Deep Learning for Computer Vision

From Edges to Blobs and Corners

Vineeth N Balasubramanian

Department of Computer Science and Engineering
Indian Institute of Technology, Hyderabad

Review: Using Canny Edges to get Straight Lines

- Compute Canny edges
  - Compute $\nabla_x f, \nabla_y f$ (gradients in x, y directions)
  - Compute $\theta = tan^{-1} \frac{\nabla_y f}{\nabla_x f}$
- Assign each edge to one of 8 directions. For each direction d, obtain *"edgelets"*:
  - Find connected components for edge pixels with directions in $\{d-1, d, d+1\}$

Review: Using Canny Edges to get Straight Lines

- Compute Canny edges
  - Compute $\nabla_x f, \nabla_y f$ (gradients in x, y directions)
  - Compute $\theta = tan^{-1}\frac{\nabla_y f}{\nabla_x f}$
- Assign each edge to one of 8 directions. For each direction d, obtain *"edgelets"*:
  - Find connected components for edge pixels with directions in $\{d-1, d, d+1\}$
- Compute straightness and orientation, θ of edgelets using eigenvectors and eigenvalues of second moment matrix, M, of edge pixels

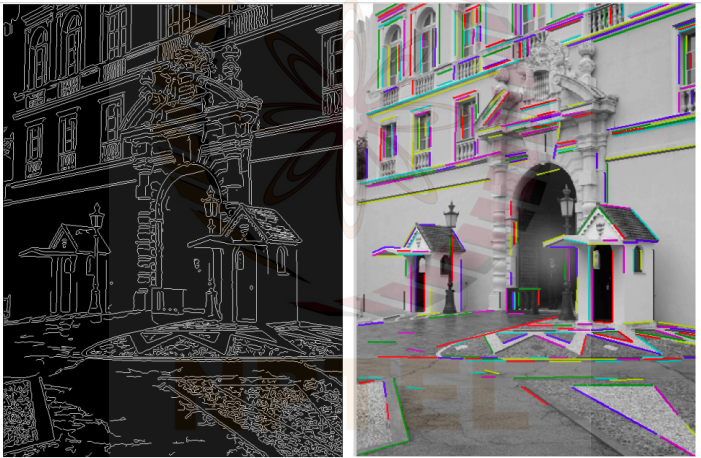
$$M = \begin{bmatrix} \sum(x-\mu_x)^2 & \sum(x-\mu_x)(y-\mu_y) \\ \sum(x-\mu_x)(y-\mu_y) & \sum(y-\mu_y)^2 \end{bmatrix} \quad [\mathbf{v}, \lambda] = eig(M)$$

$$\theta = tan^{-1}(\mathbf{v}_1/\mathbf{v}_0) \text{ where } \mathbf{v}_1 \text{ is the 'larger' eigenvector; straightness} = \lambda_2/\lambda_1$$

- Threshold straightness appropriately and store line segments

Credit: Derek Hoiem

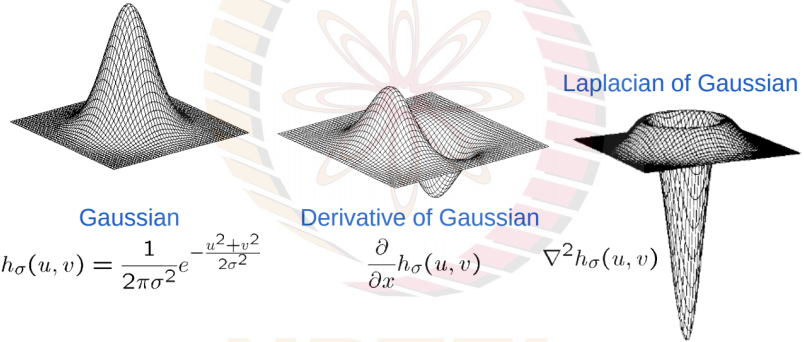
Review: Using Canny Edges to get Straight Lines



Credit: Derek Hoiem

Going Further to a Second Derivative

- What if we took the Laplacian of Gaussian?



$$\text{Laplacian } \nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 y}{\partial y^2} \tag{1}$$

Credit: S Seitz, K Grauman

Laplacian of Gaussian

Example of a 3×3 Laplacian of Gaussian filter:

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

How did we obtain this filter?

Laplacian of Gaussian

Example of a 3×3 Laplacian of Gaussian filter:

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

How did we obtain this filter?

- Discrete approximation of the second derivative:

$$\frac{\partial^2 f}{\partial x^2} = f(x+1, y) + f(x-1, y) - 2f(x, y)$$

$$\frac{\partial^2 f}{\partial y^2} = f(x, y+1) + f(x, y-1) - 2f(x, y)$$

Substituting in the LoG equation: $\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 y}{\partial y^2}$

$$\nabla^2 f = f(x+1, y) + f(x-1, y) + f(x, y+1) + f(x, y-1) - 4f(x, y)$$

- Converting this equation to a filter results in the given LoG matrix.

Laplacian of Gaussian

Original Image

Laplacian

Laplacian of Gaussian

Laplacian of Gaussian

Original Image

Laplacian

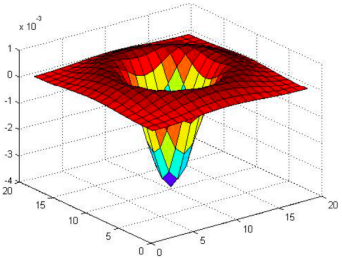
Laplacian of Gaussian

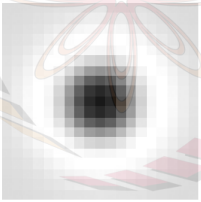
What else can LoG do?

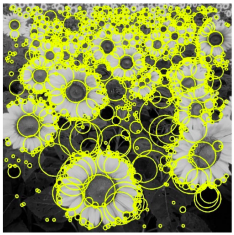
Credit: K Grauman

LoG as a Blob Detector

- Recall that convolution with a filter can be viewed as comparing a little "picture" of what you want to find against all local regions in the image.







- Observing the LoG filter matrix reveals that it is circularly symmetric. Thus it can be used for blob detection!

Credit: S Lazebnik

From Blobs to Corners

- In the following image, what are some interesting features to choose?

Credit: K Grauman, R Urtasun

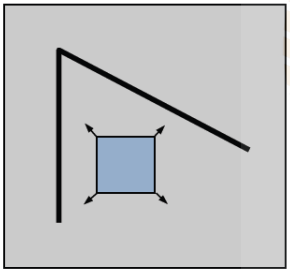
From Blobs to Corners

- Look for image regions that are unusual. How to define "unusual"?
- Textureless patches are nearly impossible to localize.
- Patches with **large contrast changes** (gradients) are easier to localize.
- But straight line segments at a single orientation suffer from the **aperture problem** (we'll see next slide), i.e., it is only possible to align the patches along the direction normal to the edge direction.
- Gradients in at least two (significantly) different orientations are the easiest, e.g., corners.

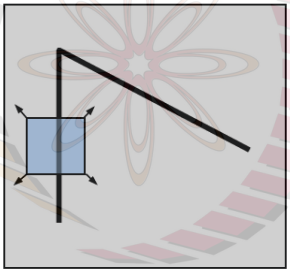
Credit: R Urtasun

From Blobs to Corners

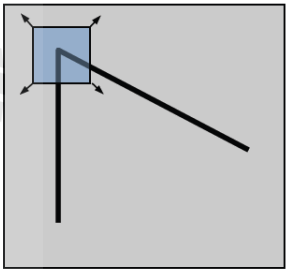
- Consider a small window of pixels. How does the window change when you shift it?



"flat" region:
no change in all directions



"edge":
no change along the edge direction



"corner":
significant change in all directions

Credit: S Seitz, D Frolova, D Simakova, R Urtasun

Autocorrelation

- In the previous slide, how to quantify the "significant" change of the window?

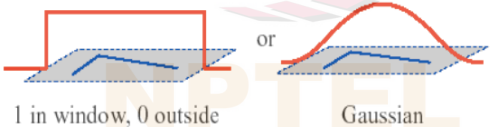
Autocorrelation

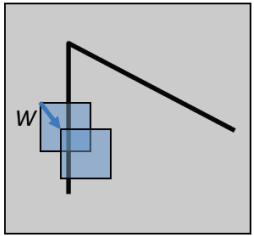
- In the previous slide, how to quantify the "significant" change of the window?
- Answer: **Autocorrelation function**. Compute the sum of squared differences between pixel intensities with respect to small variations in the image patch position.

$$E_{AC}(\Delta \mathbf{u}) = \sum_{x,y} w(\mathbf{p}_i)[I(\mathbf{p}_i + \delta u) - I(\mathbf{p}_i)]^2$$

where $\mathbf{p}_i = (x, y)$, a particular position on the image.

Window function $w(x,y)$ =





Credit: R Urtasun

Computing Autocorrelation

- Using a Taylor Series expansion $I(\mathbf{p}_i + \Delta\mathbf{u}) = I(\mathbf{p}_i) + \nabla I(\mathbf{p}_i)\Delta\mathbf{u}$ with the image gradient

$$\nabla I(\mathbf{p}_i) = \left(\frac{\delta I(\mathbf{p}_i)}{\delta x}, \frac{\delta I(\mathbf{p}_i)}{\delta y} \right)$$

Computing Autocorrelation

- Using a Taylor Series expansion $I(\mathbf{p}_i + \Delta\mathbf{u}) = I(\mathbf{p}_i) + \nabla I(\mathbf{p}_i)\Delta\mathbf{u}$ with the image gradient

$$\nabla I(\mathbf{p}_i) = \left(\frac{\delta I(\mathbf{p}_i)}{\delta x}, \frac{\delta I(\mathbf{p}_i)}{\delta y} \right)$$

- Autocorrelation can be approximated as:

$$\begin{aligned} E_{AC}(\Delta\mathbf{u}) &= \sum_{x,y} w(\mathbf{p}_i)[I(\mathbf{p}_i + \delta u) - I(\mathbf{p}_i)]^2 \\ &\approx \sum_{x,y} w(\mathbf{p}_i)[I(\mathbf{p}_i) + \delta I(\mathbf{p}_i)\delta u - I(\mathbf{p}_i)]^2 \\ &= \sum_{x,y} w(\mathbf{p}_i)[\delta I(\mathbf{p}_i)\delta u]^2 \\ &= \Delta\mathbf{u}^T \mathbf{A} \Delta\mathbf{u} \end{aligned}$$

Credit: R Urtasun

Computing Autocorrelation

- The autocorrelation is $E_{AC}(\Delta\mathbf{u}) = \Delta\mathbf{u}^T\mathbf{A}\Delta\mathbf{u}$, with

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

- The weighted summations have been replaced with discrete convolutions with the weighting kernel w.
- The eigenvalues of $\mathbf{A}$ reveal the amount of intensity change in the two principal orthogonal gradient directions in the window.

$$\mathbf{A} = \mathbf{U} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \mathbf{U}^T \quad \text{with} \quad \mathbf{A}\mathbf{u}_i = \lambda \mathbf{u}_i$$

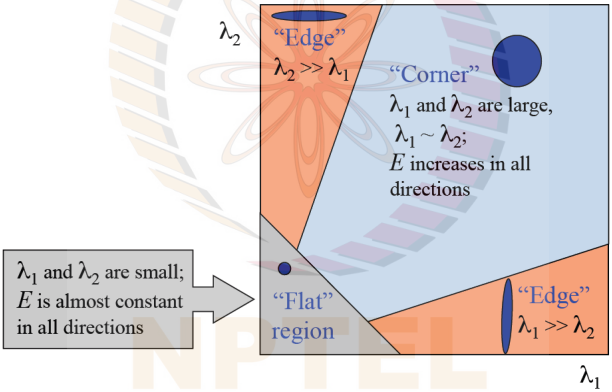
Credit: R Urtasun

Computing Autocorrelation

- How do the eigenvalues determine if an image point is a corner?

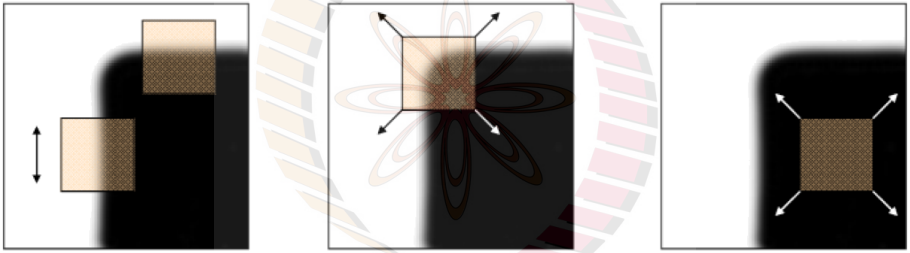
Computing Autocorrelation

- How do the eigenvalues determine if an image point is a corner?



Credit: N Snavely, R Urtasun

Computing Autocorrelation



"edge":

$\lambda_1 >> \lambda_2$

$\lambda_2 >> \lambda_1$

"corner":

λ_1 and λ_2 are large,

$\lambda_1 \sim \lambda_2$;

"flat" region

λ_1 and λ_2 are small;

Credit: K Grauman, R Urtasun

Harris Corner Detector

- Compute gradients at each point in the image.
- Compute **A** for each image window to get its cornerness scores.
- Compute the eigenvalues/compute the following function M_c

$$M_c = \lambda_1\lambda_2 - \kappa(\lambda_1 + \lambda_2)^2 = det(A) - \kappa trace^2(A)$$

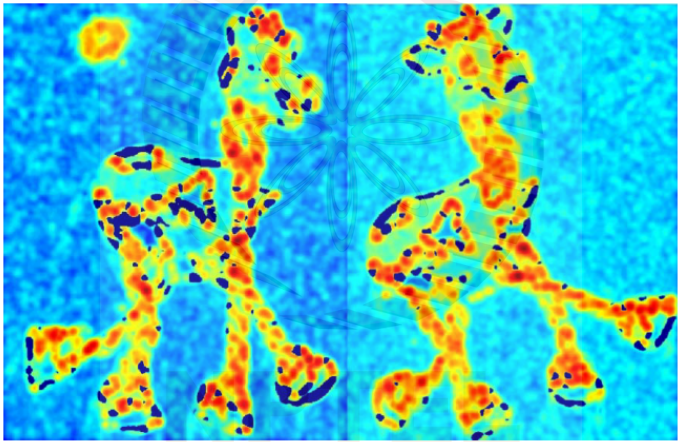
- Find points whose surrounding window gave larger cornerness response ($M_c >$threshold)
- Take points of local maxima, perform non-maximum suppression.

Credit: R Urtasun

Harris Corner Detector: Example

Credit: K Grauman, R Urtasun

Computing Cornerness

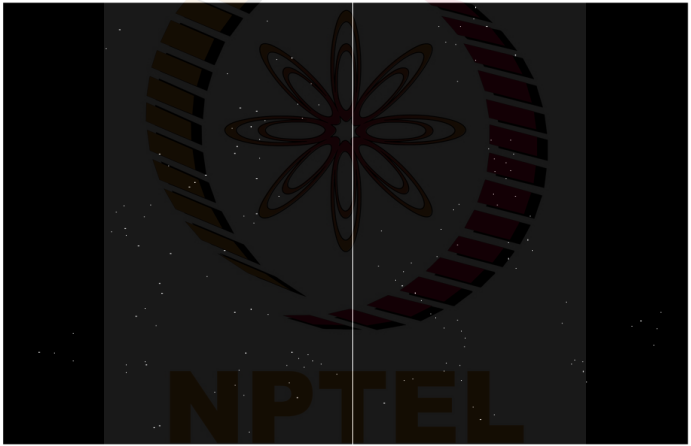


Credit: K Grauman, R Urtasun

Finding High Response

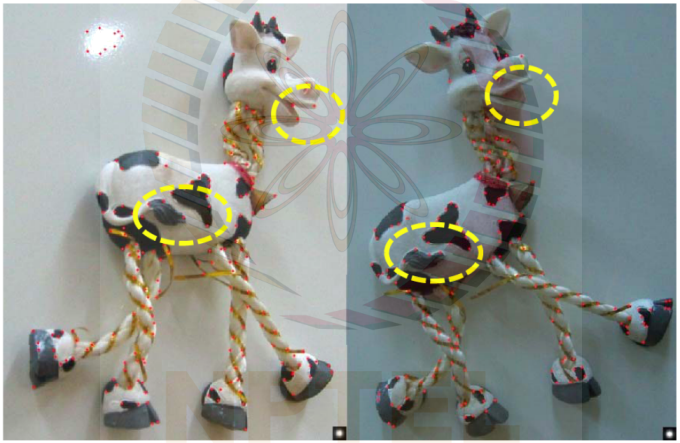
Credit: K Grauman, R Urtasun

Non-max Suppression



Credit: K Grauman, R Urtasun

Results



Credit:K Grauman, R Urtasun

Harris Corner Detector: Variants

- Harris and Stephens '88 is rotationally invariant and downweights edge-like features where $\lambda_1 >> \lambda_0$.

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

- Triggs '04 suggested $\lambda_0 - \alpha\lambda_1$.
- Brown et al, '05 use harmonic mean:

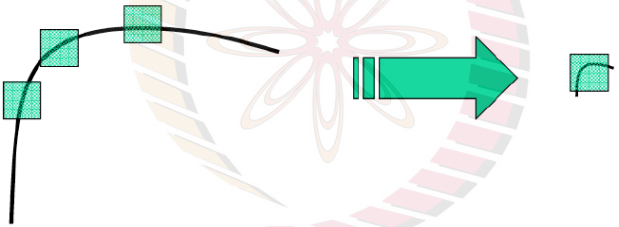
$$\frac{det(\mathbf{A})}{trace(\mathbf{A})} = \frac{\lambda_0\lambda_1}{\lambda_0 + \lambda_1}$$

  which is smoother when $\lambda_0 \approx \lambda_1$

Credit: R Urtasun

Harris Corner Detector: Properties

- Scale-invariant?



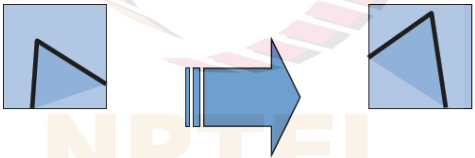
Credit: R Urtasun

Harris Corner Detector: Properties

- Rotation-invariant?

$$\mathbf{A} = w * \begin{bmatrix} I_x^2 & I_x I_y \\ I_y I_x & I_y^2 \end{bmatrix} = \mathbf{U} \begin{bmatrix} \lambda_0 & 0 \\ 0 & \lambda_1 \end{bmatrix} \mathbf{U}^T \quad \text{with} \quad \mathbf{A}\mathbf{u}_i = \lambda_i \mathbf{u}_i$$

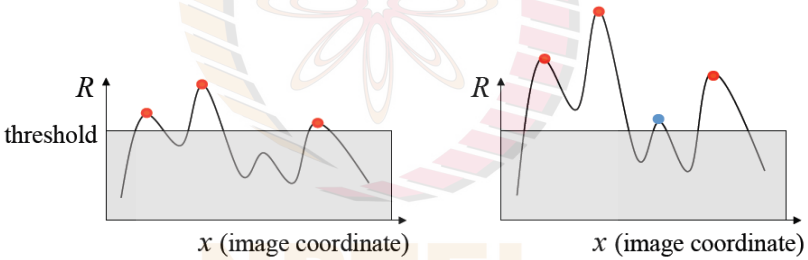
- Relative cornerness remains the same!



Credit: N Snavely, R Urtasun

Harris Corner Detector: Properties

- Photometric change: Affine intensity change $I = aI + b$?
- Only derivatives are used, so it's invariant to shift $I = I + b$.
- What about intensity scale?



Partially invariant to affine intensity change

Credit: K Grauman, R Urtasun

Homework

Reaadings

- Chapter 2, Szeliski, *Computer Vision: Algorithms and Applications*

Questions: Linear Algebra review

- Show that the trace of a matrix is the sum of its eigenvalues.
- Show that the determinant of a matrix is the product of its eigenvalues.

References

C. Harris and M. Stephens. "A Combined Corner and Edge Detector". In: *Proceedings of the 4th Alvey Vision Conference*. 1988, pp. 147–151.

Bill Triggs. "Detecting Keypoints with Stable Position, Orientation, and Scale under Illumination Changes". In: *Computer Vision - ECCV 2004*. Ed. by Tomás Pajdla and Jiří Matas. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 100–113.

M. Brown, R. Szeliski, and S. Winder. "Multi-image matching using multi-scale oriented patches". In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. Vol. 1. 2005, 510–517 vol. 1.

Richard Szeliski. *Computer Vision: Algorithms and Applications*. Texts in Computer Science. London: Springer-Verlag, 2011.

David Forsyth and Jean Ponce. *Computer Vision: A Modern Approach*. 2 edition. Boston: Pearson Education India, 2015.