Seminar 3 - Lectures 5, 6 and 7
Computer Vision 1, Master AI, 2025

Arnoud Visser, Martin Oswald, Qi Bi, Roan van Blanken

1 Exercise 1: Edges and Corners

In this exercise, we explore key structures in images: edges and corners. At these interest points, there are sharp differences in brightness, which can indicate various phenomena, such as changes in depth, material, and illumination. Edges and corners are fundamental features in classical computer vision and are widely used in more complex image processing algorithms.

We consider two simple image patches, A and B, from a random 4-bit image shown below. The patches are highlighted in yellow. The values in the cells represent the intensity values in the range $[0, 15]$ at different locations in the image.

$A =$

0	0	6	13	12
0	0	6	13	12
0	0	6	13	12
0	0	6	13	12
0	0	6	13	12

$B =$

0	0	0	0	0
0	0	0	0	0
12	7	7	0	0
12	12	7	0	0
12	12	10	0	0

Q.1.a What structures do you see in image patches A and B?

Answer: Image patch A shows a single vertical edge running along the y-direction of the patch. Image patch B contains two edges: one horizontal and one vertical, which meet to form a corner in the middle of the patch.

For the next couple of questions, we use the following partial derivatives: $I_x = \frac{\partial I}{\partial x}$ and $I_y = \frac{\partial I}{\partial y}$. To compute these, we apply the following simple correlation kernels. These kernels are already flipped, and we omit the normalization constant $\frac{1}{2}$:

Correlation kernel (x-direction):

-1	0	1

Correlation kernel (y-direction):

-1
0
1

Q.1.b Cross-correlate patch A with the derivative kernel. Compute the patch derivatives in x and y directions. Compute the gradient magnitude M_A.

Answer: Remember, the kernels given in the question are correlation kernels, not the formally correct differentiation kernels. We simplify the exercise by using unnormalized flipped differentiation kernels.

The patch derivatives are computed as follows:

$$A_x = \begin{array}{|c|c|c|}\hline 6 & 13 & 6 \\ \hline 6 & 13 & 6 \\ \hline 6 & 13 & 6 \\ \hline \end{array} \quad \text{so} \quad A_x^2 = \begin{array}{|c|c|c|}\hline 36 & 169 & 36 \\ \hline 36 & 169 & 36 \\ \hline 36 & 169 & 36 \\ \hline \end{array}$$

Similarly, for the y-direction derivative:

$$A_y = \begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline 0 & 0 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array} \quad \text{so} \quad A_y^2 = \begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline 0 & 0 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array}$$

Now, to compute the gradient magnitude M_A:

$$M_A = \sqrt{A_x^2 + A_y^2} = \begin{array}{|c|c|c|}\hline \sqrt{36} & \sqrt{169} & \sqrt{36} \\ \hline \sqrt{36} & \sqrt{169} & \sqrt{36} \\ \hline \sqrt{36} & \sqrt{169} & \sqrt{36} \\ \hline \end{array} = \begin{array}{|c|c|c|}\hline 6 & 13 & 6 \\ \hline 6 & 13 & 6 \\ \hline 6 & 13 & 6 \\ \hline \end{array}$$

The Harris Transform is used to estimate the *cornerness* H of an image patch. This measure helps in detecting corners, edges, and flat regions by analyzing the intensity variations within the patch. A simplified definition of the *cornerness* is given by:

$$H = \lambda_1 \lambda_2 - 0.04(\lambda_1 + \lambda_2)^2$$

where λ_1 and λ_2 are the eigenvalues of the structure tensor matrix M. An alternative formulation, which avoids computing eigenvalues directly, is given by:

$$H = \det(M) - 0.04 \cdot (\text{trace}(M))^2$$

where:

- $\det(M)$ is the determinant of matrix M,
- $\text{trace}(M)$ is the sum of the diagonal elements of matrix M (i.e., $M_{11} + M_{22}$).

Matrix M, known as the structure tensor, encapsulates information about the gradient of the image patch and is defined as:

$$M = \sum_{(x,y) \in W} \begin{bmatrix} I_x(x,y)^2 & I_x(x,y)I_y(x,y) \\ I_x(x,y)I_y(x,y) & I_y(x,y)^2 \end{bmatrix}$$

Here:

- $I_x(x,y)$ and $I_y(x,y)$ represent the partial derivatives of the image intensity with respect to x and y at pixel location (x,y),
- W is the image patch (a window of pixels) over which the structure tensor M is computed.

In essence, the Harris Transform evaluates the cornerness of a patch by analyzing how the intensity gradients I_x and I_y vary within the patch. The matrix M captures this information, and the value of H helps classify the patch as an edge, a flat region, or a corner, depending on the eigenvalues of M.

Q.1.c Compute the Harris corner response for patch A. What can you conclude based on your result?

Answer: Use the patch derivatives from the previous question:

$$\sum_{(x,y) \in A} I_x(x,y)^2 = 6^2 + 13^2 + 6^2 + 6^2 + 13^2 + 6^2 + 6^2 + 13^2 + 6^2$$

$$= 6 \times 36 + 3 \times 169 = 216 + 507 = 723$$

$$\sum_{(x,y) \in A} I_x(x,y) I_y(x,y) = 6 \times 0 + 13 \times 0 + 6 \times 0 + 6 \times 0 + 13 \times 0 + 6 \times 0 + 6 \times 0 + 13 \times 0 + 6 \times 0 = 0$$

$$\sum_{(x,y) \in A} I_y(x,y)^2 = 9 \times 0^2 = 0$$

The structure tensor matrix M is then:

$$M = \begin{bmatrix} 723 & 0 \\ 0 & 0 \end{bmatrix}$$

Now, using the Harris *cornerness* equation, we compute H:

$$H = \det(M) - k \cdot (\text{trace}(M))^2 = 723 \times 0 - 0.04 \times (723)^2 = -20909.16$$

Since H is large and negative, we can conclude that patch A contains an edge. If H were large and positive, we would conclude that patch A contains a corner. If the value of H had been close to zero, we could have concluded that the patch was homogeneous (i.e., no edge or corner was present).

Q.1.d Now repeat the exercise for patch B. Compute the gradient magnitude and the Harris corner response of image patch B. What can you conclude based on your results?

Answer: We start by computing the patch derivatives for patch B in the x- and y-directions:

$$B_x = \begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline -5 & -7 & -7 \\ \hline -5 & -12 & -7 \\ \hline \end{array} \quad \text{so} \quad B_x^2 = \begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline 25 & 49 & 49 \\ \hline 25 & 144 & 49 \\ \hline \end{array}$$

Similarly, for the y-direction derivative:

$$B_y = \begin{array}{|c|c|c|}\hline 7 & 7 & 0 \\ \hline 12 & 7 & 0 \\ \hline 5 & 3 & 0 \\ \hline \end{array} \quad \text{so} \quad B_y^2 = \begin{array}{|c|c|c|}\hline 49 & 49 & 0 \\ \hline 144 & 49 & 0 \\ \hline 25 & 9 & 0 \\ \hline \end{array}$$

Now, to compute the gradient magnitude M_B:

$$M_B = \sqrt{B_x^2 + B_y^2} = \begin{array}{|c|c|c|}\hline \sqrt{49} & \sqrt{49} & 0 \\ \hline \sqrt{169} & \sqrt{98} & \sqrt{49} \\ \hline \sqrt{50} & \sqrt{153} & \sqrt{49} \\ \hline \end{array} = \begin{array}{|c|c|c|}\hline 7 & 7 & 0 \\ \hline 13 & \sqrt{98} & 7 \\ \hline \sqrt{50} & \sqrt{153} & 7 \\ \hline \end{array}$$

We now use these results to compute the Harris corner response H:

$$\sum_{(x,y)\in B} I_x(x,y)^2 = 0+0+0+25+49+49+25+144+49 = 341$$

$$\sum_{(x,y)\in B} I_x(x,y)I_y(x,y) = 0+0+0-60-49+0-25-36 = -170$$

$$\sum_{(x,y)\in B} I_y(x,y)^2 = 49+49+0+144+49+0+25+9+0 = 325$$

The structure tensor matrix M is then:

$$M = \begin{bmatrix} 341 & -170 \\ -170 & 325 \end{bmatrix}$$

Now, using the Harris cornerness equation, we compute H:

$$H = \det(M) - k \cdot (\text{trace}(M))^2 = \left(341 \times 325 - (-170)^2\right) - 0.04 \times (341+325)^2$$

$$H = 81925 - 17742.24 = 64182.76$$

Since H is large and positive, we can conclude that patch B contains a corner.

Q.1.e Alternatively, we could use the eigenvalues of structure tensor M. These eigenvalues can also help us make a statement about the *cornerness* of the patch. Compute the eigenvalues of the structure tensor M for patch B. What do these values indicate?

Answer: To compute the eigenvalues, we first find the characteristic equation of the matrix M by solving $\det(\mathcal{M} - \lambda I) = 0$, where I is the identity matrix. The steps are as follows:

$$M = \begin{bmatrix} 341 & -170 \\ -170 & 325 \end{bmatrix}$$

$$\det(\mathcal{M} - \lambda I) = \det\left(\begin{bmatrix} 341 & -170 \\ -170 & 325 \end{bmatrix} - \lambda \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\right) = 0$$

This simplifies to:

$$\det\left(\begin{bmatrix} 341-\lambda & -170 \\ -170 & 325-\lambda \end{bmatrix}\right) = 0$$

Now, expanding the determinant:

$$(341-\lambda)(325-\lambda) - (-170)^2 = 0$$

Simplifying further:

$$\lambda^2 - 666\lambda + 81925 = 0$$

We solve this quadratic equation using the quadratic formula:

$$\lambda = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

where $a = 1$, $b = -666$, and $c = 81925$. Substituting these values:

$$\lambda = \frac{666 \pm \sqrt{666^2 - 4 \times 1 \times 81925}}{2 \times 1}$$

$$\lambda = \frac{666 \pm \sqrt{443556 - 327700}}{2}$$

$$\lambda = \frac{666 \pm 340.38}{2}$$

Thus, the two eigenvalues are:

$$\lambda_1 = \frac{666 + 340.38}{2} = 503.19 \quad \text{and} \quad \lambda_2 = \frac{666 - 340.38}{2} = 162.81$$

The eigenvalues are $\lambda_1 = 503.19$ and $\lambda_2 = 162.81$.

Since both eigenvalues are relatively large, it indicates that there is significant variation in the image intensity in both directions (corresponding to the eigenvectors of the matrix). This suggests that patch B contains a *corner*, as both directions exhibit strong gradients, which is a key characteristic of corners.

2 Exercise 2: Histogram of Gradients

Consider the following image patch (I):

$$I_{\mathrm{A}} = \begin{array}{|c|c|c|c|c|}\hline 1 & 1 & 1 & 1 & 1 \\ \hline 1 & 1 & 1 & 1 & 0 \\ \hline 1 & 1 & 1 & 0 & 0 \\ \hline 1 & 1 & 0 & 0 & 0 \\ \hline 1 & 0 & 0 & 0 & 0 \\ \hline \end{array}$$

Q.2.a Compute the gradient G_x and G_y, using image filters. Which filters do you use?

Answer: We use $F_x = [1, 0, -1]$ and $F_y = F_x^\top$. The gradients are computed as follows:

$$G_x = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \end{bmatrix} \quad \text{and} \quad G_y = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \end{bmatrix}$$

Does it matter if we use cross-correlation or convolution? No, it is only a sign flip in this case. The results remain the same. Could we also use $F_x = [1, -1]$? Yes, but we need to remain consistent throughout.

Q.2.b Compute the gradient magnitude

Answer: The gradient magnitude is computed as follows:

$$M = \sqrt{(G_x)^2 + (G_y)^2}$$

For the given patch:

$$M = \begin{bmatrix} 0 & 0 & \sqrt{2} \\ 0 & \sqrt{2} & \sqrt{2} \\ \sqrt{2} & \sqrt{2} & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1.414 \\ 0 & 1.414 & 1.414 \\ 1.414 & 1.414 & 0 \end{bmatrix}$$

Q.2.c Compute the gradient orientation (in degrees)

Answer: The gradient orientation θ is computed as:

$$\theta = \arctan\left(\frac{G_y}{G_x}\right)\frac{180}{\pi}$$

Using the assumptions that:

$$\frac{0}{0} \equiv 0, \quad \frac{1}{0} \equiv \infty, \quad \frac{-1}{0} \equiv -\infty$$

The gradient orientation is:

$$\theta = \begin{bmatrix} 0 & 0 & 45 \\ 0 & 45 & 45 \\ 45 & 45 & 0 \end{bmatrix}$$

Q.2.d Compute the HoG descriptor, using a 9-bin histogram

Answer: The HoG descriptor combines the direction of the gradient with its magnitude. For real HoG descriptors, unsigned gradients are used (ranging from 0 to 180 degrees), and gradients are often shared over bins. For this patch, the computed HoG descriptor is:

$$\begin{aligned}\text{HoG descriptor (No weight)} &= [4, 0, 5, 0, 0, 0, 0, 0, 0] \\ \text{HoG descriptor (Weighted by magnitude)} &= [4 \cdot 0, 0, 5 \cdot \sqrt{2}, 0, 0, 0, 0, 0, 0] \\ &= [0, 0, 7.071, 0, 0, 0, 0, 0, 0]\end{aligned}$$

Q.2.e Is the HoG descriptor invariant to overall lighting, *i.e.*, is the HoG descriptor of $I = 2 * I_A$ equal to the HoG descriptor of I_A?

Answer: No, the HoG descriptor is invariant to additive lighting ($I = I_A + c$), but not to scaling. For example, if the image is scaled by a factor of 2, the HoG descriptor becomes:

$$\text{HoG descriptor for scaled image} = [0, 0, 14.142, 0, 0, 0, 0, 0, 0]$$

This is different from the original descriptor, which demonstrates that HoG is not invariant to scaling.

Q.2.f Is the HoG descriptor invariant to rotation, *i.e.*, is the HoG descriptor of I_A equal to the HoG descriptor of I_A after a 90-degree rotation?

Answer: No, the HoG descriptor is not invariant to rotation. For example, after rotating the image by 90 degrees, the HoG descriptor becomes:

$$\text{HoG descriptor for rotated image} = [0, 0, 0, 0, 0, 0, 7.071, 0, 0]$$

This is different from the original descriptor, which demonstrates that HoG is not invariant to rotation.

3 Exercise 3: SIFT

The SIFT (Scale Invariant Feature Transform) has become a popular technique for detecting and representing features for structure-from-motion and object recognition tasks.

Q.3.a Explain how feature locations are determined in the SIFT operator.

Answer: Features are located by searching for local extrema in both location and scale within the Difference of Gaussian (DoG) scale space. The DoG function approximates the response of a scale-normalized Laplacian operator, which functions as a match filter for blob-like structures. Local extrema in the DoG scale space are used to determine the feature locations.

Q.3.b Explain how the SIFT operator achieves invariance to (i) illumination; (ii) scale; (iii) rotation.

Answer: Illumination invariance is achieved by using local maxima, which are invariant to rescaling of the DoG function. Invariance to scale is obtained by associating feature points with structures (blobs) that are searched across all locations and scales. Rotation invariance is achieved by aligning the feature point's orientation with the dominant orientation in the region surrounding the feature point.

Q.3.c A SIFT descriptor for a 16×16 image window $\mathbf{W}$ is a vector of 128 numbers, which can be thought of as being grouped in 16 consecutive groups of 8 numbers each. What does each group of 8 numbers represent? Explain in one or two brief, clear, and accurate sentences.

Answer: The 16×16 image window $\mathbf{W}$ is divided into a 4×4 grid, where each section is a 4×4 square. Each group of 8 numbers in the SIFT descriptor represents a histogram of image gradient orientations in one of the 16 sections, with each orientation weighted by the corresponding gradient magnitude. The gradient magnitude is proportionally split between two adjacent bins depending on how close the orientation is to the center of each bin.

4 Exercise 4: Optical Flow

Q.4.a For the Lukas-Kanade optical flow technique, mention two of the main assumptions

Answer: With the Lukas-Kanade technique one can estimate the flow vector between two images. The main assumptions are:

- Color constancy / brightness constancy
- Small Motion
- Pixel neighbors have similar flow (u, v)
- High textured region (both eigenvalues of moments matrix A^TA are large)

Q.4.b How can we use optical flow for traffic monitoring i.e. ticketing higher than allowed speeds?

Answer: We can use the vector magnitude of the vector which shows the optical flow from the first frame that a car appears in the camera field to the next frame. Then by using either the distance to the camera we can compute the actual velocity or by a canonical optical flow vector, we can calculate the speed of each car.

Q.4.c How could you use optical flow to add effects like slow motion or video smoothing and when this doesn't work?

Answer: Let optical flow for pixel p in frame f_t to frame f_{t+1} be $o = (u, v)$. To generate pixels for frame $f_{t+\frac{1}{2}}$, we simply consider two optical vectors $o_1 = o_2 = (\frac{u}{2}, \frac{v}{2})$. From the definition of optical flow, we know that brightness of $p + o$ in frame f_{t+1} is the same as brightness of p in frame f_t, so to generate frame $f_{t+\frac{1}{2}}$ we simply consider the brightness of $p + o_1$ in this frame as brightness of p in frame f_t. Also the optical flow for each point $p + o_1$ would be o_2. Note that since for frame f_{t+1} we can consider the optical flow for pixel $p + o$ as $-o$ (from frame f_{t+1} to frame f_t i.e. in reverse order), we can calculate in the same manner this time using frame f_{t+1}. Optical flow for generating frames doesn't work if the optical flow vectors are not reliable (change in lightning, extreme motion, change of shots) and will lead to artifacts.

Q.4.d How can we use optical flow for shaking camera and image stabilization?

Answer: Let's consider camera motion as vector s. If camera was stable each pixel p_{ij} had and optical flow o_{ij}. Now from optical flow definition, we know that this optical flow in shaking mode will be $s + o_{ij}$. if we consider the average optical flow of the frames we will have the vector $s + \frac{\sum_{i,j} o_{ij}}{\text{Total number of pixels}}$. If there are few motions in the in stable camera mode scene, the second part becomes negligible, So we can find the total motion of camera from frame to frame by averaging over all optical flow vectors. Then we can decrease s from current optical flows to achieve o_{ij}s, finally we simply generate the next frame using the stable mode flows.

Q.4.e How we can use optical flows to estimate depth?

Answer: When a point is closer to camera its optical flow is larger than when it is farther. If we consider these difference in magnitudes we can estimate the depth of a scene (up to an unknown scale factor if the camera calibration is unknown).

Q.4.f What is the difference between dense and sparse optical flows? What is the pros and cons of each?

Answer: Sparse optical flows only produces the flow vectors for keypoints while dense optical flows consider the flows for all pixels. Sparse optical flow methods are typically faster on single cores, dense methods tend to be more accurate.

5 Exercise 5: 2D Image Warping

Q.5.a For 2D transformations, write down an example transformation matrix for a rotation for homogeneous coordinates.

Answer: A 2D transformation matrix for homogeneous coordinates is 3 x 3. For some angle θ, the rotation matrix looks like:

$$\begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{1}$$

Q.5.b For the same dimensions, give an example *affine* transformation.

Answer: An affine transformation is a combination of rotation, translation, scale, shear and looks like:

$$\begin{pmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{pmatrix} \tag{2}$$

Q.5.c Explain difference between forward and backward transformation

Answer: Any geometric transformation, including an affine transformation, can be implemented as forward or backward mapping. The forward mapping iterates over each pixel of the input image, computes new coordinates for it, and copies its value to the new location. But the new coordinates may not lie within the bounds of the output image and may not be integers. The former problem is easily solved by checking the computed coordinates before copying pixel values. The second problem is solved by assigning the nearest integers to x' and y' and using these as the output coordinates of the transformed pixel. The problem is that each output pixel may be addressed several times or not at all (the latter case leads to "holes" where no value is assigned to a pixel in the output image). The backward mapping iterates over each pixel of the output image and uses the inverse transformation to determine the position in the input image from which a value must be sampled. In this case the determined positions also may not lie within the bounds of the input image and may not be integers. But the output image has no holes.

6 Exercise 6: 3D Global Transformation

Write the homogeneous 4x4 matrices for the following transforms:

Q.6.a Translate by +5 units in the X direction

Answer: For a global transform with translation in cartesian 3D we need homogeneous 4D. A general representation for 4x4 matrices involving rotation and translation is

$$\begin{pmatrix} R_{3\times3} & T_{3\times1} \\ 0_{1\times3} & 1_{1\times1} \end{pmatrix} \tag{3}$$

where R is 3×3 rotation matrix, and T is a 3×1 translation matrix. For a translation along the X axis by 3 units $T = (5, 0, 0)^T$, while R is the identity. Hence, we have

$$\begin{pmatrix} 1 & 0 & 0 & 5 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4}$$

Q.6.b Rotate by 30 degrees about the X axis

Answer: In the second case, where we are rotating about the X axis, the translation matrix is just 0. We need to remember the formula for rotation about an axis, which is (with angle θ),

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \sqrt{3}/2 & -1/2 & 0 \\ 0 & 1/2 & \sqrt{3}/2 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{5}$$

Q.6.c The rotation, followed by the translation above, followed by scaling by a factor of 2.

Answer: Finally, when we are combining these transformations, S*T*R, we apply the rotation first, followed by a translation. It is easy to verify by matrix multiplication, that this simply has the same form as equation 1 (but see the next problem for when we have R*T). The scale just multiplies everything by a factor of 2, giving

$$\begin{pmatrix} 2 & 0 & 0 & 10 \\ 0 & \sqrt{3} & -1 & 0 \\ 0 & 1 & \sqrt{3} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{6}$$

It is also possible to obtain this result by matrix multiplication of S*T*R

$$\begin{pmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 5 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \sqrt{3}/2 & -1/2 & 0 \\ 0 & 1/2 & \sqrt{3}/2 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 2 & 0 & 0 & 10 \\ 0 & \sqrt{3} & -1 & 0 \\ 0 & 1 & \sqrt{3} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{7}$$