Deep Learning

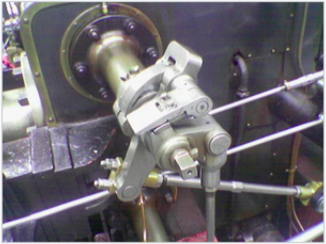
6. Overview of classical computer vision - 2. Corner detectors. SIFT. Viola-Jones Face Detector.

Viacheslav Dudar

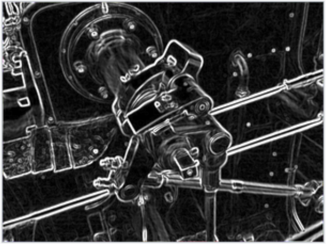
Taras Shevchenko National University of Kyiv

2018

Recall: edge detection



A color picture of a steam engine



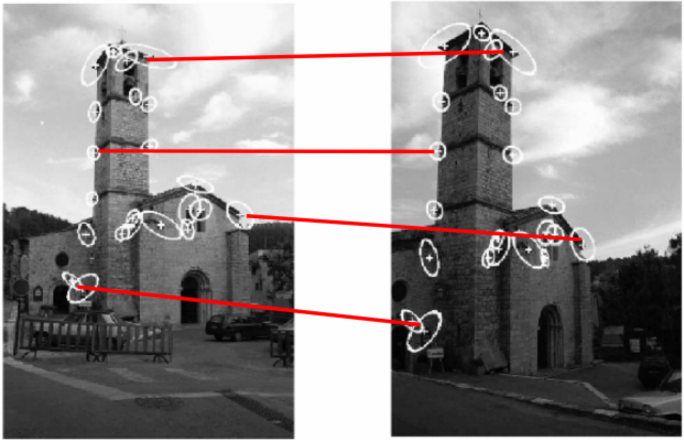
The Sobel operator applied to that image

$$\mathbf{G}_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} * \mathbf{A}$$

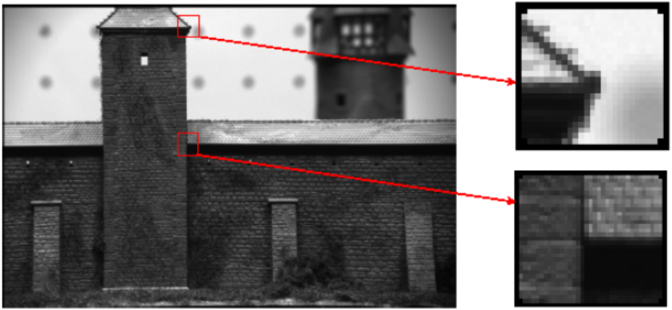
$$\mathbf{G}_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * \mathbf{A}$$

Corner detection: motivation

Vision tasks such as stereo and motion estimation require finding corresponding features across two or more views.



What are corners?



- Intuitively, junctions of contours.
- Generally more stable features over changes of viewpoint
- Intuitively, large variations in the neighborhood of the point in all directions
- They are good features to match!

Harris corner detection (1988)

Sum of squared differences between patches:

$$S(x,y) = \sum_u \sum_v w(u,v) \left(I(u+x, v+y) - I(u,v) \right)^2$$

Approximation with Taylor expansion:

$$I(u+x, v+y) \approx I(u,v) + I_x(u,v)x + I_y(u,v)y$$

$$S(x,y) \approx \sum_u \sum_v w(u,v) \left(I_x(u,v)x + I_y(u,v)y \right)^2$$

Harris matrix:

$$S(x,y) \approx \begin{pmatrix} x & y \end{pmatrix} A \begin{pmatrix} x \\ y \end{pmatrix}$$

$$A = \sum_u \sum_v w(u,v) \begin{bmatrix} I_x(u,v)^2 & I_x(u,v)I_y(u,v) \\ I_x(u,v)I_y(u,v) & I_y(u,v)^2 \end{bmatrix} = \begin{bmatrix} \langle I_x^2 \rangle & \langle I_x I_y \rangle \\ \langle I_x I_y \rangle & \langle I_y^2 \rangle \end{bmatrix}$$

Harris corner detection

Corner is characterized by large variation of S in all directions.
λ_1, λ_2: eigenvalues of the matrix A.
Harris matrix A is positive definite, so $\lambda_1, \lambda_2 \geq 0$

- $\lambda_1 \approx 0, \lambda_2 \approx 0$ - pixel (x,y) has no features of interest
- $\lambda_1 \approx 0, \lambda_2$ is large - pixel (x,y) lies on the edge
- λ_1 is large, λ_2 is large - pixel (x,y) lies on the corner

Corner response measure

Measure of corner response:

$$R = \det M - k\left(\operatorname{trace} M\right)^2$$

$$\det M = \lambda_1 \lambda_2$$
$$\operatorname{trace} M = \lambda_1 + \lambda_2$$

(k is an empirically determined constant; k = 0.04 - 0.06)

Harris corner detection algorithm

1. Compute x and y derivatives of image

$$I_x = G_\sigma^x * I \quad I_y = G_\sigma^y * I$$

2. Compute products of derivatives at every pixel

$$I_{x2} = I_x . I_x \quad I_{y2} = I_y . I_y \quad I_{xy} = I_x . I_y$$

3. Compute the sums of the products of derivatives at each pixel

$$S_{x2} = G_{\sigma'} * I_{x2} \quad S_{y2} = G_{\sigma'} * I_{y2} \quad S_{xy} = G_{\sigma'} * I_{xy}$$

4. Define at each pixel (x, y) the matrix

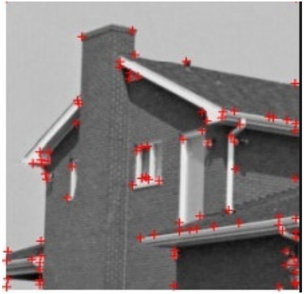
$$H(x,y) = \begin{bmatrix} S_{x2}(x,y) & S_{xy}(x,y) \\ S_{xy}(x,y) & S_{y2}(x,y) \end{bmatrix}$$

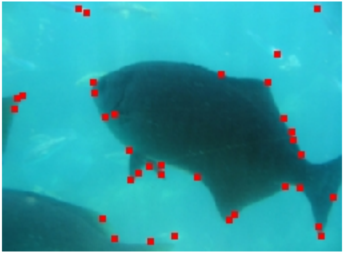
5. Compute the response of the detector at each pixel

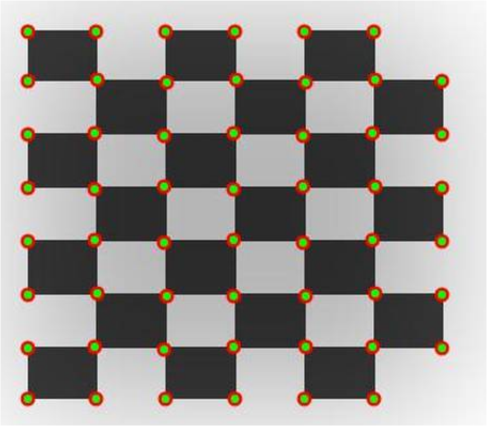
$$R = Det(H) - k(Trace(H))^2$$

6. Threshold on value of R. Compute nonmax suppression.

Harris corner application







Other methods

Try other corner detection methods:

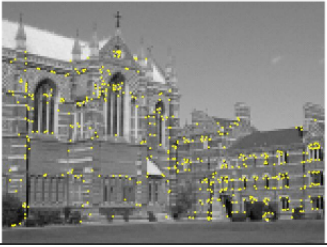
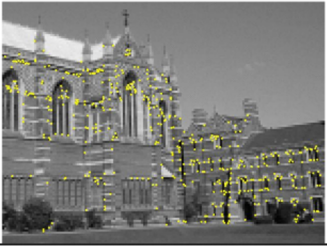
- Harris
- Shi and Tomasi
- Level curve curvature
- SUSAN
- FAST

SIFT (1999)

- SIFT: Scale Invariant Feature Transform
- It is a technique for detecting salient, stable feature points in an image.
- For every such point, it also provides a set of features that characterize/describe a small image region around the point. These features are **invariant to rotation and scale**.

Motivation for SIFT

- Image matching
  - Estimation of affine transformation/homography between images
  - Estimation of fundamental matrix in stereo
- Structure from motion, tracking, motion segmentation

Stages of SIFT algorithm

- Determine approximate location and scale of salient feature points (also called keypoints)
- Refine their location and scale
- Determine orientation(s) for each keypoint.
- Determine descriptors for each keypoint.

Stage 1: key point detection

We search for intensity changes using the difference of Gaussians at two nearby scales:

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

Convolution operator: refers to the application of a filter (in this case Gaussian filter to an image)

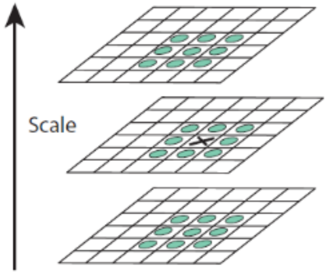
$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y)$$

$$\begin{aligned} D(x, y, \sigma) &= (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) \\ &= L(x, y, k\sigma) - L(x, y, \sigma). \end{aligned}$$

Difference of Gaussians = "DoG".

Scale refers to the σ of the Gaussian.

Stage 1



We compare pixel X with 26 neighbors in 3*3 regions at the current and adjacent scales.
Keypoints are maxima or minima in the scale space pyramid.

Step 2: refining keypoints

We interpolate DoG function in a small neighborhood around a keypoint.

$$D(x,y,\sigma) = D(x_i,y_i,\sigma_i) + \boxed{\left(\frac{\partial D(x,y,\sigma)}{\partial(x,y,\sigma)}\right)^T_{\substack{x=x_i,\\ y=y_i,\\ \sigma=\sigma_i}}} \Delta + \frac{1}{2}\Delta^T \boxed{\left(\frac{\partial^2 D(x,y,\sigma)}{\partial(x,y,\sigma)^2}\right)_{\substack{x=x_i,\\ y=y_i,\\ \sigma=\sigma_i}}} \Delta;$$

$$\Delta = \begin{pmatrix} x - x_i \\ y - y_i \\ \sigma - \sigma_i \end{pmatrix}$$

Gradient vector evaluated digitally at the keypoint

3 x 3 Hessian matrix evaluated digitally at the keypoint

Step 2: refining keypoints

- To find an extremum of the DoG values in this neighborhood, set the derivative of D(.) to 0. This gives us:

$$\begin{pmatrix} \hat{x} \\ \hat{y} \\ \hat{\sigma} \end{pmatrix} = -\left(\frac{\partial^2 D(x,y,\sigma)}{\partial(x,y,\sigma)^2}\right)^{-1}_{\substack{x=x_i,\\ y=y_i,\\ \sigma=\sigma_i}} \left(\frac{\partial D(x,y,\sigma)}{\partial(x,y,\sigma)}\right)_{\substack{x=x_i,\\ y=y_i,\\ \sigma=\sigma_i}}$$

$$\therefore D_{extremal} = D(x_i, y_i, \sigma_i) + \frac{1}{2}\left(\frac{\partial D(x,y,\sigma)}{\partial(x,y,\sigma)}\right)^{T}_{\substack{x=x_i,\\ y=y_i,\\ \sigma=\sigma_i}} \begin{pmatrix} \hat{x} \\ \hat{y} \\ \hat{\sigma} \end{pmatrix}$$

- The keypoint location is updated.
- All extrema with $|D_{extremal}| < 0.03$, are discarded as “weak extrema” or “low contrast points”.

Step 2: refining keypoints

We use eigenvalues of the Hessian matrix at the point to get rid of edge points:

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

$$\boxed{\frac{\mathrm{Tr}(\mathbf{H})^2}{\mathrm{Det}(\mathbf{H})}} = \frac{(\alpha+\beta)^2}{\alpha\beta} = \frac{(r\beta+\beta)^2}{r\beta^2} = \frac{(r+1)^2}{r}$$

Should be less than a threshold (say 10).

$$\mathrm{Tr}(\mathbf{H}) = D_{xx} + D_{yy} = \alpha + \beta,$$

$$\mathrm{Det}(\mathbf{H}) = D_{xx}D_{yy} - (D_{xy})^2 = \alpha\beta.$$

$$\boxed{\alpha = r\beta}$$

For an edge, α >> β, leading to a large value of this measure.

Why this measure instead of r? To save computations – we need not compute eigenvalues!

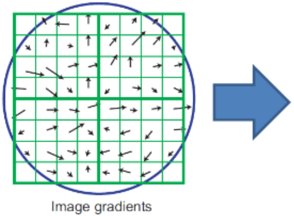
Step 3: assigning orientations

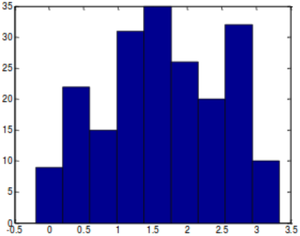
- Compute the gradient magnitudes and orientations in a small window around the keypoint – at the appropriate scale.

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

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

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





Histogram of gradient orientation – the bin-counts are weighted by gradient magnitudes and a Gaussian weighting function. Usually, 36 bins are chosen for the orientation.

Step 4: Descriptors for each keypoint

- Consider a small region around the keypoint. Divide it into *n* x *n* cells (usually *n* = 2). Each cell is of size 4 x 4.
- Build a **gradient orientation histogram** in each cell. Each histogram entry is weighted by the *gradient magnitude* and a *Gaussian weighting function* with σ = 0.5 times window width.
- **Sort** each gradient orientation histogram bearing in mind the dominant orientation of the keypoint (assigned in step 3).

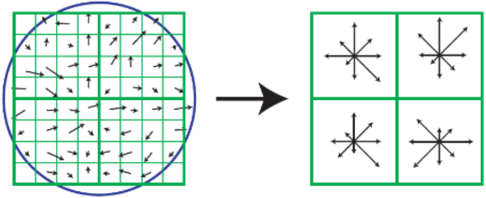
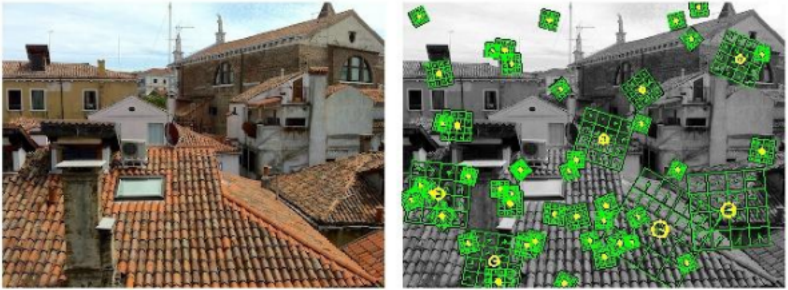


Image gradients   Keypoint descriptor

Image taken from D. Lowe, "Distinctive Image Features from Scale-Invariant Points", IJCV 2004

Step 4: Descriptors for each keypoint

For scale invariance, we adjust size of the window by the scale of the keypoint.



Step 4: Descriptors for each keypoint

- We now have a descriptor of size rn^2 if there are r bins in the orientation histogram.
- Typical case used in the SIFT paper: $r = 8$, $n = 4$, so length of each descriptor is 128.
- The descriptor is invariant to rotations due to the sorting.

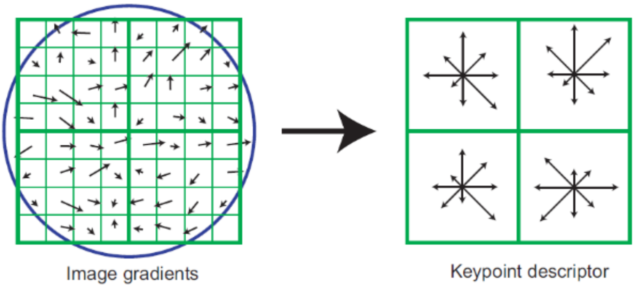
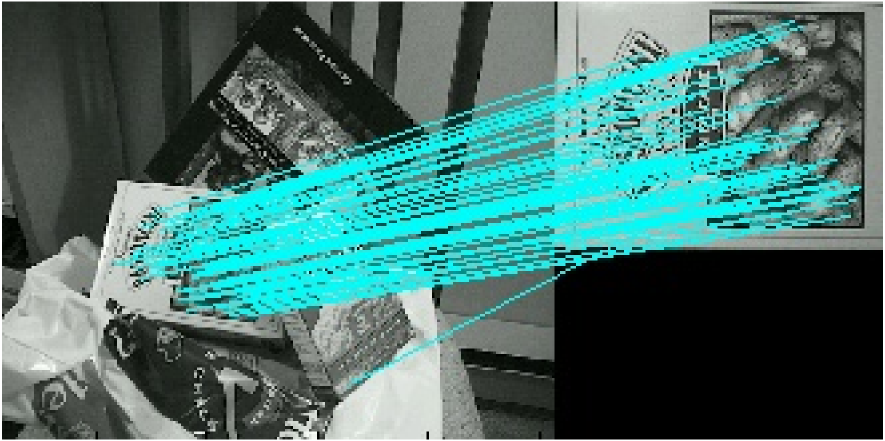


Image taken from D. Lowe, "Distinctive Image Features from Scale-Invariant Points", IJCV 2004

SIFT example



Feature descriptors

Try other feature descriptors:

- SIFT
- SURF
- GLOH
- HOG

Applications of key points matching

- Image detection
- Image recognition
- Image stitching
- 3D surface restoration

ViolaJones object detection framework (2001)

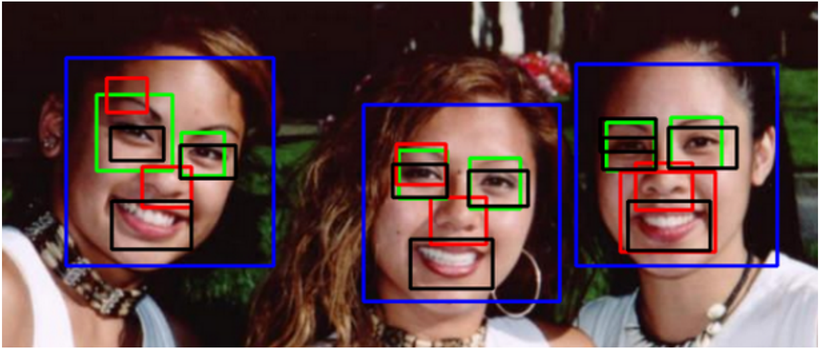
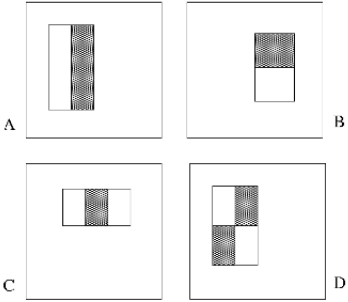


Image features

"Rectangle filters"

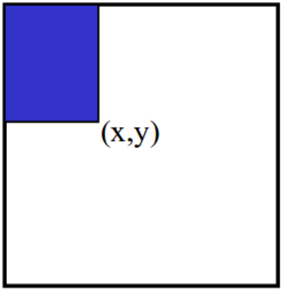


Value =

∑ (pixels in white area) –
∑ (pixels in black area)

Fast features computation

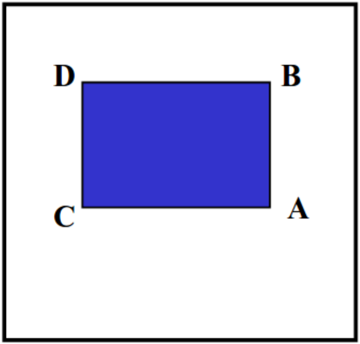
- The *integral image* computes a value at each pixel (x,y) that is the sum of the pixel values above and to the left of (x,y), inclusive
- This can quickly be computed in one pass through the image



Fast features computation

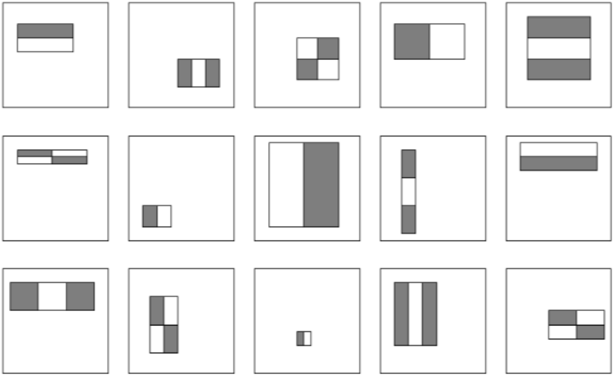
- Let A,B,C,D be the values of the integral image at the corners of a rectangle
- Then the sum of original image values within the rectangle can be computed as:

  sum = A – B – C + D
- Only 3 additions are required for any size of rectangle!
  - This is now used in many areas of computer vision



Feature selection

- For a 24x24 detection region, the number of possible rectangle features is ~180,000!



How to select small number of good features?

Boosting

Boosting is a classification scheme that works by combining weak learners into a more accurate ensemble classifier.
Weak learner: classifier with accuracy that need be only better than chance.

- Given a set of weak classifiers originally : $h_j(\mathbf{x}) \in \{+1, -1\}$
  - None much better than random
- Iteratively combine classifiers
  - Form a linear combination

$$C(x) = \theta\left(\sum_t h_t(x) + b\right)$$

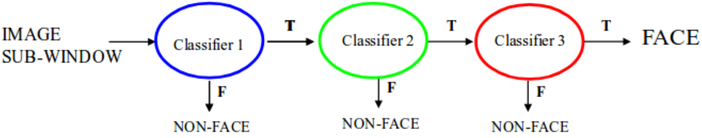
  - Training error converges to 0 quickly
  - Test error is related to training margin

Boosting outline

- Initially, give equal weight to each training example
- Iterative training procedure
  - Find best weak learner for current weighted training set
  - Raise the weights of training examples misclassified by current weak learner
- Compute final classifier as linear combination of all weak learners (weight of each learner is related to its accuracy)

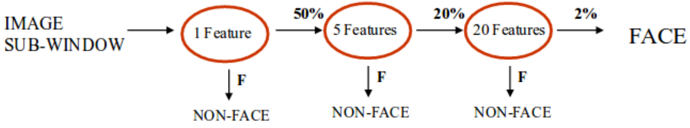
Cascading classifiers

- We start with simple classifiers which reject many of the negative sub-windows while detecting almost all positive sub-windows
- Positive results from the first classifier triggers the evaluation of a second (more complex) classifier, and so on
- A negative outcome at any point leads to the immediate rejection of the sub-window



Cascading classifiers

- Adjust weak learner threshold to minimize *false negatives* (as opposed to total classification error)
- Each classifier trained on false positives of previous stages
  - A single-feature classifier achieves 100% detection rate and about 50% false positive rate
  - A five-feature classifier achieves 100% detection rate and 40% false positive rate (20% cumulative)
  - A 20-feature classifier achieve 100% detection rate with 10% false positive rate (2% cumulative)



Face detection results