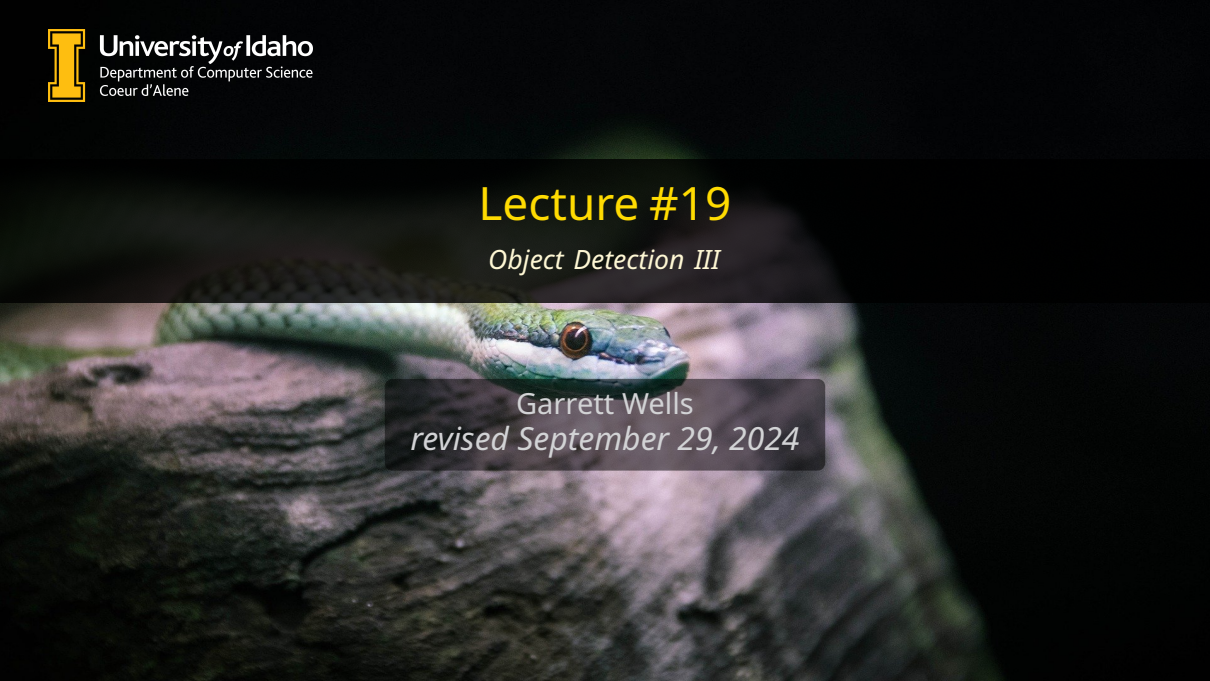
University of Idaho
Department of Computer Science
Coeur d'Alene

Lecture #19

Object Detection III



Garrett Wells

revised September 29, 2024

Table of Contents I

Review

- Introduced Keypoints & Descriptors
  - Keypoints - *trackable features, point, diameter, etc.*
  - Descriptors - *descriptions of the area around a keypoint*
- Several algorithms
  1. SIFT
  2. BRISK
  3. SURF
  4. FAST
  5. BRIEF
  6. ORB

Feature Matching

- Features…
  - corners
  - lines
  - texture
  - data encoded form? ☞ Keypoints & Descriptors
- Match features from two different images
- Feature Matching Algorithms…
  1. Brute Force Matcher
  2. FLANN Matcher
  3. Homography
  4. Image Stitching

Brute Force Matcher

Brute Force Matcher

Compares *descriptor* from set of descriptors found in image *A* and compares to all features from image *B*. Best match is returned.

Comparison method is a “distance” calculation. The calculation method varies by the feature detection method used (SIFT, SURF, *etc.*).

Brute Force Matcher in OpenCV

```
# get features from feature detector (SIFT/ORB, etc)
# create feature matcher object
bfmatcher = cv.BFMatcher(
  # use L2 for SIFT/SURF, use HAMMING for ORB/BRISK/BRIEF
  normType=cv.NORM_L2,
  # if true, run on both sets, make sure matches are same
  crossCheck=false
)

# get list of matches with:
matches = bfmatcher.match(descripors_a, descriptors_b)
cv.drawMatches(...)
```

Brute Force Matcher in OpenCV

KNN Matches…

```
matches = bfmatcher.knnMatch(
  descriptors_a,
  descriptors_b,
  k,              # how many matches to calculate
  masks=noArray(),
  compactResult=false)

cv.drawMatchesKnn(...)
```

Notes on Distance Calculations

`cv.NORM_L2`	Euclidean Distance
`cv.NORM_HAMMING`	Minimum number of substitutions to match two strings

Notes on Cross-Checking

Cross-Check

Calculates matches from set *A* to *B*. Then calculates matches from *B* to *A* (opposite direction). Requires matches to be the same in both directions.

FLANN Matcher

Fast **L**ibrary for **A**pproximate **N**earest **N**eighbors [1]

A collection of different algorithms for Nearest Neighbors(NN) feature matching. Faster than Brute Force Matcher for large datasets.

- Implemented in C++, with bindings for C/MATLAB/Python
  - Documentation on OpenCV side is sparse
  - FLANN GitHub [2], [3]
  - Main Documentation (PDF User Manual): [4]
- Calculates distance ratio between two nearest matches

FLANN Matcher

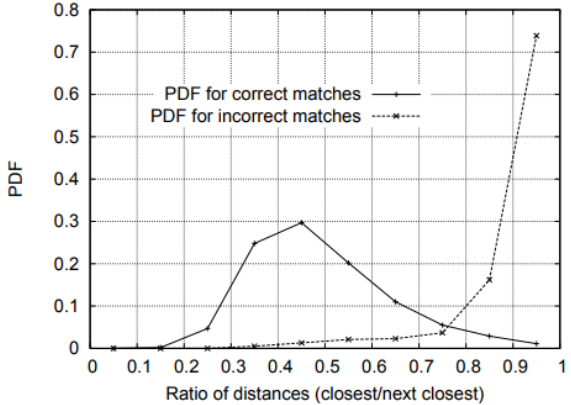


Figure 11: The probability that a match is correct can be determined by taking the ratio of distance from the closest neighbor to the distance of the second closest. Using a database of 40,000 keypoints, the solid line shows the PDF of this ratio for correct matches, while the dotted line is for matches that were incorrect.

FLANN Matcher in OpenCV

```
# select params
index_params = dict(
  algorithm=6,         # which algorithm to use, LSH
  table_number=6,      # number of hash tables to use
  key_size=12,         # length of keys in hash tables
  multiprobe_level=1   # number of probel levels to use
)

search_params = dict(
    checks=50   # how many times to traverse trees in index
)
matcher = cv.FlannBasedMatcher(
  indexParams=index_params,
  searchParams=search_params
)
```

FLANN Matcher in OpenCV

Simpler version…

```
matcher = cv.DescriptorMatcher_create(
                cv.DescriptorMatcher_FLANNBASED
                )
knn_matches = matcher.knnMatch(descriptors1, descriptors2, 2)
```

Image Homography

Homography [5]

A Homography is a transformation(3×3 matrix) that maps points in one image to the corresponding points in another image.

$$\begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = H \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} \tag{1}$$

Image Homography

Notes

- If the points are on different planes(in real world), you will need additional homographies (one per plane) [5]
- Generally need to find features first

Applications

1. Perspective Correction
2. Image Stitching
3. Virtual Billboard [6]

Image Stitching

Image Stitching [7]

The process of combining multiple photographic images with overlapping fields of view to produce a segmented panorama or high-resolution image.

Algorithm

1. Start with 2+ images
2. Find image homography
3. Use homography to “stitch”/sew image together into one image

Image Stitching

see [8]...

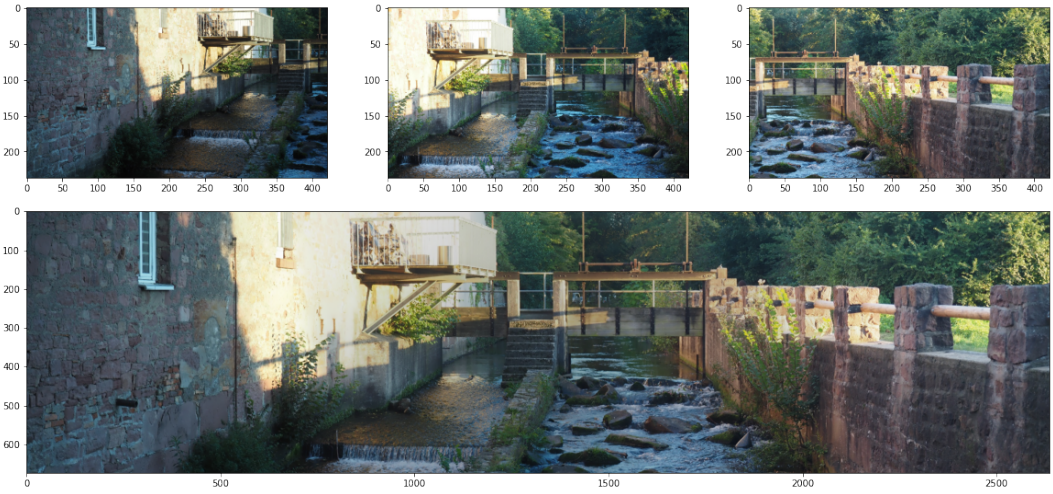


Image Stitching

Image Stitching Examples…

1. Static Image Panorama[9]
2. Video Stream Panorama from Multiple Cameras [10]
3. Color Alignment In Historical Cameras [11]

Haar Cascade Classifiers

Cascade Classifier [12]

A machine learning based approach where a cascade function is trained from positive(image containing class instance) and negative (image missing class instance)images and then used to detect objects in other images.

Haar Cascade Classifiers

Cascade Classifier [12]

A machine learning based approach where a cascade function is trained from positive(image containing class instance) and negative (image missing class instance)images and then used to detect objects in other images.

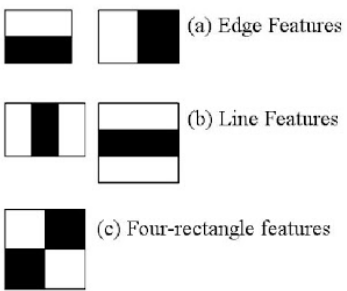


Figure: Haar Features

Haar Cascade Classifiers

Cascade Classifier [12]

A machine learning based approach where a cascade function is trained from positive(image containing class instance) and negative (image missing class instance)images and then used to detect objects in other images.

1. Pass binary feature masks over input image

Haar Cascade Classifiers

Cascade Classifier [12]

A machine learning based approach where a cascade function is trained from positive(image containing class instance) and negative (image missing class instance)images and then used to detect objects in other images.

1. Pass binary feature masks over input image
2. Sum of foreground(white) pixels is subtracted from sum of background(black) pixels

Haar Cascade Classifiers

Cascade Classifier [12]

A machine learning based approach where a cascade function is trained from positive(image containing class instance) and negative (image missing class instance)images and then used to detect objects in other images.

1. Pass binary feature masks over input image
2. Sum of foreground(white) pixels is subtracted from sum of background(black) pixels
3. Feature is output value of calculation.

Haar Cascade Classifiers

Cascade Classifier [12]

A machine learning based approach where a cascade function is trained from positive(image containing class instance) and negative (image missing class instance)images and then used to detect objects in other images.

1. Pass binary feature masks over input image
2. Sum of foreground(white) pixels is subtracted from sum of background(black) pixels
3. Feature is output value of calculation.
4. Repeated for all input images.

Haar Cascade Classifiers

Cascade Classifier [12]

A machine learning based approach where a cascade function is trained from positive(image containing class instance) and negative (image missing class instance)images and then used to detect objects in other images.

1. Pass binary feature masks over input image
2. Sum of foreground(white) pixels is subtracted from sum of background(black) pixels
3. Feature is output value of calculation.
4. Repeated for all input images.
5. Features all given weights.

Haar Cascade Classifiers

Cascade Classifier [12]

A machine learning based approach where a cascade function is trained from positive(image containing class instance) and negative (image missing class instance)images and then used to detect objects in other images.

1. Pass binary feature masks over input image
2. Sum of foreground(white) pixels is subtracted from sum of background(black) pixels
3. Feature is output value of calculation.
4. Repeated for all input images.
5. Features all given weights.
6. Apply threshold to classify images.

Haar Cascade Classifiers

Cascade Classifier [12]

A machine learning based approach where a cascade function is trained from positive(image containing class instance) and negative (image missing class instance)images and then used to detect objects in other images.

1. Pass binary feature masks over input image
2. Sum of foreground(white) pixels is subtracted from sum of background(black) pixels
3. Feature is output value of calculation.
4. Repeated for all input images.
5. Features all given weights.
6. Apply threshold to classify images.
7. If class is wrong, weights of features in image are increased. Error rate for each feature updated.

Haar Cascade Classifiers

Cascade Classifier [12]

A machine learning based approach where a cascade function is trained from positive(image containing class instance) and negative (image missing class instance)images and then used to detect objects in other images.

1. Pass binary feature masks over input image
2. Sum of foreground(white) pixels is subtracted from sum of background(black) pixels
3. Feature is output value of calculation.
4. Repeated for all input images.
5. Features all given weights.
6. Apply threshold to classify images.
7. If class is wrong, weights of features in image are increased. Error rate for each feature updated.
8. Set of features with minimum error rate are selected for next prediction.

Haar Cascade Classifiers

Cascade Classifier [12]

A machine learning based approach where a cascade function is trained from positive(image containing class instance) and negative (image missing class instance)images and then used to detect objects in other images.

1. Pass binary feature masks over input image
2. Sum of foreground(white) pixels is subtracted from sum of background(black) pixels
3. Feature is output value of calculation.
4. Repeated for all input images.
5. Features all given weights.
6. Apply threshold to classify images.
7. If class is wrong, weights of features in image are increased. Error rate for each feature updated.
8. Set of features with minimum error rate are selected for next prediction.
9. Algorithm continues until (a) accuracy rate is reached or (b) the required

Cascade Classifiers

BUT... images often only have a small region containing the target.

1. Instead of applying features to whole image, group features into “stages”.
2. Look at image in “windows”, regions that could contain target.
3. Apply first stage(subgroup) of features to the window.
4. If window passes(positive for class instance), apply next stage of features to test. Otherwise, move to next window.

Author’s Architecture [12]

6000+ features, 38 stages, with first five stages containing 1, 10, 25, 25, and 50 features.

make your own [13]

Bibliography I

[1] “Feature matching — OpenCV-python tutorials 1 documentation,” (), [Online]. Available: https://opencv24-python-tutorials.readthedocs.io/en/stable/py_tutorials/py_feature2d/py_matcher/py_matcher.html (visited on 09/29/2024).

[2] *Flann-lib/flann*, original-date: 2011-03-28T18:45:55Z, Sep. 27, 2024. [Online]. Available: https://github.com/flann-lib/flann (visited on 09/29/2024).

[3] “FLANN documentation,” (), [Online]. Available: https://chpatrick.github.io/flann/index.html#document-intro (visited on 09/29/2024).

[4] “FLANN - fast library for approximate nearest neighbors | FLANN / FLANN,” (), [Online]. Available: https://www.cs.ubc.ca/research/flann/ (visited on 09/29/2024).

Bibliography II

[5] "Homography examples using OpenCV (python / c ++) |," (Jan. 4, 2016), [Online]. Available: https://learnopencv.com/homography-examples-using-opencv-python-c/ (visited on 09/29/2024).

[6] "Python OpenCV: Object tracking using homography," GeeksforGeeks, Section: Python. (Jul. 20, 2020), [Online]. Available: https://www.geeksforgeeks.org/python-opencv-object-tracking-using-homography/ (visited on 09/29/2024).

[7] *Image stitching*, in *Wikipedia*, Page Version ID: 1243737350, Sep. 3, 2024. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Image_stitching&oldid=1243737350 (visited on 09/29/2024).

Bibliography III

[8] "GitHub - OpenStitching/stitching: A python package for fast and robust image stitching," (), [Online]. Available: https://github.com/OpenStitching/stitching?tab=readme-ov-file (visited on 09/29/2024).

[9] A. Rosebrock, "Image stitching with OpenCV and python," PyImageSearch, (Dec. 17, 2018), [Online]. Available: https://pyimagesearch.com/2018/12/17/image-stitching-with-opencv-and-python/ (visited on 09/29/2024).

[10] A. Rosebrock, "Real-time panorama and image stitching with OpenCV," PyImageSearch, (Jan. 25, 2016), [Online]. Available: https://pyimagesearch.com/2016/01/25/real-time-panorama-and-image-stitching-with-opencv/ (visited on 09/29/2024).

Bibliography IV

[11] “Image alignment (ECC) in OpenCV (c++ / python) | LearnOpenCV #,” (Jul. 2, 2015), [Online]. Available: https://learnopencv.com/image-alignment-ecc-in-opencv-c-python/ (visited on 09/29/2024).

[12] “OpenCV: Cascade classifier,” (), [Online]. Available: https://docs.opencv.org/4.x/db/d28/tutorial_cascade_classifier.html (visited on 09/29/2024).

[13] “Python programming tutorials,” (), [Online]. Available: https://pythonprogramming.net/haar-cascade-object-detection-python-opencv-tutorial/ (visited on 09/30/2024).