Lecture 04: Feature Detection and Matching I

[AIX7021] Computer Vision

Seongsik Park (s.park@dgu.edu)

AI Department, Dongguk University

Feature detection and matching

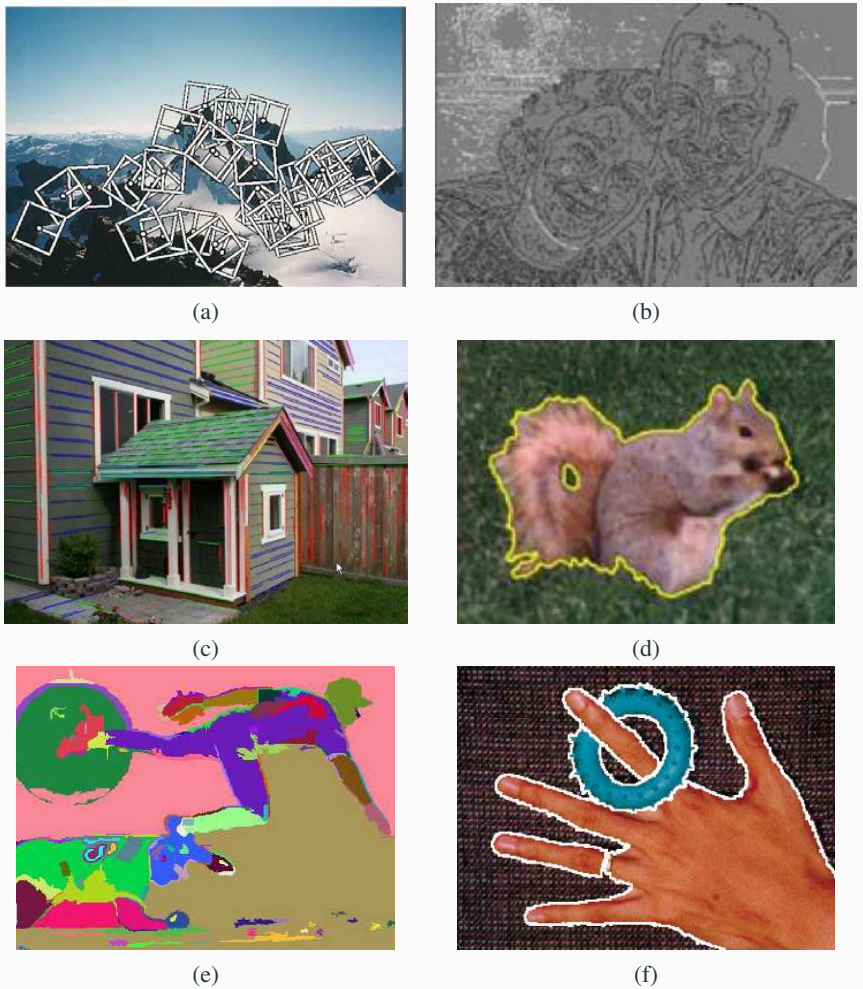


Figure 7.1 *Feature detectors and descriptors can be used to analyze, describe and match images: (a) point-like interest operators (Brown, Szeliski, and Winder 2005) © 2005 IEEE; (b) edges (Elder and Goldberg 2001) © 2001 IEEE; (c) straight lines (Sinha, Steedly* et al. *2008) © 2008 ACM; (d) level sets (Cremers, Rousson, and Deriche 2007) © 2007 Springer; (e) graph-based merging (Felzenszwalb and Huttenlocher 2004) © 2004 Springer; (f) mean shift (Comaniciu and Meer 2002) © 2002 IEEE.*

Feature detection and matching

Feature detection and matching are an essential component of many computer vision applications.

For the first pair, we may wish to *align* the two images so that they can be seamlessly stitched into a composite mosaic. For the second pair, we may wish to establish a dense set of *correspondences* so that a 3D model can be constructed or an in-between view can be generated.

In either case, what kinds of *features* should you detect and then match in order to establish such an alignment or set of correspondences?

feature detection and matching

What is Feature?

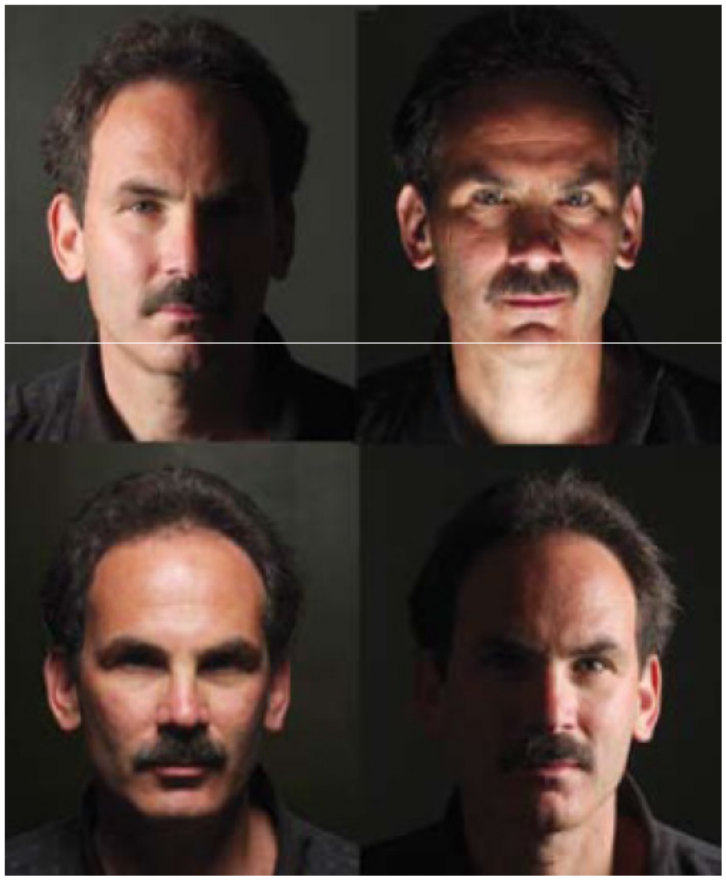
- An interesting part of an image
- A starting point for many computer vision algorithms

Good feature to do something related to computer vision

- Repeatability: the same feature should be detected in two or more different images of the same scene despite various geometric and photometric transformations
- Saliency: a feature includes an “interesting” part of an image
- Locality: a feature occupies a small area in an image
- Note: Global features are sometimes useful, too

Feature detection and matching: What is a good feature?

Invariant to view point, lighting, pose, and so on



Prof. D. Kriegman, UCSD

Feature detection and matching: What is a good feature?

The first kind of feature that you may notice are specific locations in the images, such as mountain peaks, building corners, doorways, or interestingly shaped patches of snow. These kinds of localized feature are often called *keypoint features* or *interest points* (or even *corners*) and are often described by the appearance of pixel patches surrounding the point location.

Figure 7.2 *Two pairs of images to be matched. What kinds of features might one use to establish a set of* correspondences *between these images?*

Feature detection and matching: What is a good feature?

Another class of important features are *edges*, e.g., the profile of mountains against the sky. These kinds of features can be matched based on their orientation and local appearance (edge profiles) and can also be good indicators of object boundaries and occlusion events in image sequences.

Edges can be grouped into longer *curves* and *contours*, which can then be tracked. They can also be grouped into *straight line segments*, which can be directly matched or analyzed to find *vanishing points* and hence internal and external camera parameters.

Feature detection and matching: What is a good feature?

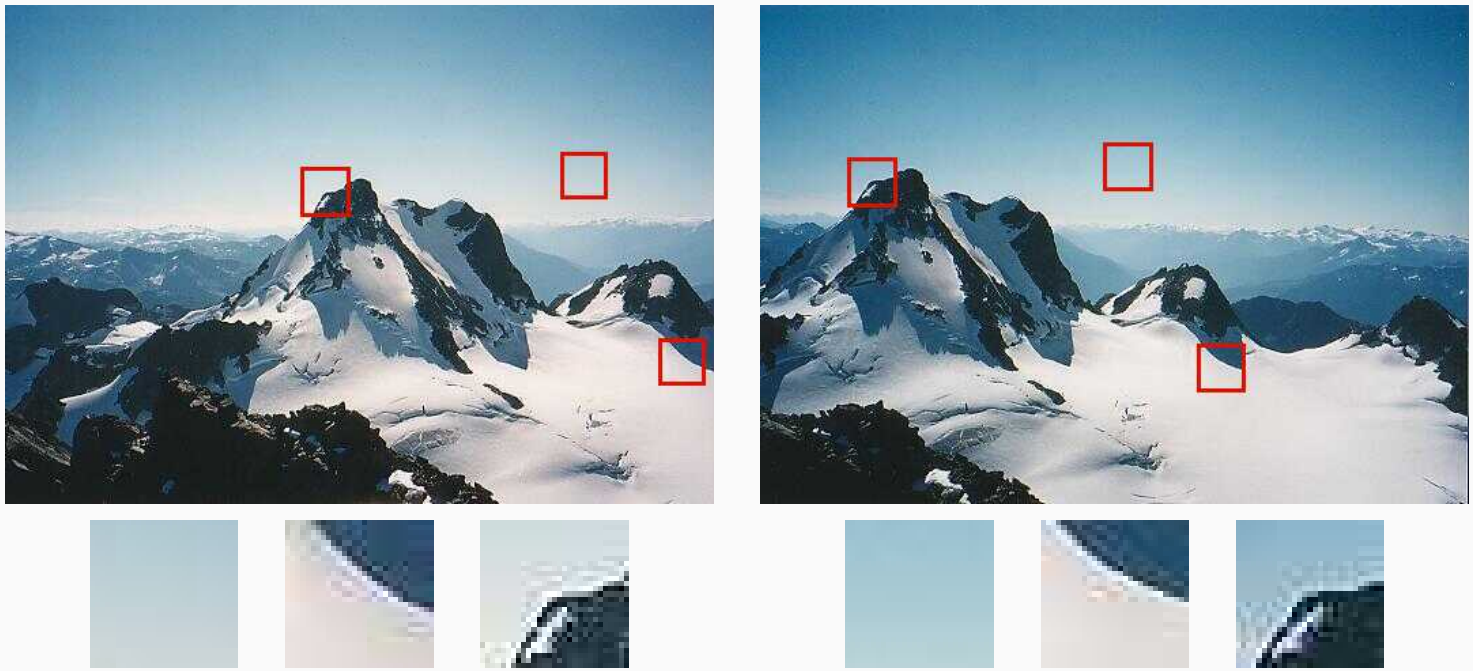


Figure 7.3 *Image pairs with extracted patches below. Notice how some patches can be localized or matched with higher accuracy than others.*

Examples of features

Examples

- Intensity
- Color
- Edge
- Line
- Corner
- Curve
- Blob
- Shape
- Texture
- Gradient
- Motion
- etc.

Other feature descriptors

- SIFT
- HOG
- LBP
- etc.

Feature detectors

Feature detectors: aperture problem

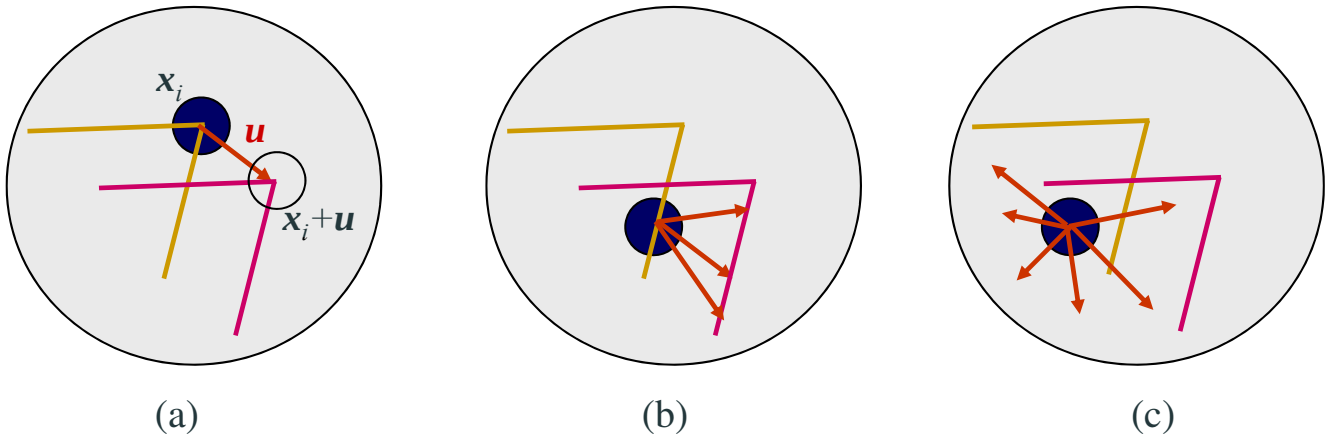


Figure 7.4 *Aperture problems for different image patches: (a) stable ("corner-like") flow; (b) classic aperture problem (barber-pole illusion); (c) textureless region. The two images I_0 (yellow) and I_1 (red) are overlaid. The red vector $\mathbf{u}$ indicates the displacement between the patch centers and the $w(\mathbf{x}_i)$ weighting function (patch window) is shown as a dark circle.*

Feature detectors: weighted summed square difference

These intuitions can be formalized by looking at the simplest possible matching criterion for comparing two image patches, i.e., their (weighted) summed square difference,

$$E_{\mathrm{WSSD}}(\mathbf{u}) = \sum_i w(\mathbf{x}_i) \left[I_1(\mathbf{x}_i + \mathbf{u}) - I_0(\mathbf{x}_i) \right]^2 \tag{1}$$

where I_0 and I_1 are the two images being compared, $\mathbf{u} = (u, v)$ is the *displacement* vector, $w(\mathbf{x})$ is a spatially varying weighting (or window) function, and the summation i is over all the pixels in the patch.

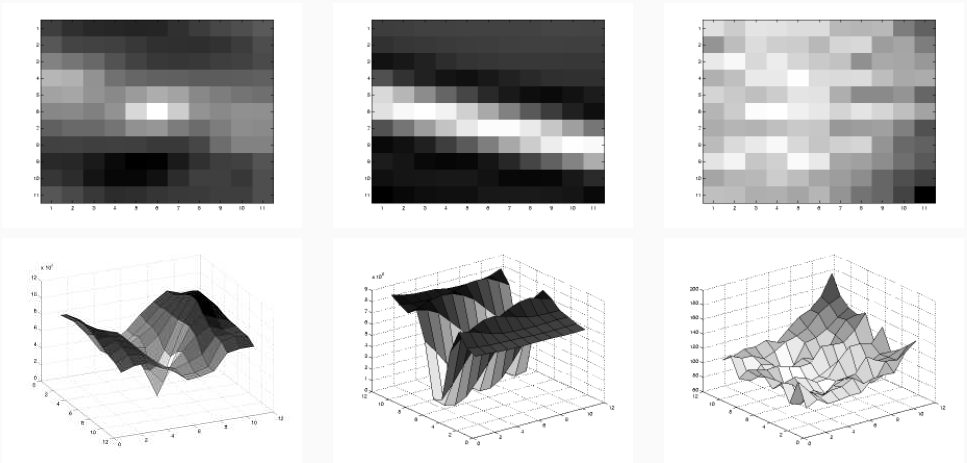
Feature detectors: auto-correlation or surface

When performing feature detection, we do not know which other image locations the feature will end up being matched against. Therefore, we can only compute how stable this metric is with respect to small variations in position $\Delta\mathbf{u}$ by comparing an image patch against itself, which is know as an *auto-correlation function* or *surface*

$$E_{\mathbf{AC}}(\Delta\mathbf{u}) = \sum_i w(\mathbf{x}_i) \left[I_0(\mathbf{x}_i + \Delta\mathbf{u}) - I_0(\mathbf{x}_i)\right]^2 \tag{2}$$

Feature detectors: autocorrelation surface

(a)



(b) (c) (d)

Figure 7.5 *Three auto-correlation surfaces $E_{\mathrm{AC}}(\Delta\mathbf{u})$ shown as both grayscale images and surface plots: (a) The original image is marked with three red crosses to denote where the auto-correlation surfaces were computed; (b) this patch is from the flower bed (good unique minimum); (c) this patch is from the roof edge (one-dimensional aperture problem); and (d) this patch is from the cloud (no good peak). Each grid point in figures b–d is one value of $\Delta\mathbf{u}$.*

Feature detectors: autocorrelation surface

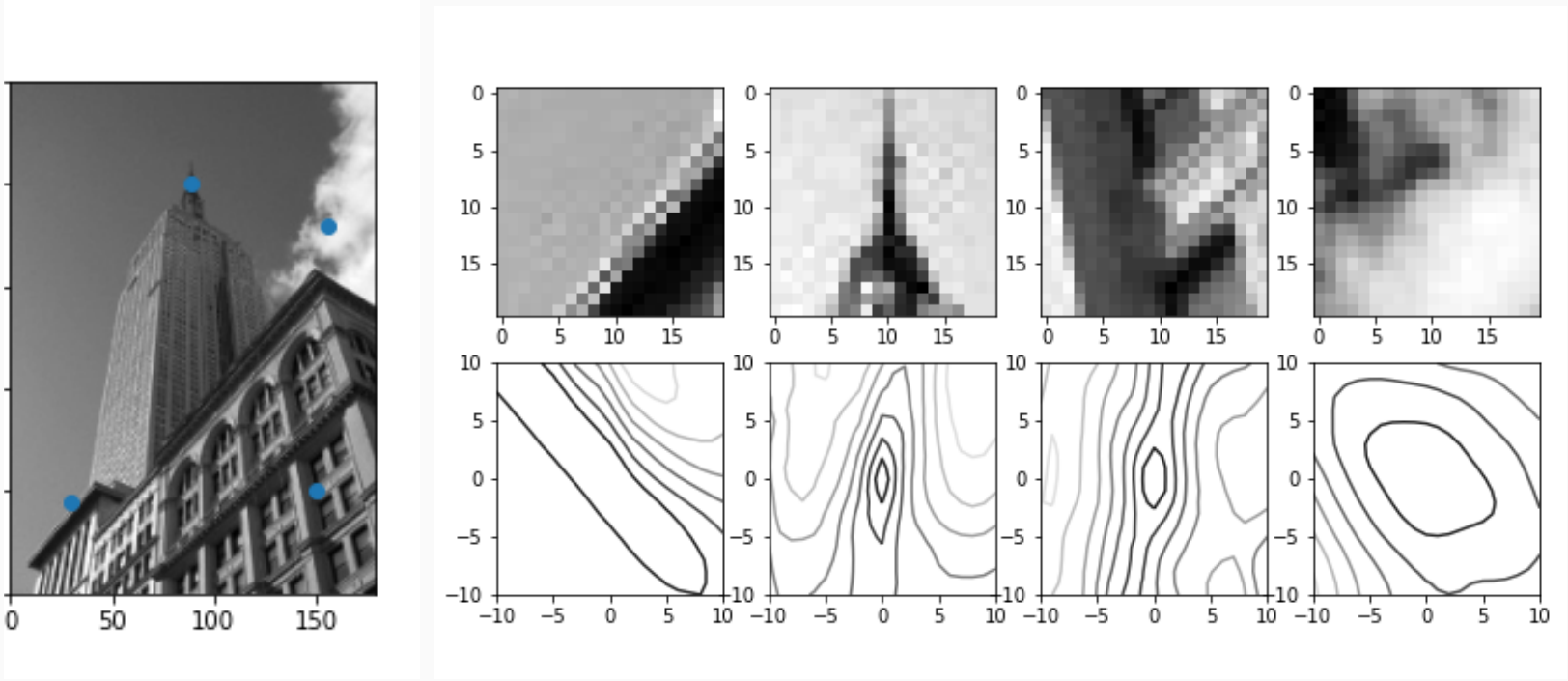


Figure 1: Original image (left), patches (top) and autocorrelation surface $E_{\mathrm{AC}}(\Delta\mathbf{u})$ (bottom).

Feature detectors: autocorrelation matrix

Using a Taylor series expansion

$$E_{\mathbf{AC}}(\Delta\mathbf{u}) = \sum_i w(\mathbf{x}_i)\left[I_0(\mathbf{x}_i + \Delta\mathbf{u}) - I_0(\mathbf{x}_i)\right]^2 \tag{3}$$

$$\approx \sum_i w(\mathbf{x}_i)\left[I_0(\mathbf{x}_i) + \nabla I_0(\mathbf{x}_i)\Delta\mathbf{u} - I_0(\mathbf{x}_i)\right]^2 \tag{4}$$

$$= \sum_i w(\mathbf{x}_i)\left[\nabla I_0(\mathbf{x}_i)\Delta\mathbf{u}\right]^2 \tag{5}$$

$$= \Delta\mathbf{u}^T\mathbf{A}\Delta\mathbf{u} \tag{6}$$

where $\nabla I_0(\mathbf{x}_i) = \left(\frac{\partial I_0}{\partial x}, \frac{\partial I_0}{\partial y}\right)\mathbf{x}_i$ is the *image gradient* at $\mathbf{x}_i$.

Feature detectors: autocorrelation matrix

The classic "Harris" detector uses a $[\ -2 \quad -1 \quad 0 \quad 1 \quad 2\ ]$ filter, but more modern variants convolve the image with horizontal and vertical derivatives of a Gaussian (typically with $\sigma = 1$).

The auto-correlation matrix $\mathbf{A}$ can be written as

$$\mathbf{A} = w * \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \tag{7}$$

where we have replaced the weighted summations with discrete convolutions with the weighting kernel w.

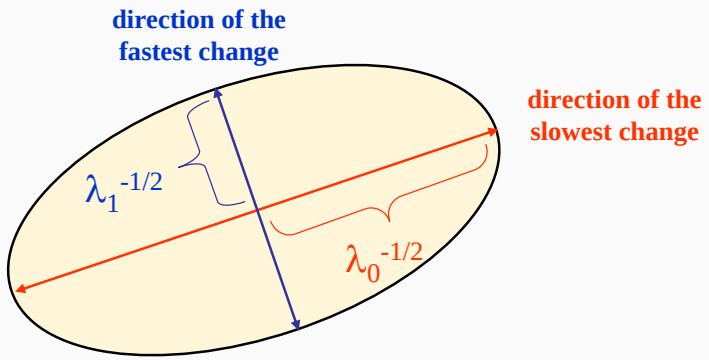


Figure 7.6 *Uncertainty ellipse corresponding to an eigenvalue analysis of the auto-correlation matrix* $\mathbf{A}$.

Feature detectors: autocorrelation matrix

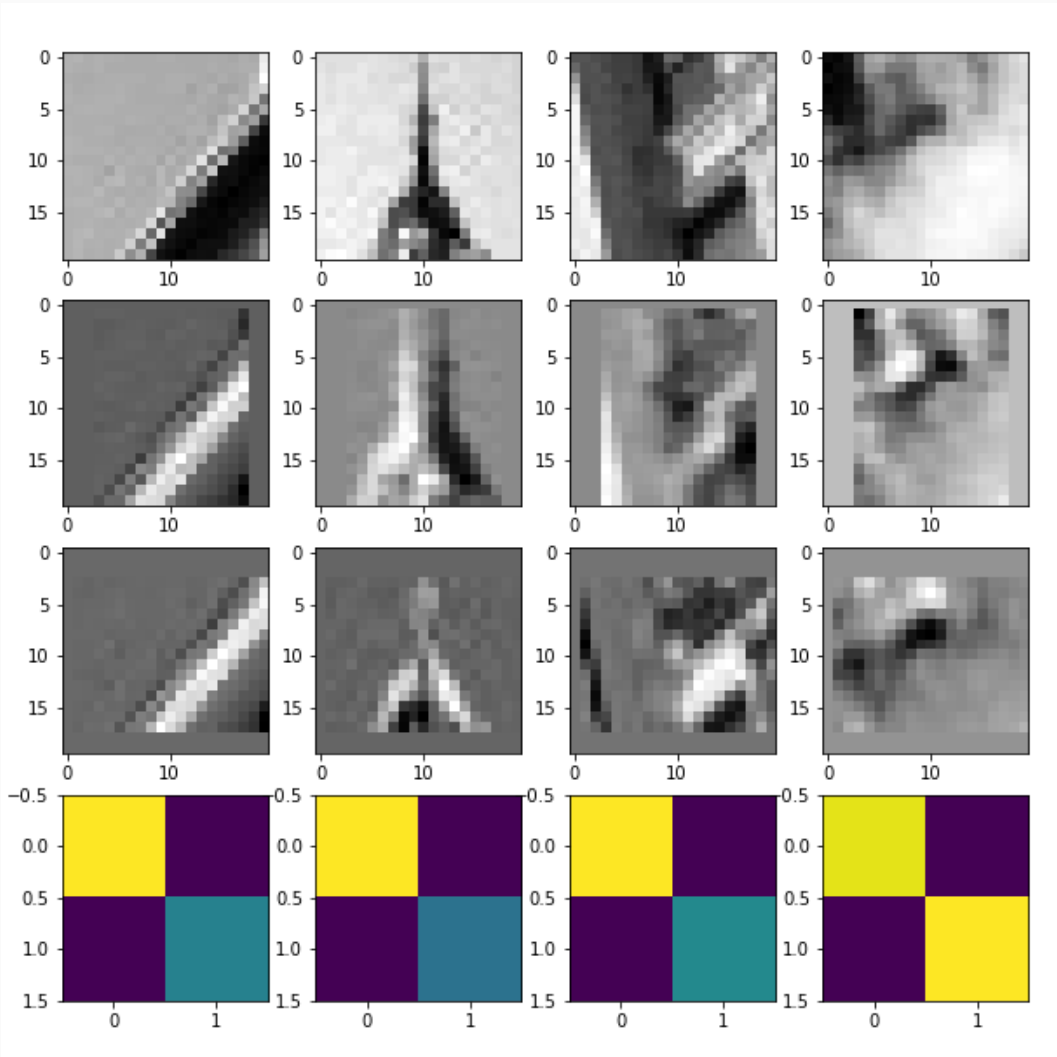


Figure 2: Patches (1st), gradient images (2nd and 3rd) and autocorrelation matrix (4th).

Feature detectors: autocorrelation matrix

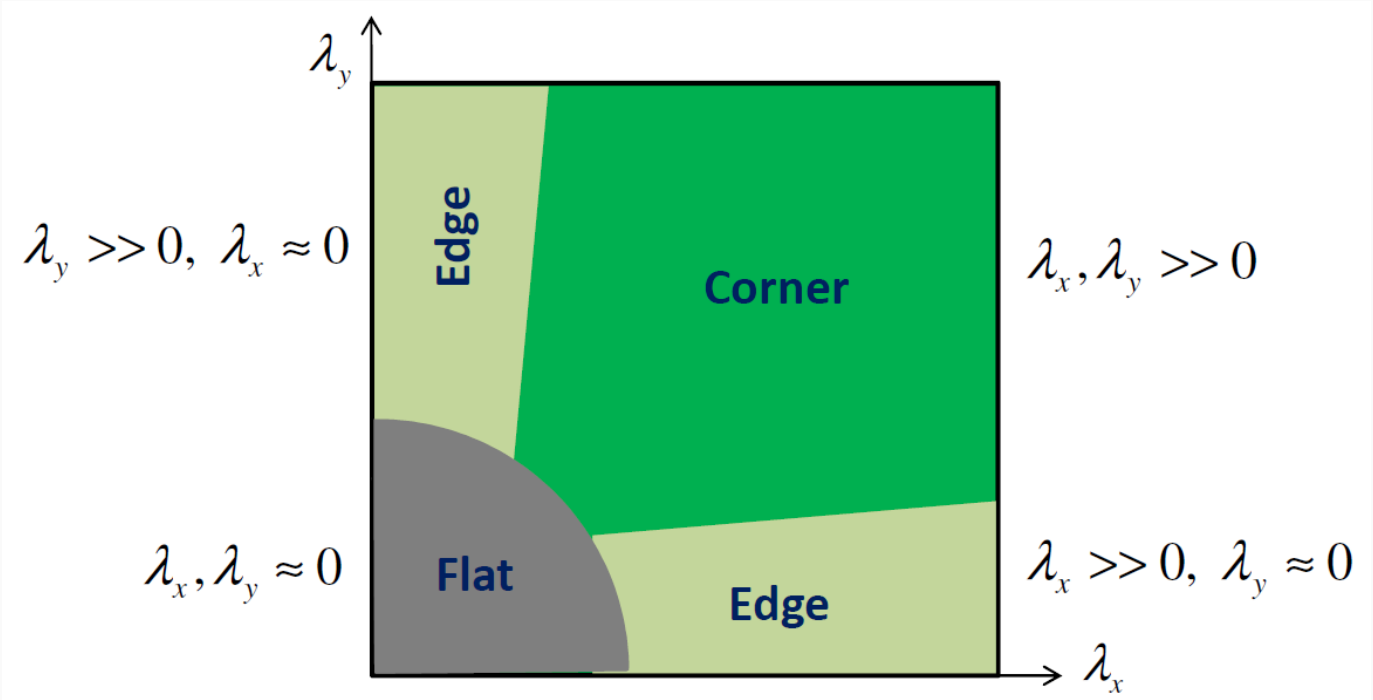


Figure 3: Corners and edges detection by eigenvalue analysis.

Feature detectors: autocorrelation matrix

- Inverse of the matrix $\mathbf{A}$ provides a lower bound on the uncertainty n the location of a matching patch.
- The easiest way to visualize this uncertainty is to perform an eigenvalue analysis of the autocorrelation matrix $\mathbf{A}$, which provides two eigenvalues (λ_0, λ_1) and two eigenvector directions
- Since the larger uncertainty depends on the smaller eigenvalue, i.e., $\lambda_0^{-1/2}$
- It makes sense to find maxima in the smaller eigenvalue to locate good features to track
- Simpler quantity than the minimum eigenvalue λ_0 is

$$R = \det(\mathbf{A}) - \alpha \mathrm{trace}(\mathbf{A})^2 = \lambda_0 \lambda_1 - \alpha(\lambda_0 + \lambda_1)^2 \tag{8}$$

  with $\alpha = 0.06$.
- and so on...

Feature detectors: Harris corner detection

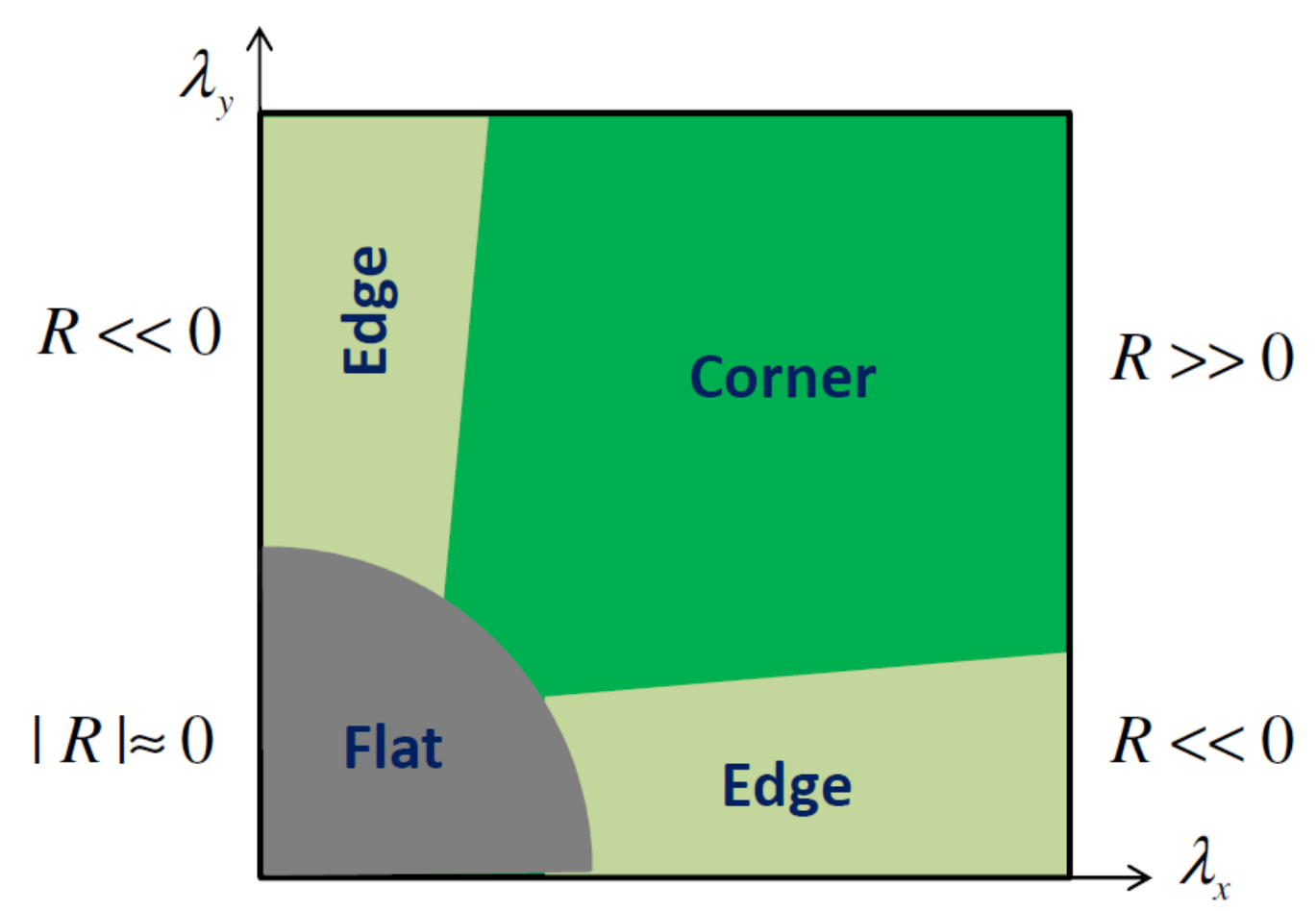


Figure 4: Corners and edges detection by Harris detection.

Feature detectors: summarize

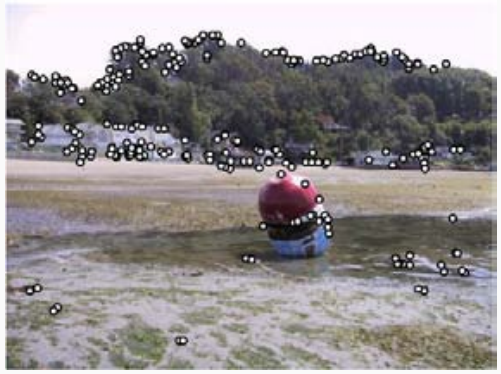
1. Compute the horizontal and vertical derivatives of the image I_x and I_y by convolving the original image with derivatives of Gaussians (Section 3.2.3).
2. Compute the three images corresponding to the outer products of these gradients. (The matrix $\boldsymbol{A}$ is symmetric, so only three entries are needed.)
3. Convolve each of these images with a larger Gaussian.
4. Compute a scalar interest measure using one of the formulas discussed above.
5. Find local maxima above a certain threshold and report them as detected feature point locations.

Algorithm 4.1 Outline of a basic feature detection algorithm.

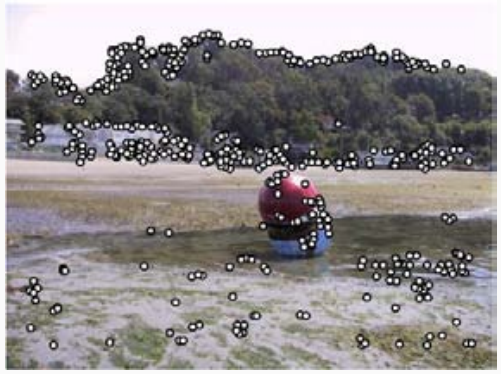
Feature detectors: adaptive non-maximal suppression (ANMS)

- While most feature detectors simply look for local maxima in the interest function, this can lead to an uneven distribution of feature points across the image, e.g., points will be denser in regions of higher contrast.
- Only detect features that are both local maxima and whose response value is significantly (10%) greater than that of all of its neighbors within a radius r

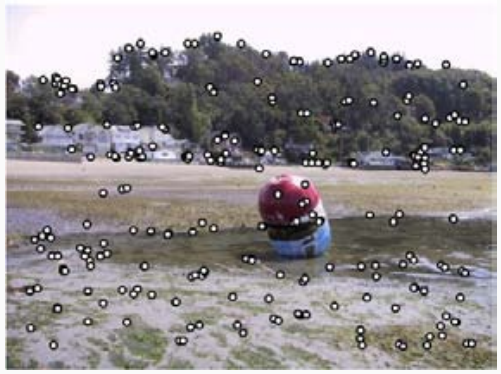
Feature detectors: adaptive non-maximal suppression (ANMS)



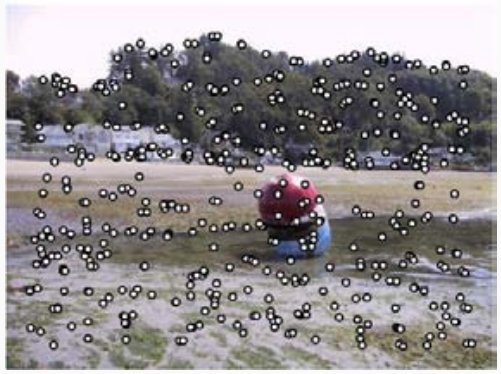
(a) Strongest 250



(b) Strongest 500



(c) ANMS 250, $r = 24$



(d) ANMS 500, $r = 16$

Figure 4.9 Adaptive non-maximal suppression (ANMS) (Brown, Szeliski, and Winder 2005) © 2005 IEEE: The upper two images show the strongest 250 and 500 interest points, while the lower two images show the interest points selected with adaptive non-maximal suppression, along with the corresponding suppression radius r. Note how the latter features have a much more uniform spatial distribution across the image.

Feature detectors: scale invariance

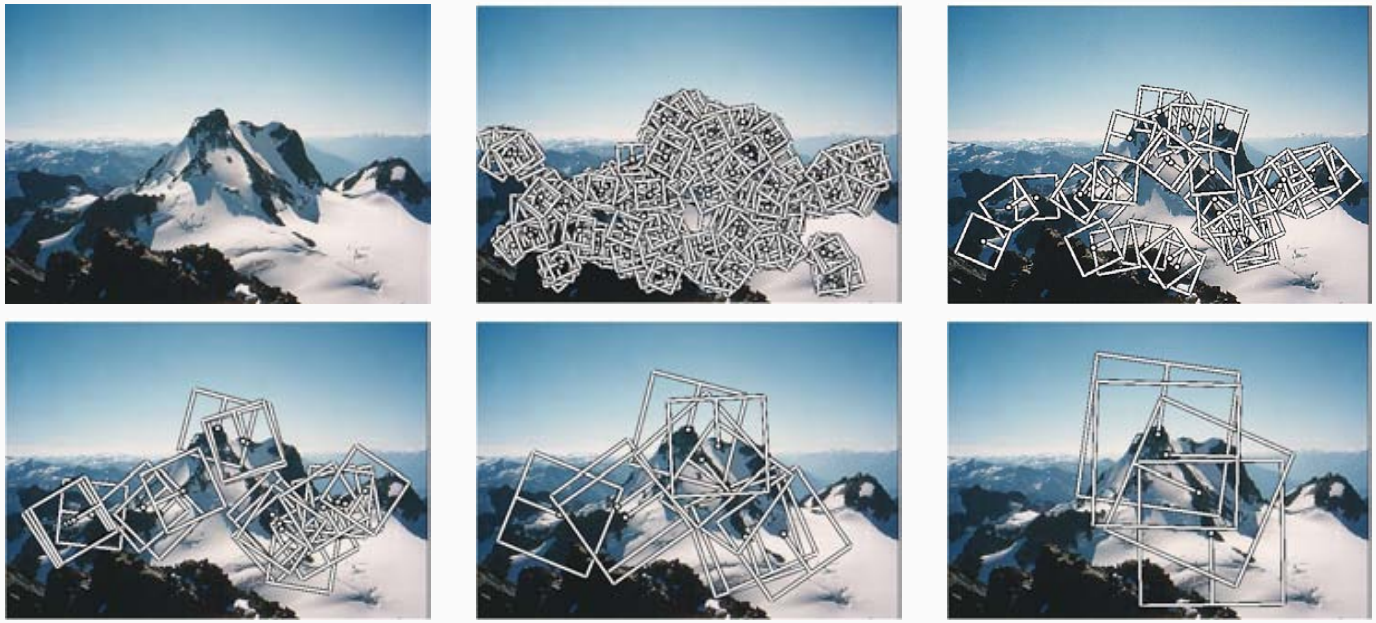


Figure 4.10 Multi-scale oriented patches (MOPS) extracted at five pyramid levels (Brown, Szeliski, and Winder 2005) © 2005 IEEE. The boxes show the feature orientation and the region from which the descriptor vectors are sampled.

Feature detectors: scale invariance

- To extract features a t a variety of scales, e.g., by performing the same operations at multiple resolutions in a pyramid and then matching features a t the same level.
- This kind of approach is suitable when the images being matched do not undergo large scale changes, e.g., when matching successive aerial images taken form an airplane or stitching panoramas
- For most object recognition applications, the scale of the object in the image is unknown
- Instead of extracting features at many different scales and then matching all of them, it is more efficient to extract features that are stable in both location *and* scale

Feature detectors: scale invariance

- Computing a set of sub-octave Difference of Gaussian filters (DoG)
- Looking for 3D (space + scale) maxima in the resulting structure
- and then computing a sub-pixel space+scale location using a quadratic fit
- The number of sub-octave levels was determined, to be three, which corresponds to a quarter-octave pyramid
- As with the Harris operator, pixels where there is strong asymmetry in the local curvature of the indicator function (in this case, the DoG) are rejected. This is implemented by first computing the local Hessian of the difference image D,

$$\mathbf{H} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix} \tag{9}$$

and then rejecting keypoints for which

$$\frac{\mathrm{Tr}(\mathbf{H})^2}{\mathrm{Det}(\mathbf{H})} > 10 \tag{10}$$

Feature detectors: scale invariance

Different of Gaussians on image pyramid

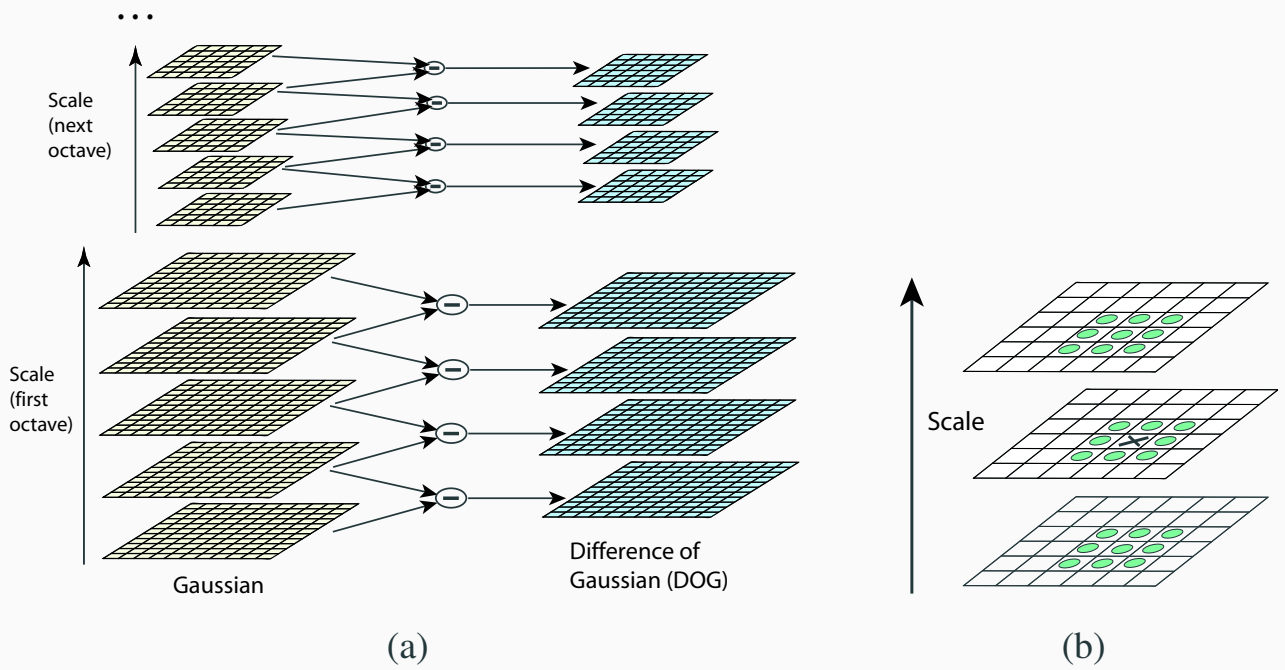
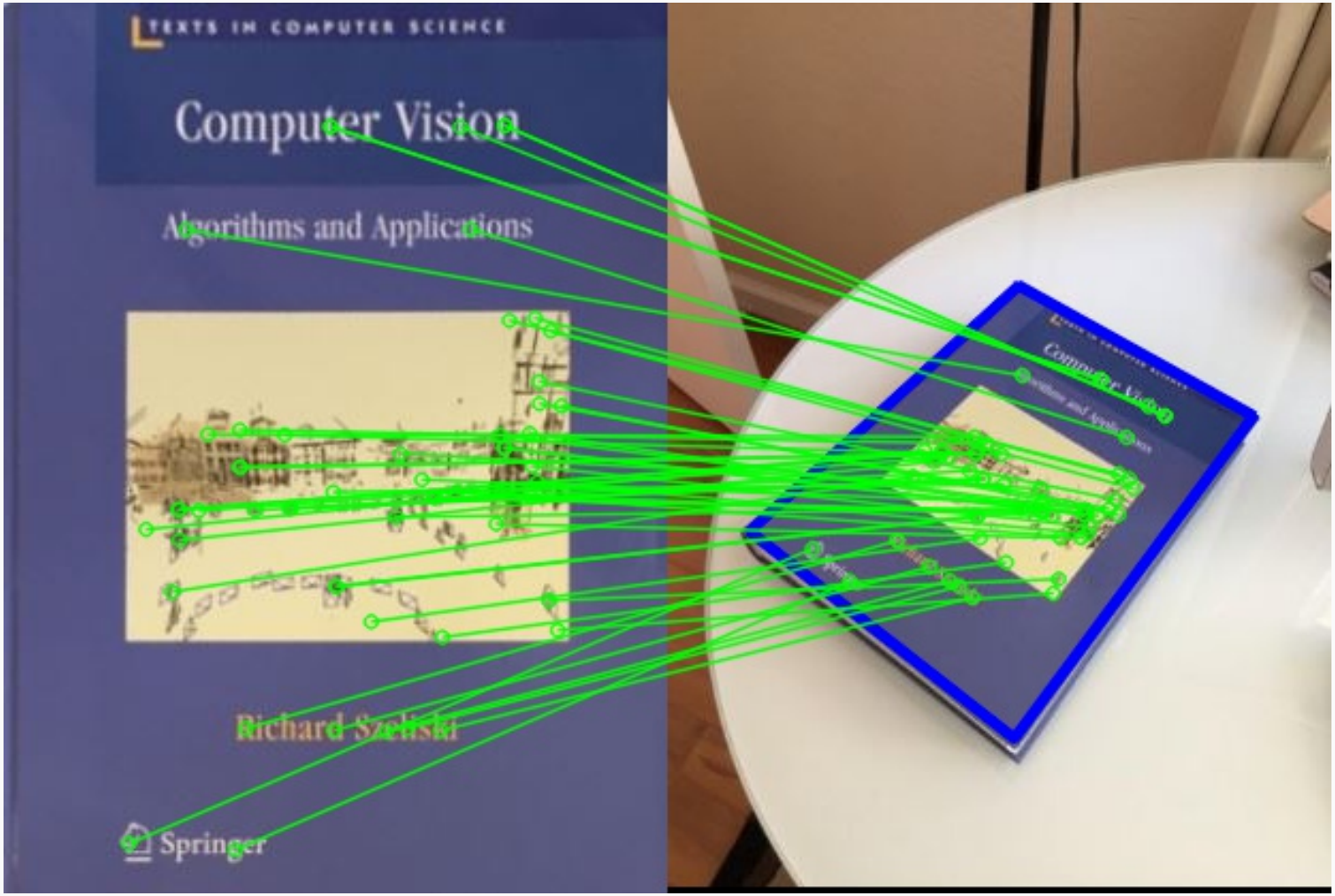


Figure 4.11 Scale-space feature detection using a sub-octave Difference of Gaussian pyramid (Lowe 2004) © 2004 Springer: (a) Adjacent levels of a sub-octave Gaussian pyramid are subtracted to produce Difference of Gaussian images; (b) extrema (maxima and minima) in the resulting 3D volume are detected by comparing a pixel to its 26 neighbors.

Next lecture: Feature detection and matching II



Appendix

Reference and further reading

Textbook

- “Chap 4: Feature detection and matching” of R. Szeliski, Computer Vision: Algorithms and Applications
- “Chap 5” of Forsyth and Ponce, Computer Vision: A Modern Approach

Links

- https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_matcher/py_matcher.html
- https://webnautes.tistory.com/1291