Feature Detection

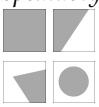
Adam Kalisz

18 September 2016

This handout accompanies the presentation about feature detection by Adam Kalisz during the Ferienakademie 2016.

What are (good) features?

"Repeatably recognizable regions in an image." (compare [1] Radke 2014).


Flat or Edges -> **Bad** (Ambiguous, Aperature Problem)

Corners or Blobs -> **Good**

Feature Detector

A detector is a mathematical way to formulate such good features, mentioned above.

This is usually done by computing the image gradient ∇I.

$$\nabla I = [I_x, I_y]^T$$

To get the partial Derivatives I_x and I_y we convolve a n-by-n ImageRegion (called "Patch") I with a respective n-by-n Filter-Kernel (like Prewitt, Sobel or Gaussian).

$$I_x = I * Kernel_x, I_y = I * Kernel_y$$

Filter kernels like Prewitt or Sobel have the drawbacks to be influenced by noise and image resolution. So in our case we take a Gaussian filter Kernel with $n = 5$ and $\sigma = 1.0$:

$$Kernel_x = [0.06136, 0.24477, 0.38774, 0.24477, 0.06136]^T$$

$$Kernel_y = [0.06136, 0.24477, 0.38774, 0.24477, 0.06136]$$

Invariance to noise, illumination, translation, rotation and scale are properties of good feature detectors.

Feature Descriptor

A descriptor is a vector which describes the properties of a detected feature, such as position, scale, rotation and its local 128 bin neighborhood histogram.

$$SiftFeature_1 = [318.861, 7.48227, 1.12, 1.685, 0, 0, 0, 1, 0, ...]$$

Harris-Stephens Corner Detector

Detector

"The Harris corner detection algorithm [...] is one of the simplest corner indicators available". (compare [3] Solem 2012, p. 29)

$$M_{Image} = \nabla I \nabla I^T = \begin{bmatrix} I_x \\ I_y \end{bmatrix} \begin{bmatrix} I_x & I_y \end{bmatrix} = \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

Using a Weight-Matrix we get a local averaging of the above Matrix over the neighboring pixels, often called the *Harris Matrix*:

$$H = W * M_{Image}$$

The Harris response is given by computing the eigenvalues of H or simpler by an indicator function, like this:

$$H_{Response} = det(H) - \kappa \cdot trace(H)^2 \approx \frac{det(H)}{trace(H)^2}$$

Descriptor

A possible way to create a descriptor for Harris features is through *Normalized cross-correlation*:

$$ncc(I_1, I_2) = \frac{1}{n-1} \sum_{\vec{\mathbf{x}}} \frac{(I_1(\vec{\mathbf{x}}) - \mu_1)}{\sigma_1} \cdot \frac{(I_2(\vec{\mathbf{x}}) - \mu_2)}{\sigma_2}$$

The similarity of two images I_1 and I_2 is determined by the sum of all positions $\vec{\mathbf{x}}$ in image patches using the number of Pixels n in a patch, the mean intensities μ_1 and μ_2 as well as the standard deviations in each patch σ_1 and σ_2.

SIFT (Scale-Invariant-Feature-Transform)

Detector

The only possible scale-space kernel is the Gaussian:

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}$$

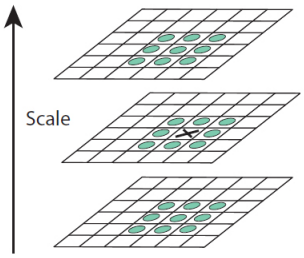
Hence we use it to define the scale-space of our image via convolution:

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y)$$

We can use the scale-space extrema in the difference-of-Gaussian function $D(x, y, \sigma)$ to efficiently detect stable keypoint locations:

$$D(x,y,\sigma) = (G(x,y,k\sigma) - G(x,y,\sigma)) * I(x,y) = L(x,y,k\sigma) - L(x,y,\sigma)$$

The local extrema detection is accomplished by comparing a sample point to its eight neighbors in the current image, as well as the nine neighbors in the scale above and below, like this:



The sample point is considered a feature point only, if it differs by a certain threshold to all of these neighboring points.

To improve the accuracy of the keypoint localization several filtering methods are being used (e.g. rejecting samples with low contrast or positioned along edges). Additionally principal curvatures can be computed from a 2x2 Hessian matrix at the keypoint location utilizing the efficient Harris-Stephens indicator function mentioned earlier.

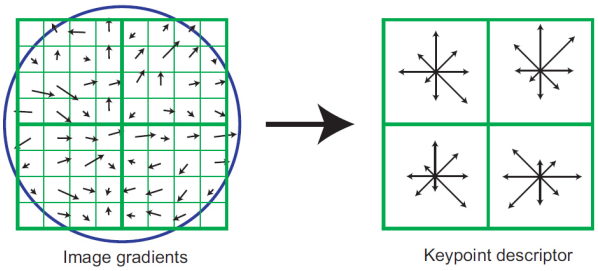
Descriptor

In order to provide a robust way to compare features across multiple images, a vector containing unique attributes is assigned to each keypoint, such as magnitude and orientation of the gradient using pixel differences:

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2}$$

$$\theta(x,y) = tan^{-1}(\frac{L(x,y+1) - L(x,y-1)}{L(x+1,y) - L(x-1,y)})$$

This can be visualized as follows:



References

[1] Rich Radke, 20.02.2014, *"CVFX Lecture 9: Feature Detectors"*, https://www.youtube.com/watch?v=9-F5-gUIAWE
[2] Rich Radke, 24.02.2014, *"CVFX Lecture 10: Feature descriptors"*, https://www.youtube.com/watch?v=oFVexhcltzE
[3] Jan Erik Solem, *"Programming Computer Vision with Python"*, 17.08.2012, Print ISBN: 978-1-4493-1654-9
[4] Dr. Mubarak Shah, 19.09.2012, *"Lecture 04 - Interest Point Detection"*, https://www.youtube.com/watch?v=_qgKQGsuKeQ
[5] Dr. Mubarak Shah, 19.09.2012, *"Lecture 05 - Scale-invariant Feature Transform (SIFT)"*, https://www.youtube.com/watch?v=NPcMS49V5hg
[6] David G. Lowe, 05.01.2004, *"Distinctive Image Features from Scale-Invariant Keypoints"*, https://www.cs.ubc.ca/~lowe/papers/ijcv04.pdf
[7] OpenCV, 28.08.2016, *"OpenCV: Introduction to SIFT (Scale-Invariant Feature Transform):"*, http://docs.opencv.org/3.1.0/da/df5/tutorial_py_sift_intro.html#gsc.tab=0