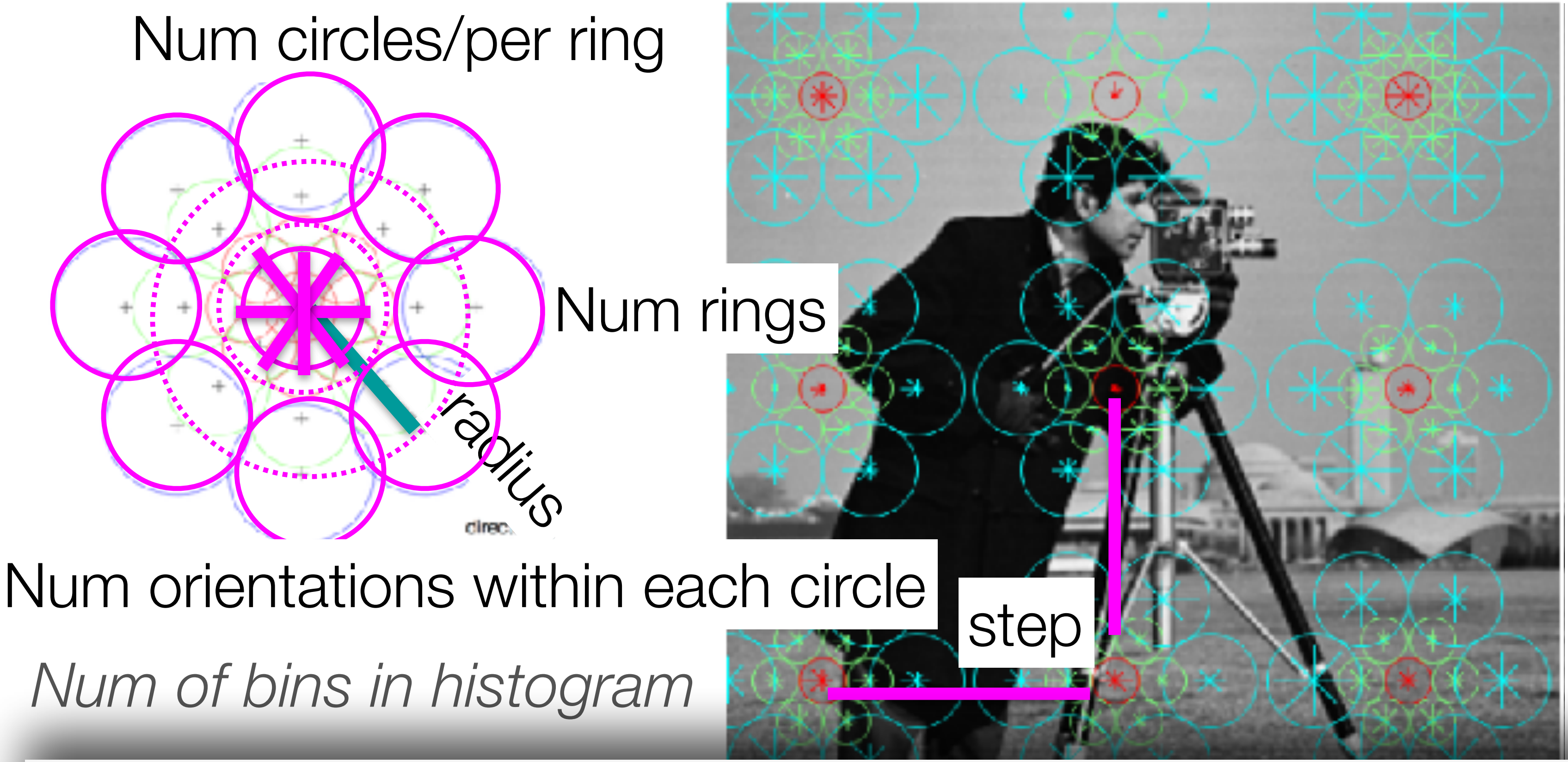
Convolutions on Image

Convolutions of Previous Convolutions

Statistics of Convolution Magnitudes



Hyper Parameters in DAISY, need selection

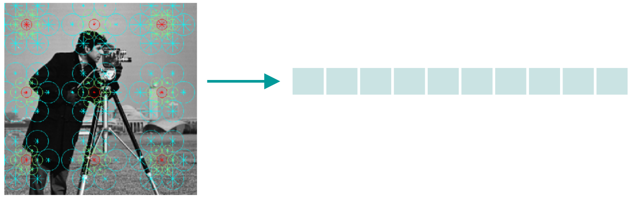



```
daisy(img, step=180, radius=58, rings=2, histograms=6,
      orientations=8, visualize=True)
```

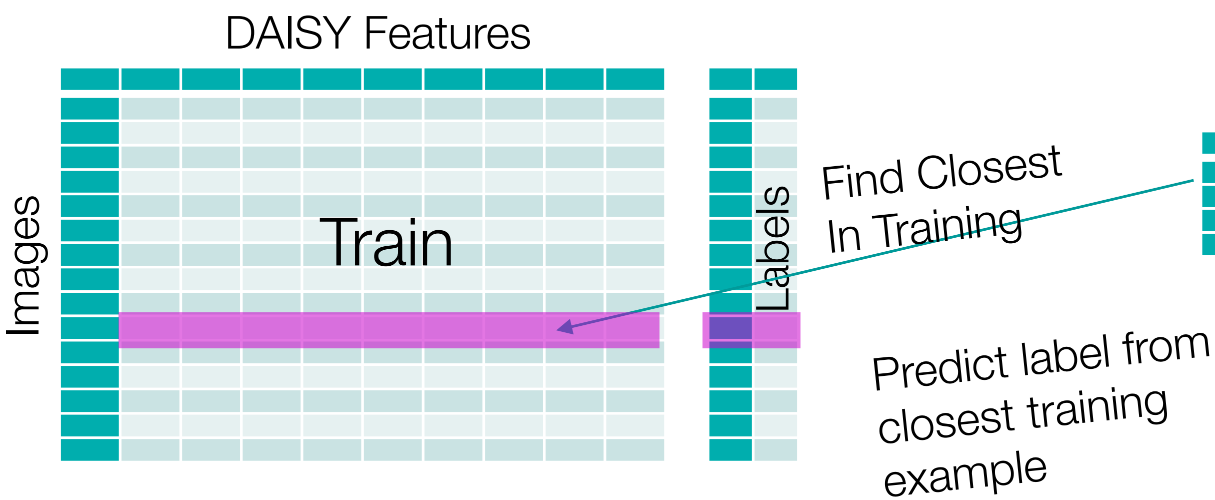
Params step, radius, num rings, num histograms per ring, orientations, *bins per histogram*

Classification with Daisy

- For each image:
  - Calculate daisy matrix (operator values)
  - Flatten into row



- Now we have a Table of Daisy Features (for each image)
- Separate Table into train and test
- Train your favorite classifier
  - Maybe a nearest neighbor classifier



More Image Processing

Demo

Gradients

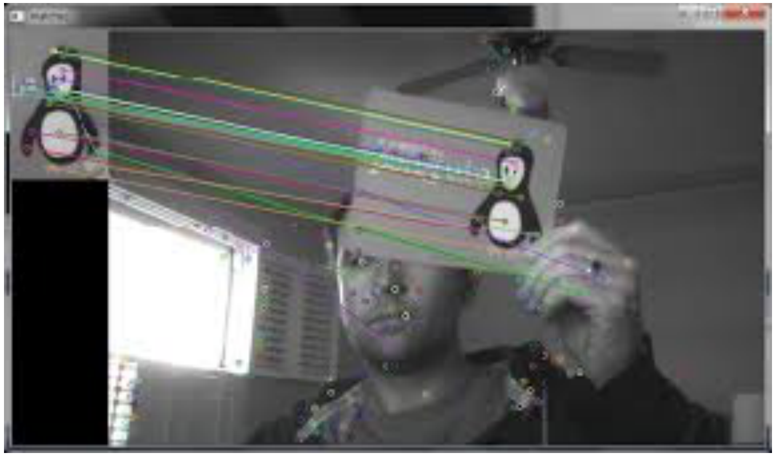
DAISY

Other Tutorials:

http://scikit-image.org/docs/dev/auto_examples/

Matching versus Bag of Features

- Not a difference of vectors, but a percentage of matching points



- SURF, ORB, SIFT, DAISY

Feature Matching

Matching test image to source dataset

1. Choose src image from dataset
2. Take keypoints of src image
3. Take keypoints of test image
4. For each kp in src:
    1. Match with closest kp in test
    2. *How to define match?*
5. Count number of matches between images
6. Determine if src and test are similar based on number of matches
7. Repeat for new src image in dataset
8. Once all images measured, choose best match as the target for the test image

Scikit-image Implementation

match_descriptors

`skimage.feature.match_descriptors(descriptors1, descriptors2, metric=None, p=2, max_distance=inf, cross_check=True, max_ratio=1.0)` [source]

Brute-force matching of descriptors.

For each descriptor in the first set this matcher finds the closest descriptor in the second set (and vice-versa in the case of enabled cross-checking).

Town Hall for Lab 2, Images

- **Quiz is live**: Image Processing!
- **Next Time**: Logistic Regression