University of Idaho
Department of Computer Science
Coeur d'Alene

Lecture #18

Object Detection II

Garrett Wells

revised September 29, 2024

Table of Contents I

Review

1. Features
2. Object Detection
   - Histograms
   - Image Moments
   - Hough Transforms
   - Optical Flow, Sparse & Dense
   - Harris Corners

Shi-Tomasi Corner Detection

- Remember...
  - Harris Corners
  - Corners maximize gradient in all directions
- Modification: $min(\lambda_1, \lambda_2)$ rather than using Harris scoring equation $(\lambda_1\lambda_2 - k(\lambda_1 + \lambda_2)^2)$) [1]

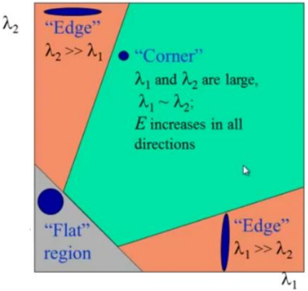


Figure: Harris Space

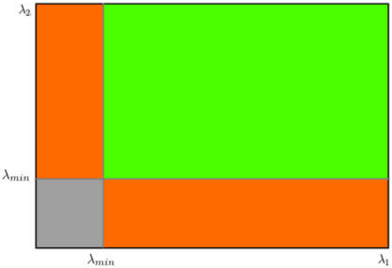


Figure: Shi-Tomasi Space

Shi-Tomasi Corner Detection in OpenCV

```
# Uses Shi-Tomasi function
corners = cv.goodFeaturesToTrack(
      image,                     # 8-bit/fp-32, single channel
      maxCorners,                # max corners returned
      qualityLevel,              # min quality, [0,1] for corner
      minDistance,               # min dist between corners
      mask = noArray(),          # region of interest
      blockSize = 3,             # size of pixel neighborhood
      useHarrisDetector=false,   # use Harris corner detector
      k=0.04)                    # free param for Harris
```

Keypoints

Keypoints

Points that are interesting in an image.

Components:

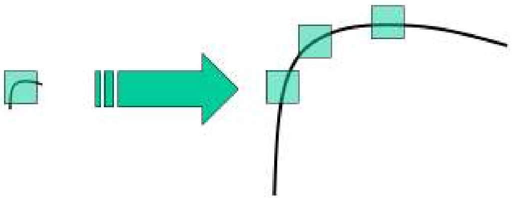
1. (x, y) coordinate
2. diameter
3. orientation
4. strength
5. octave
6. object ID

Algorithms That Use Keypoints

1. SIFT
2. BRISK
3. SURF
4. FAST
5. BRIEF
6. ORB

SIFT

- Scale Invariant Transform [2]
- Harris Corners are rotation invariant, not scale invariant
- This is a problem for corners



SIFT

1. Scale-space Extrema Detection
2. Keypoint Localization
3. Orientation Assignment
4. Keypoint Descriptor
5. Keypoint Matching

SIFT

1. Scale-space Extrema Detection
   - *Search image at various scales for corners, local maxima and potential keypoint*
2. Keypoint Localization

3. Orientation Assignment

4. Keypoint Descriptor

5. Keypoint Matching

SIFT

1. Scale-space Extrema Detection
   - *Search image at various scales for corners, local maxima and potential keypoint*
2. Keypoint Localization
   - *Taylor series expansion to increase extrema location accuracy*

3. Orientation Assignment

4. Keypoint Descriptor

5. Keypoint Matching

SIFT

1. Scale-space Extrema Detection
   - *Search image at various scales for corners, local maxima and potential keypoint*
2. Keypoint Localization
   - *Taylor series expansion to increase extrema location accuracy*
   - *If intensity at extrema is less than threshold(0.33), reject*
3. Orientation Assignment
4. Keypoint Descriptor
5. Keypoint Matching

SIFT

1. Scale-space Extrema Detection
   - *Search image at various scales for corners, local maxima and potential keypoint*
2. Keypoint Localization
   - *Taylor series expansion to increase extrema location accuracy*
   - *If intensity at extrema is less than threshold(0.33), reject*
3. Orientation Assignment
   - *Create orientation histogram using pixel neighborhood*

4. Keypoint Descriptor

5. Keypoint Matching

SIFT

1. Scale-space Extrema Detection
   - *Search image at various scales for corners, local maxima and potential keypoint*
2. Keypoint Localization
   - *Taylor series expansion to increase extrema location accuracy*
   - *If intensity at extrema is less than threshold(0.33), reject*
3. Orientation Assignment
   - *Create orientation histogram using pixel neighborhood*
   - *Weight results to calculate outcome*
4. Keypoint Descriptor
5. Keypoint Matching

SIFT

1. Scale-space Extrema Detection
   - *Search image at various scales for corners, local maxima and potential keypoint*
2. Keypoint Localization
   - *Taylor series expansion to increase extrema location accuracy*
   - *If intensity at extrema is less than threshold(0.33), reject*
3. Orientation Assignment
   - *Create orientation histogram using pixel neighborhood*
   - *Weight results to calculate outcome*
4. Keypoint Descriptor
   - *Create keypoint descriptor*
5. Keypoint Matching

SIFT

1. Scale-space Extrema Detection
   - *Search image at various scales for corners, local maxima and potential keypoint*
2. Keypoint Localization
   - *Taylor series expansion to increase extrema location accuracy*
   - *If intensity at extrema is less than threshold(0.33), reject*
3. Orientation Assignment
   - *Create orientation histogram using pixel neighborhood*
   - *Weight results to calculate outcome*
4. Keypoint Descriptor
   - *Create keypoint descriptor*
   - Take 16×16 neighborhood
5. Keypoint Matching

SIFT

1. Scale-space Extrema Detection
   - *Search image at various scales for corners, local maxima and potential keypoint*
2. Keypoint Localization
   - *Taylor series expansion to increase extrema location accuracy*
   - *If intensity at extrema is less than threshold(0.33), reject*
3. Orientation Assignment
   - *Create orientation histogram using pixel neighborhood*
   - *Weight results to calculate outcome*
4. Keypoint Descriptor
   - *Create keypoint descriptor*
   - Take 16×16 neighborhood
   - Subdivided in to 4×4 sub-blocks
5. Keypoint Matching

SIFT

1. Scale-space Extrema Detection
   - *Search image at various scales for corners, local maxima and potential keypoint*
2. Keypoint Localization
   - *Taylor series expansion to increase extrema location accuracy*
   - *If intensity at extrema is less than threshold(0.33), reject*
3. Orientation Assignment
   - *Create orientation histogram using pixel neighborhood*
   - *Weight results to calculate outcome*
4. Keypoint Descriptor
   - *Create keypoint descriptor*
   - Take 16×16 neighborhood
   - Subdivided in to 4×4 sub-blocks
   - *8 bin orientation histograms created*

5. Keypoint Matching

SIFT

1. Scale-space Extrema Detection
   - *Search image at various scales for corners, local maxima and potential keypoint*
2. Keypoint Localization
   - *Taylor series expansion to increase extrema location accuracy*
   - *If intensity at extrema is less than threshold(0.33), reject*
3. Orientation Assignment
   - *Create orientation histogram using pixel neighborhood*
   - *Weight results to calculate outcome*
4. Keypoint Descriptor
   - *Create keypoint descriptor*
   - Take 16×16 neighborhood
   - Subdivided in to 4×4 sub-blocks
   - *8 bin orientation histograms created*
   - 128 bin values $\rightarrow$ vector to form keypoint descriptor
5. Keypoint Matching

SIFT

1. Scale-space Extrema Detection
   - *Search image at various scales for corners, local maxima and potential keypoint*
2. Keypoint Localization
   - *Taylor series expansion to increase extrema location accuracy*
   - *If intensity at extrema is less than threshold(0.33), reject*
3. Orientation Assignment
   - *Create orientation histogram using pixel neighborhood*
   - *Weight results to calculate outcome*
4. Keypoint Descriptor
   - *Create keypoint descriptor*
   - Take 16×16 neighborhood
   - Subdivided in to 4×4 sub-blocks
   - *8 bin orientation histograms created*
   - 128 bin values $\rightarrow$ vector to form keypoint descriptor
5. Keypoint Matching
   - *Keypoints matched through nearest neighbors*

SIFT

1. Scale-space Extrema Detection
   - *Search image at various scales for corners, local maxima and potential keypoint*
2. Keypoint Localization
   - *Taylor series expansion to increase extrema location accuracy*
   - *If intensity at extrema is less than threshold(0.33), reject*
3. Orientation Assignment
   - *Create orientation histogram using pixel neighborhood*
   - *Weight results to calculate outcome*
4. Keypoint Descriptor
   - *Create keypoint descriptor*
   - Take 16×16 neighborhood
   - Subdivided in to 4×4 sub-blocks
   - *8 bin orientation histograms created*
   - 128 bin values $\rightarrow$ vector to form keypoint descriptor
5. Keypoint Matching
   - *Keypoints matched through nearest neighbors*
   - *Takes ratio of closest and second closest match, rejects based on threshold*

SIFT

1. Scale-space Extrema Detection
    - *Search image at various scales for corners, local maxima and potential keypoint*
2. Keypoint Localization
    - *Taylor series expansion to increase extrema location accuracy*
    - *If intensity at extrema is less than threshold(0.33), reject*
3. Orientation Assignment
    - *Create orientation histogram using pixel neighborhood*
    - *Weight results to calculate outcome*
4. Keypoint Descriptor
    - *Create keypoint descriptor*
    - Take 16×16 neighborhood
    - Subdivided in to 4×4 sub-blocks
    - *8 bin orientation histograms created*
    - 128 bin values → vector to form keypoint descriptor
5. Keypoint Matching
    - *Keypoints matched through nearest neighbors*
    - *Takes ratio of closest and second closest match, rejects based on threshold*
    - Paper claims approx. 90% of false matches discarded, while only discarding 5% of correct matches

SIFT

Notes...

- SIFT is a patented algorithm
- Thus, some OpenCV versions dropped support
- OpenCV 4.10 contains SIFT in main packages

SIFT in OpenCV

```
# create sift object
sift = cv.SIFT_create()
# pass input image, mask
kp = sift.detect(img, None)
# draw output
img = cv.drawKeypoints(still[1], kp, outImage=None)
```

BRISK

- Open source alternative to SIFT [3]
- Uses concentric rings around a center point
- Multi-scale and non-maximum suppression
- Based on AGAST detector

BRISK in OpenCV

```
# create BRISK object
brisk = cv.BRISK_create()
# gather keypoints
kp = brisk.detect(img, None)
# draw keypoints
img = cv.drawKeypoints(still[1], kp, outImage=None)
```

SURF

- Speeded-Up Robust Features [4]
- Faster than SIFT (3x)
- Approximates Laplacian of Gaussian with a Box Filter
- Good for:
  1. blurring changes
  2. rotation changes
- Not great with:
  1. viewpoint changes
  2. illumination changes

SURF

Notes. . .

- SURF also patented, thus not included in OpenCV
- may be able to install in extra module

FAST

- Features from Accelerated Segment Test [5]
- Proposed to accelerate corner detection
- by Edward Rosten & Tom Drummond

FAST

From [5]

1. For given image, select pixel p with intensity I_p
2. Apply threshold value t
3. Consider circle of 16 pixels surrounding p
4. p is a corner pixel if n contiguous pixels in the circle are brighter than $I_p + t$ or darker than $I_p - t$
5. **Enhancement:** A high-speed test is applied → check a few pixels to determine existence of corner

Weaknesses of High-Speed Test

- For $n < 12$, may not reject properly
- Efficiency varies based on order in which they are checked & the distribution of corner types
- Multiple features adjacent to each other are detected (solve with non-maxima suppression)

FAST in OpenCV

```
# create FAST object
fast = cv.FastFeatureDetector_create()
# detect keypoints
kp  = fast.detect(img, None)
# draw keypoints
img = cv.drawKeypoints(still[1], kp, outImage=None)
```

Blob Detection

Binary **L**arge **Ob**ject

“A blob is a group of connected pixels that share some common property” [6]

1. Thresholding
2. Grouping
3. Merging
4. Center & Radius Calculation

Blob Detection

Binary **L**arge **Obj**ect

“A blob is a group of connected pixels that share some common property” [6]

1. Thresholding
   - Make multiple binary images using different thresholding methods
2. Grouping
3. Merging
4. Center & Radius Calculation

Blob Detection

Binary **L**arge **Obj**ect

“A blob is a group of connected pixels that share some common property” [6]

1. Thresholding
   - Make multiple binary images using different thresholding methods
2. Grouping
   - Create curves from continuous points(same color/intensity) in binary images
3. Merging
4. Center & Radius Calculation

Blob Detection

Binary Large Object

"A blob is a group of connected pixels that share some common property" [6]

1. Thresholding
   - Make multiple binary images using different thresholding methods
2. Grouping
   - Create curves from continuous points(same color/intensity) in binary images
3. Merging
   - Compute centers of binary blobs, if distance less than threshold, merge
4. Center & Radius Calculation

Blob Detection

Binary **L**arge **Ob**ject

“A blob is a group of connected pixels that share some common property” [6]

1. Thresholding
   - Make multiple binary images using different thresholding methods
2. Grouping
   - Create curves from continuous points(same color/intensity) in binary images
3. Merging
   - Compute centers of binary blobs, if distance less than threshold, merge
4. Center & Radius Calculation
   - Calculate center and radius of each blob

Blob Detection in OpenCV

```python
# create blob detector object
blob = cv.SimpleBlobDetector_create()
# detect keypoints
kp = blob.detect(img, None)
# draw keypoints
img = cv.drawKeypoints(
    still[1],
    kp,
    outImage=None,
    color=(0, 0, 255),
    flags=cv.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)
```

Descriptors

- Keypoints ☞ areas of interest
- Descriptors ☞ describe keypoints
- Not calculated by all algorithms seen up to this point
  - `<obj>.detect(...)` gives keypoints
  - `<obj>.compute(...)` gives descriptors
  - `<obj>.detectAndCompute(...)` for both

BRIEF

- Binary Robust Independent Elementary Features [7]
- Uses less memory than SIFT & SURF
- Finds location pairs and compares pixel intensity
- Finds binary strings without finding initial descriptors
- Must be provided features
- BRIEF paper recommends CenSurE (STAR in OpenCV)

ORB

- Oriented FAST and Rotated BRIEF [8]
- FAST → finds keypoints
- Harris to find top *N* points
- Pyramid to check various scales
- Calculates orientation
- BRIEF → descriptors with modifications

ORB in OpenCV

```
# create ORB
orb = cv.ORB_create()
# detect
kp = orb.detect(img, None)
# draw output
img = cv.drawKeypoints(
      still[1],
      kp,
      outImage=None,
      color=(0, 255, 0),
      flags=0)
```

Bibliography I

[1] "Shi-Tomasi Corner Detector & Good Features to Track — OpenCV-Python Tutorials 1 documentation," (), [Online]. Available: https://opencv24-python-tutorials.readthedocs.io/en/stable/py_tutorials/py_feature2d/py_shi_tomasi/py_shi_tomasi.html#shi-tomasi (visited on 09/28/2024).

[2] "Introduction to SIFT (Scale-Invariant Feature Transform) — OpenCV-Python Tutorials 1 documentation," (), [Online]. Available: https://opencv24-python-tutorials.readthedocs.io/en/stable/py_tutorials/py_feature2d/py_sift_intro/py_sift_intro.html (visited on 09/28/2024).

Bibliography II

[3] H. Alsulaiman, “Feature-matching using BRISK,” Analytics Vidhya. (Mar. 19, 2022), [Online]. Available: https://medium.com/analytics-vidhya/feature-matching-using-brisk-277c47539e8 (visited on 09/28/2024).

[4] “Introduction to SURF (Speeded-Up Robust Features) — OpenCV-Python Tutorials beta documentation,” (), [Online]. Available: https://opencv24-python-tutorials.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_surf_intro/py_surf_intro.html (visited on 09/28/2024).

[5] “FAST Algorithm for Corner Detection — OpenCV-Python Tutorials beta documentation,” (), [Online]. Available: https://opencv24-python-tutorials.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_fast/py_fast.html (visited on 09/28/2024).

Bibliography III

[6] "Blob Detection Using OpenCV (Python, C++) |," (Feb. 17, 2015), [Online]. Available: https://learnopencv.com/blob-detection-using-opencv-python-c/ (visited on 09/28/2024).

[7] "BRIEF (Binary Robust Independent Elementary Features) — OpenCV-Python Tutorials beta documentation," (), [Online]. Available: https://opencv24-python-tutorials.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_brief/py_brief.html (visited on 09/28/2024).

[8] "ORB (Oriented FAST and Rotated BRIEF) — OpenCV-Python Tutorials beta documentation," (), [Online]. Available: https://opencv24-python-tutorials.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_orb/py_orb.html (visited on 09/28/2024).

Temporary page!

LaTeX was unable to guess the total number of pages correctly. As there was some unprocessed data that should have been added to the final page this extra page has been added to receive it.

If you rerun the document (without altering it) this surplus page will go aw

because LaTeX now knows how many pages to expect for this document.