CS-565 Computer Vision

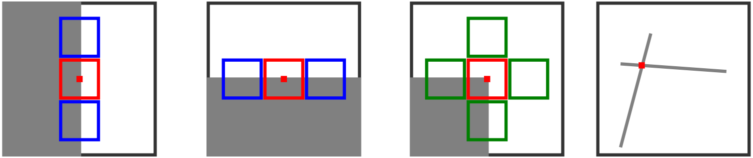
Nazar Khan

Department of Computer Science
University of the Punjab

5. Corner Detection

Corners

- Just like edges, corners are perceptually important.
- More compact summary of an image since corners are fewer than edge pixels.
- A patch around a corner pixel is different from all other surrounding patches.



Vertical edge Horizontal edge Corner Corner

Figure: A patch containing a corner is different from all surrounding patches. Blue squares represent patches similar to the red patch. Green squares represent patches different from the red patch. Author: N. Khan (2021)

How to compare patches
SSD

- For two patches P and Q of size $m \times n$ pixels, their dissimilarity can be computed using a *sum-of-squared distances*

$$SSD(P,Q) = \sum_{i=1}^{m}\sum_{j=1}^{n}(P_{ij} - Q_{ij})^2 \tag{1}$$

- Alternatively, weighted dissimilarity can be computed as

$$SSD(P,Q) = \sum_{i=1}^{m}\sum_{j=1}^{n} w_{ij}(P_{ij} - Q_{ij})^2 \tag{2}$$

  where weight w_{ij} determines the importance of location (i, j).
- For example, Gaussian weights give more importance to the central pixel difference.

Taylor's Approximation for 2D Functions

- Recall that Taylor's approximation for 1D functions is

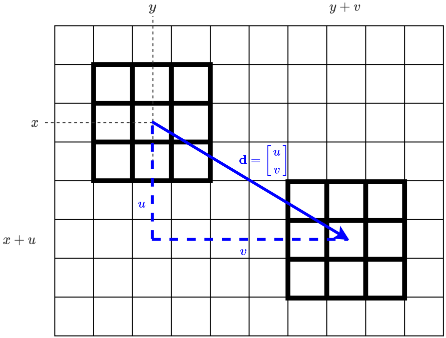
$$f(x+u) = f(x) + \frac{u}{1!}f'(x) + \frac{u^2}{2!}f''(x) + O(u^3) \tag{3}$$

- For 2D functions, a 2nd-order Taylor's approximation is

$$\begin{aligned} f(x+u, y+v) \approx & f(x,y) + \underbrace{\frac{u}{1!}f_x(x,y) + \frac{v}{1!}f_y(x,y)}_{\text{1st-order}} \\ & + \underbrace{\frac{u^2}{2!}f_{xx}(x,y) + \frac{v^2}{2!}f_{yy}(x,y) + \frac{2uv}{2!}f_{xy}(x,y)}_{\text{2nd-order}} \end{aligned}$$

Structure Tensor

- Let us consider patches of size 3×3 although the method works for patches of any size and shape.
- The color value of a pixel displaced from (x, y) by the direction vector $\mathbf{d} = (u, v)^T$ is $I(x + u, y + v)$.



Structure Tensor

- Weighted SSD between a patch at (x, y) and a patch displaced by the direction vector $\mathbf{d} = (u, v)^T$ is computed as

$$SSD(u, v) = \sum_{i=x-1}^{x+1} \sum_{j=y-1}^{y+1} w_{ij}(I(i+u, j+v) - I(i, j))^2$$

- Using a 1st-order Taylor's approximation

$$I(i+u, j+v) \approx I(i, j) + uI_x(i, j) + vI_y(i, j)$$

Structure Tensor

- Weighted SSD can be approximated as

$$
\begin{aligned}
SSD(u,v) &\approx \sum_{i=x-1}^{x+1}\sum_{j=y-1}^{y+1} w_{ij}(I(i+u,j+v)-I(i,j))^2 \\
&= \sum_{i=x-1}^{x+1}\sum_{j=y-1}^{y+1} w_{ij}(I(i,j)+uI_x(i,j)+vI_y(i,j)-I(i,j))^2 \\
&= \sum_{i=x-1}^{x+1}\sum_{j=y-1}^{y+1} w_{ij}(uI_x(i,j)+vI_y(i,j))^2 = \sum_{i=x-1}^{x+1}\sum_{j=y-1}^{y+1} w_{ij}(\mathbf{d}^T\nabla I_{ij})^2 \\
&= \sum_{i=x-1}^{x+1}\sum_{j=y-1}^{y+1} w_{ij}(\mathbf{d}^T\nabla I_{ij})(\mathbf{d}^T\nabla I_{ij})^T = \sum_{i=x-1}^{x+1}\sum_{j=y-1}^{y+1} w_{ij}\mathbf{d}^T\nabla I_{ij}\nabla I_{ij}^T\mathbf{d} \\
&= \mathbf{d}^T\left(\sum_{i=x-1}^{x+1}\sum_{j=y-1}^{y+1} w_{ij}\nabla I_{ij}\nabla I_{ij}^T\right)\mathbf{d} = \mathbf{d}^T A\mathbf{d}
\end{aligned}
$$

Structure Tensor

- The 2×2 matrix A is a weighted summation of the outer-products

$$\nabla I_{ij} \nabla I_{ij}^T = \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}_{ij}$$

- For Gaussian weights, A can be computed via Gaussian convolution

$$A = \begin{bmatrix} G_\rho * I_x^2 & G_\rho * I_x I_y \\ G_\rho * I_x I_y & G_\rho * I_y^2 \end{bmatrix}$$

- In this form A is known as the *structure tensor*.
- The structure tensor plays an important role in other areas of computer vision as well.

Corner Detection via Structure Tensor

- Basic idea: To find if pixel (x, y) is a corner, first find the direction in which patches become most dissimilar.
- That is, the direction $\mathbf{d} = (u, v)^T$ that maximises the SSD $\mathbf{d}^T A\mathbf{d}$ from the patch centered at (x, y).

$$\mathbf{d}^* = \arg\max_{\mathbf{d}} \mathbf{d}^T A\mathbf{d} \text{ s.t. } \|\mathbf{d}\| = 1$$

  where constraint $\|\mathbf{d}\| = 1$ ensures a non-trivial solution.
- Using the method of Lagrange multipliers, $\mathbf{d}^*$ is the eigenvector of A corresponding to the larger eigenvalue (Take-home Quiz 1).
- The SSD in the direction of any eigenvector is the corresponding eigenvalue. Prove it.

Corner Detection via Structure Tensor

- What do the eigenvalues of the structure tensor reveal about the local structure around a pixel?

$$
\begin{aligned}
\lambda_{\text{large}} \approx \lambda_{\text{small}} \approx 0 &\implies \text{flat region} \\
\lambda_{\text{large}} \gg \lambda_{\text{small}} \approx 0 &\implies \text{edge} \\
\lambda_{\text{large}} > \lambda_{\text{small}} \gg 0 &\implies \text{corner}
\end{aligned}
$$

- So a simple corner detection criterion could be $\lambda_{\text{small}} > \tau$.

Corner Detection via Structure Tensor

- But eigenvalue computation is a little expensive.
- Using the facts that
  1. $det(A) = A_{11}A_{22} - A_{12}^2 = \lambda_{\text{large}}\lambda_{\text{small}}$, and
  2. $trace(A) = A_{11} + A_{22} = \lambda_{\text{large}} + \lambda_{\text{small}}$

  popular corner detectors avoid eigenvalue computations.
- Popular corner detectors use a cornerness measure and then a detection criterion.

Method	**Cornerness Measure**	**Detector**
Harris	$\frac{det(A)}{trace(A)}$	$trace(A) > \tau$
Rohr	$det(A)$	$det(A) > \tau$

- To avoid multiple detections, non-maxima suppression must be performed on the cornerness values in 8-neighourhoods or larger.

Corner Detection
Algorithm

Input: Image I.
Parameters:
1) Noise smoothing scale σ,
2) Gradient smoothing scale ρ (should be greater than σ),
3) Threshold τ.

1. Compute Gaussian derivatives at noise smoothing scale σ
$$I_x = \frac{\partial G_\sigma}{\partial x} * I \quad \text{and} \quad I_y = \frac{\partial G_\sigma}{\partial y} * I$$
2. Compute the products
$$I_x^2, \quad I_y^2 \text{ and } \quad I_x I_y$$
3. Smooth the products at gradient smoothing scale ρ
$$G_\rho * I_x^2, \quad G_\rho * I_y^2 \quad \text{and} \quad G_\rho * I_x I_y$$
and construct structure tensor A at every pixel.

Corner Detection
Algorithm

4. Compute cornerness $C(i,j)$ at every pixel as

Harris	Rohr
$C_{ij} = \frac{A_{11}A_{22}-A_{12}^2}{A_{11}+A_{22}}$	$C_{ij} = A_{11}A_{22} - A_{12}^2$

5. Perform non-maxima suppression in 8-neighbourhood on cornerness image C.
6. Find corner pixels by thresholding remaining local maxima via

Harris	Rohr
$trace(A) = A_{11} + A_{22} > \tau$	$det(A) = A_{11}A_{22} - A_{12}^2 > \tau$

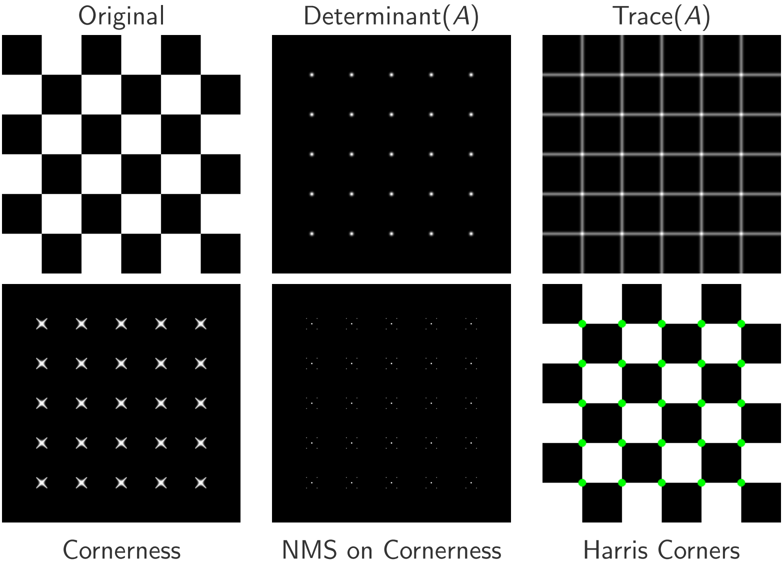


Figure: Harris corners detected with $\sigma = 0.2$, $\rho = 2$ and $\tau =$ 90th percentile of trace values. Author: N. Khan (2018)

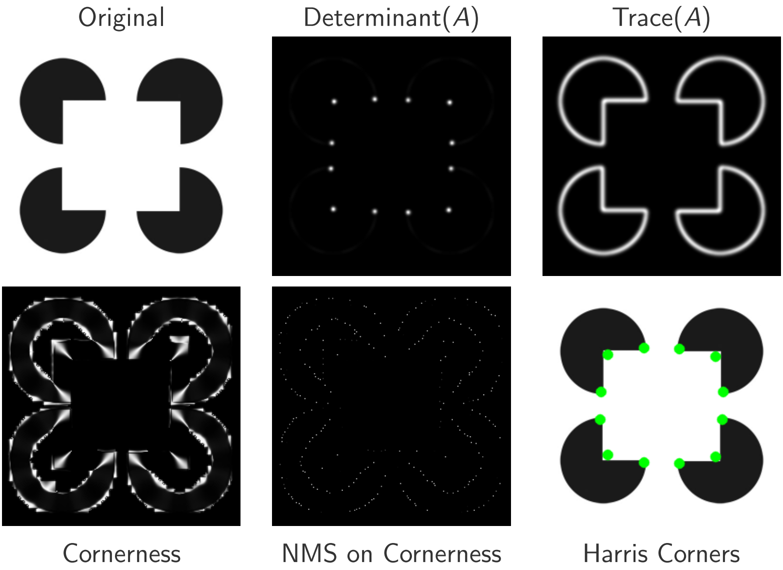


Figure: Harris corners detected with $\sigma = 0.5$, $\rho = 2$ and $\tau = 80$th percentile of trace values. Author: N. Khan (2018)

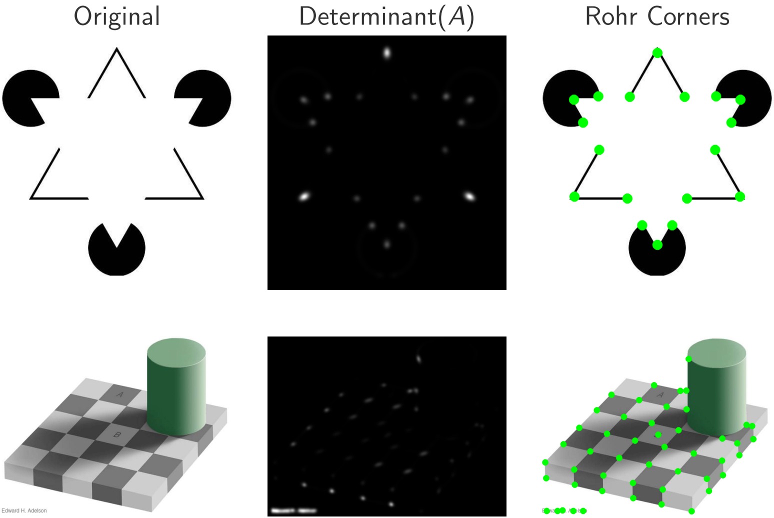


Figure: Corners detected by Rohr's method with $\sigma = 1$, $\rho = 6$ and $\tau = 98$th percentile of determinant values for **top row** and 95th for **bottom row**. Author: N. Khan (2018)

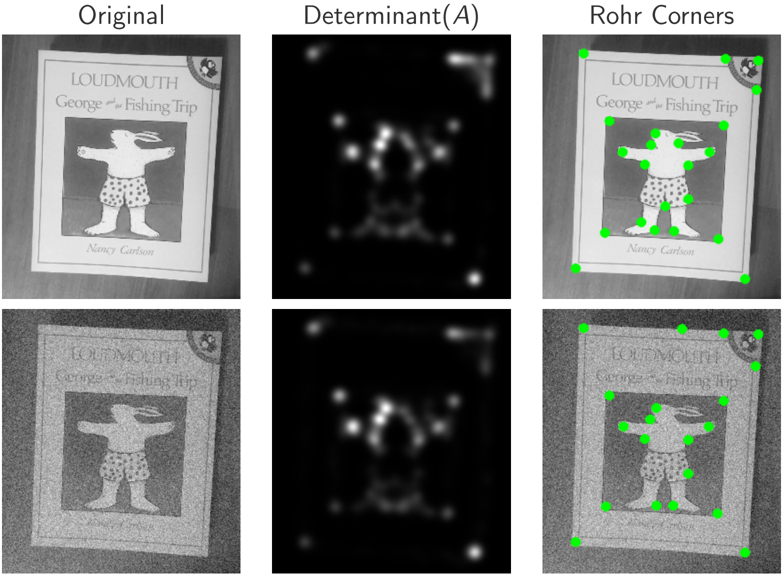
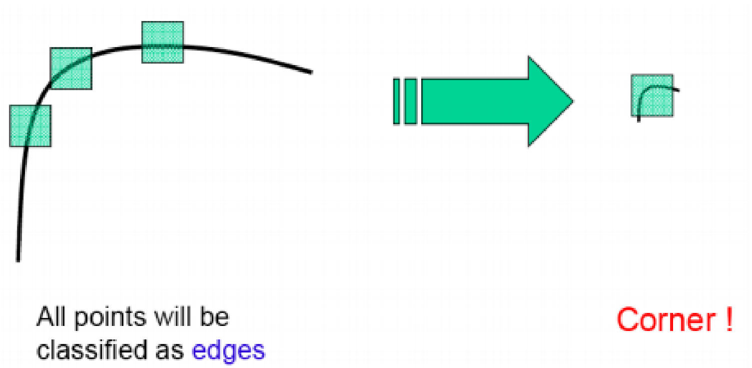


Figure: Corners detected by Rohr's method with $\rho = 6$ and $\tau =$ 95th percentile of determinant values. Noise smoothness scale was $\sigma = 3$ for **top row** and $\sigma = 4$ for **bottom row**. Author: N. Khan (2018)

Corners depend on scale



- Structure tensors and therefore corner detection are not scale invariant.
- Therefore, corner detection should take place at multiple scales.
- This leads to the concept of a *scale space*.

Scale Space via Gaussian Pyramids

Figure: A Gaussian pyramid with 3 levels and 5 smoothing scales. **Top to bottom**: Subsampling in both dimensions by factor 2^i for $i = 0, \dots, 2$. **Left to right**: Gaussian blurring with $\sigma = \sqrt{2}^j \sigma_0$ for $j = 0, \dots, 4$ and $\sigma_0 = \sqrt{2}$. Author: N. Khan (2018)

Scale Space via Gaussian Pyramids

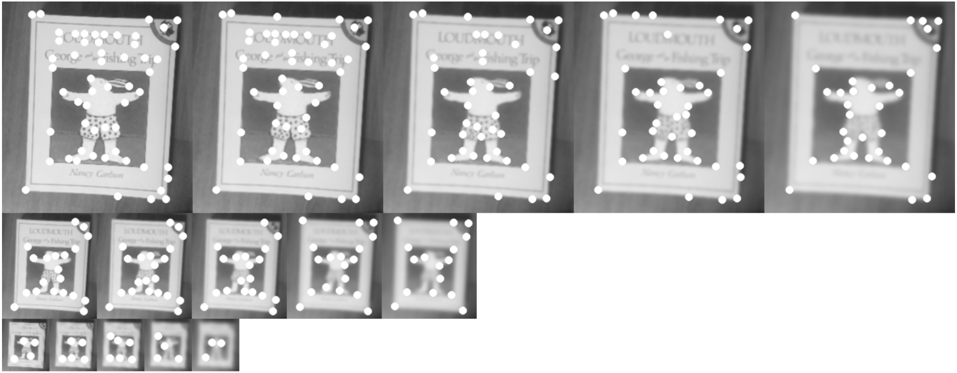


Figure: Corner detection in scale space obtained via Gaussian pyramids. Some corners are detected only at certain resolutions and certain smoothness scales. Corners that *persist across resolutions and smoothness scales* are called strong or stable corners. Author: N. Khan (2018)

Scale Space via Gaussian Pyramids

function makeGaussianPyramid(I,num_levels,num_scales,k,σ_0)
for $i = 0$ to num_levels-1
    $J =$ subsample($I, \frac{1}{2^i}$)
    for $s = 0$ to num_scales-1
        $\sigma = k^s \sigma_0$
        $GP[i, s] = J * G_\sigma$