# Harris Corner Detector

Tomáš Werner

Center for Machine Perception
Czech Technical University
Prague

Design a detector that finds points in an image such that:

- There is only a small number of isolated points detected.
- The points are reasonably invariant to
  - rotation,
  - different sampling and quantization,
  - to small changes of scale and small affine transformations.

Usage:

- Matching, finding correspondence
- Tracking

The standard detector satisfying these requirements is **Harris corner detector** (it was proposed by other people earlier, Harris became most known for some reason).

- How similar is the image function $I(x, y)$ at point $(x, y)$ similar to itself, when shifted by $(\Delta x, \Delta y)$?
- This is given by autocorrelation function

$$c(x, y; \Delta x, \Delta y) = \sum_{(u,v)\in W(x,y)} w(u, v)\big(\, I(u, v) - I(u + \Delta x, v + \Delta y)\,\big)^2$$

where

- $W(x, y)$ is a window centered at point $(x, y)$
- $w(u, v)$ is either constant or (better) Gaussian $\exp \dfrac{-(u - x)^2 - (v - y)^2}{2\sigma^2}$.

(Further on, we will replace $\sum_{(u,v)\in W(x,y)} w(u, v)$ with $\sum_{W}$ for simplicity)

Approximate the shifted function by the first-order Taylor expansion:

$$\begin{aligned} I(u+\Delta x, v+\Delta y) &\approx I(u,v) + I_x(u,v)\Delta x + I_y(u,v)\Delta y \\ &= I(u,v) + \left[\, I_x(u,v), I_y(u,v) \,\right] \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} \end{aligned}$$

where $I_x, I_y$ are partial derivatives of $I(x,y)$.

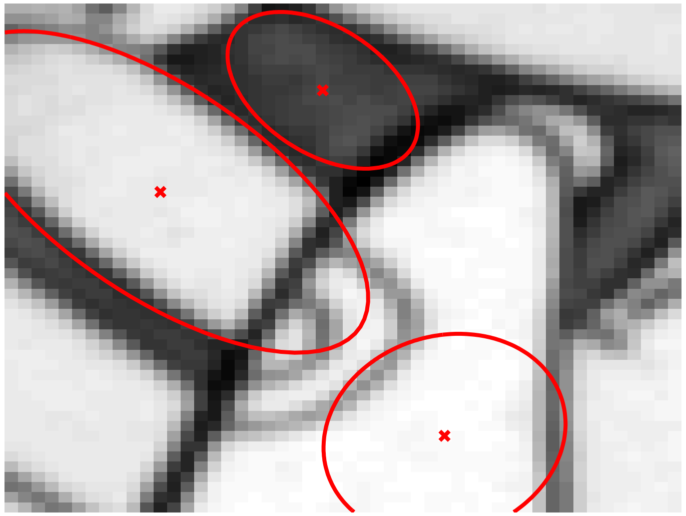
$$\begin{aligned} c(x,y;\Delta x,\Delta y) &= \sum_W \left( I(u,v) - I(u+\Delta x, v+\Delta y) \right)^2 \\ &\approx \sum_W \left( \left[\, I_x(u,v), I_y(u,v) \,\right] \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} \right)^2 \\ &= \left[\, \Delta x, \Delta y \,\right] Q(x,y) \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} \end{aligned}$$

$$\begin{aligned} Q(x,y) &= \sum_W \begin{bmatrix} I_x(x,y)^2 & I_x(x,y)I_y(x,y) \\ I_x(x,y)I_y(x,y) & I_y(x,y)^2 \end{bmatrix} \\ &= \begin{bmatrix} \sum_W I_x(x,y)^2 & \sum_W I_x(x,y)I_y(x,y) \\ \sum_W I_x(x,y)I_y(x,y) & \sum_W I_y(x,y)^2 \end{bmatrix} \end{aligned}$$

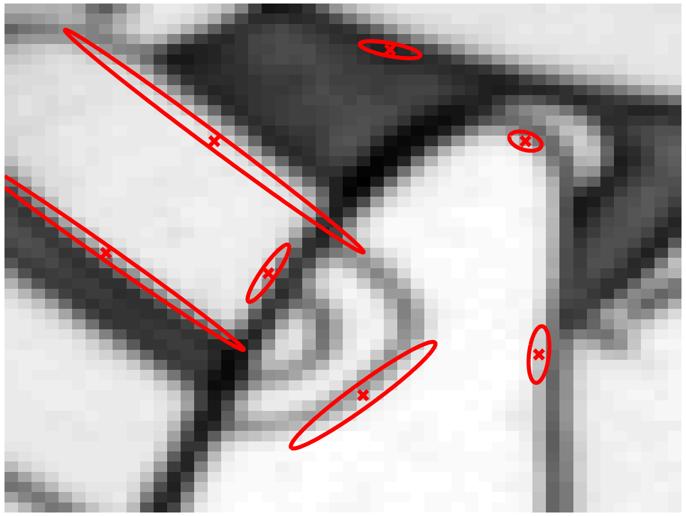
- The autocorrelation function has been approximated by quadratic function

$$c(x,y;\Delta x,\Delta y) \approx [\,\Delta x,\Delta y\,]Q(x,y)\begin{bmatrix}\Delta x\\ \Delta y\end{bmatrix} = [\,\Delta x,\Delta y\,]\begin{bmatrix}A & B\\ B & C\end{bmatrix}\begin{bmatrix}\Delta x\\ \Delta y\end{bmatrix}$$

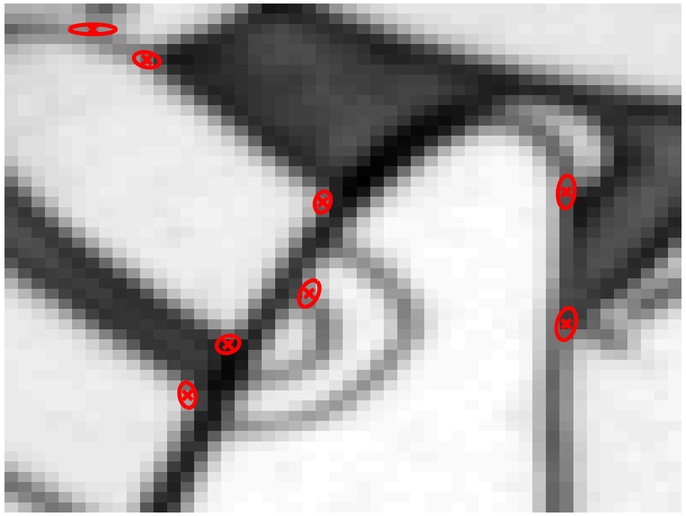
- Elongation and size of the ellipse is given by eigenvalues $\lambda_1, \lambda_2$ of $Q(x,y)$
- The rotation angle of the ellipse is given by eigenvectors of $Q(x,y)$. We don't need it.
- Ellipses with equation $[\,\Delta x,\Delta y\,]Q(x,y)\begin{bmatrix}\Delta x\\ \Delta y\end{bmatrix} = 1$:



flat region
both eigenvalues small

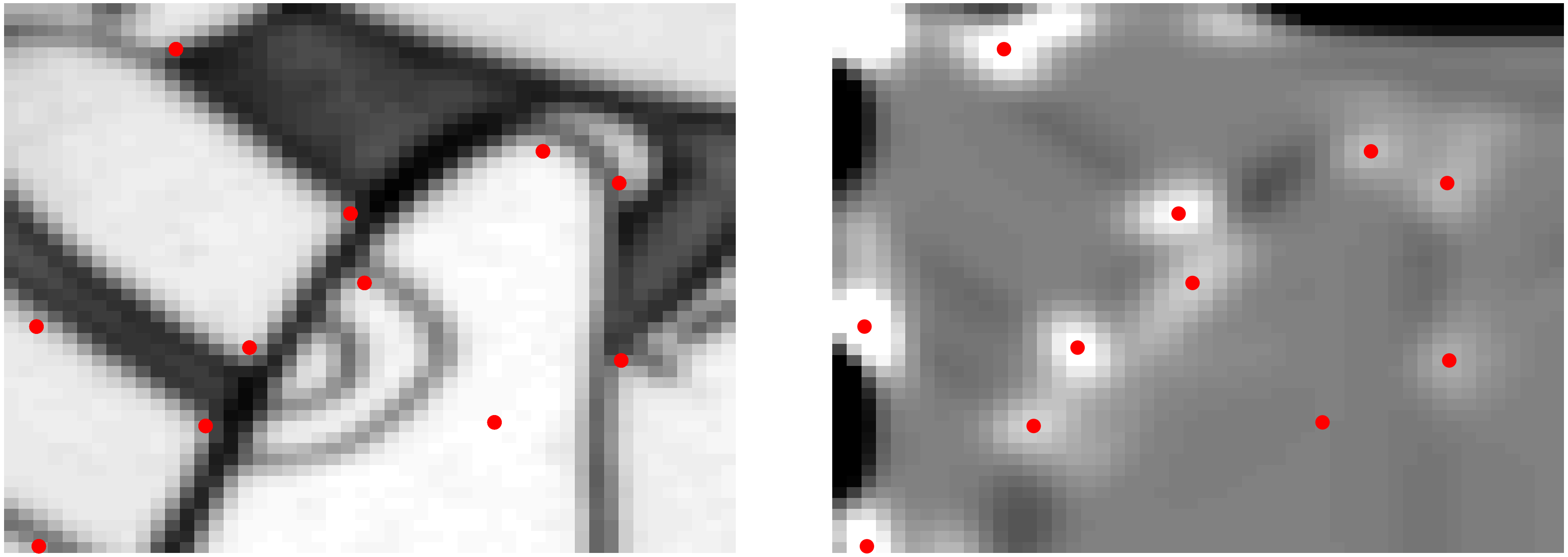


edge
one small, one large



corner
both eigenvalues large

- Characterize 'cornerness' $H(x,y)$ by eigenvalues of $Q(x,y)$:
  - $Q(x,y)$ is symmetric and positive definite $\Rightarrow \lambda_1, \lambda_2 > 0$
  - $\lambda_1 \lambda_2 = \det Q(x,y) = AC - B^2, \quad \lambda_1 + \lambda_2 = \operatorname{trace} Q(x,y) = A + C$
  - Harris suggested: Cornerness $H = \lambda_1 \lambda_2 - 0.04(\lambda_1 + \lambda_2)^2$
  - Image $I(x,y)$ and its cornerness $H(x,y)$:



- Find corner points as **local maxima** of the cornerness $H(x,y)$:
  - Local maximum in image defined as a point greater than its neighbors (in $3 \times 3$ or even $5 \times 5$ neighborhood)

- Compute partial derivatives $I_x(x,y)$, $I_y(x,y)$ by finite differences:

$$I_x(x,y) \approx I(x+1,y) - I(x,y), \quad I_y(x,y) \approx I(x,y+1) - I(x,y)$$

  Before this, it is good (but not necessary) to smooth image with Gaussian with $\sigma \sim 1$, to eliminate noise.

- Compute images

$$A(x,y) = \sum_W I_x(x,y)^2, \quad B(x,y) = \sum_W I_x(x,y) I_y(x,y), \quad C(x,y) = \sum_W I_y(x,y)^2$$

  E.g., image $A(x,y)$ is just the convolution of image $I_x(x,y)^2$ with the Gaussian. Use MATLAB function conv2.

- Compute cornerness $H(x,y)$

- Find local maxima in $H(x,y)$. This can be parallelized in MATLAB by shifting the whole image $H(x,y)$ by one pixel left/right/up/down.