Corners

(also known as interest points, keypoints, locally distinctive points)

Edges are very useful for marking boundaries between regions in an image. One limitation with edges, though, is that we cannot use them to reliably identify *single (isolated) points*, for example, points that we might try to match from one image to another. The problem is that edges have a well defined intensity gradient direction and so the intensity is by definition constant *along* the edge. As such, neighboring points along an edge are difficult to distinguish from each other.

It is common to refer to points that are locally distinctive as *corners*, though we don't necessarily need them to be the corner of anything. The word "corner" just suggests that there are two edges coming together at a sharp angle. A more general notion is the idea of a *locally distinctive point*, which means that the intensities in a small neighborhood (say 7×7) of the point (x_0, y_0) are different from those in the same size neighborhood of a nearby pixel $(x_0 + \Delta x, y_0 + \Delta y)$. Note that edges are typically *not* locally distinctive, since the intensit pattern in a local neighborhood tends to be constant as you move along the edge. Points within a constant intensity region are also not locally distinctive.

We set up the problem of finding locally distinctive points ("corners") as follows. The intensities at a point (x_0, y_0) are locally distinctive if any local shift in the image patch gives you a different pattern of image intensities. In particular, we look at the sum of squared differences of intensities of the patch and the shifted patch where the shift of the patch is $(\Delta x, \Delta y)$:

$$E(\Delta x, \Delta y) \equiv \sum_{(x,y) \in Ngd(x_0,y_0)} (I(x,y) - I(x + \Delta x, y + \Delta y))^2.$$

Here the summation is over points (x, y) in the neighborhood of (x_0, y_0). You could just check the above condition directly, by computing the sum for the eight neighbors of pixel (x_0, y_0). However, a more common way to express it as follows, which avoids having to restrict ourselves to integer values of Δx and Δy.

We approximate $I(x + \Delta x, y + \Delta y)$ with a Taylor series to first order:

$$I(x + \Delta x, y + \Delta y) \approx I(x, y) + \frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta y.$$

This is fine as long as $(\Delta x, \Delta y)$ are small, say ± 1 pixel and $I(x, y)$ has been smoothed. Typically one smooths by convolving with a 2D Gaussian $G(x, y)$ before doing these operations, but we will not write this explicitly to simplify the notation.

The image gradient[1] is

$$\nabla I = (\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y})$$

[1] think of this as $\nabla(G * I)$ since we are not explicitly writing the blurring with G

and so we have

$$\begin{aligned}\sum_{(x,y)\in Ngd(x_0,y_0)} (I(x,y) - I(x+\Delta x, y+\Delta y))^2 &\approx \sum_{(x,y)\in Ngd(x_0,y_0)} (\frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta y)^2 \\ &= \sum_{(x,y)\in Ngd(x_0,y_0)} ((\nabla I)\cdot(\Delta x, \Delta y))^2 \\ &= \sum_{(x,y)\in Ngd(x_0,y_0)} (\Delta x, \Delta y)(\nabla I)^T(\nabla I)(\Delta x, \Delta y)^T \\ &= (\Delta x, \Delta y)\{ \sum_{(x,y)\in Ngd(x_0,y_0)} (\nabla I)^T(\nabla I) \ \} \ (\Delta x, \Delta y)^T.\end{aligned}$$

Writing ∇I as a 2×1 column vector, the 2×2 matrix

$$\sum_{(x,y)\in Ngd(x_0,y_0)} (\nabla I)(\nabla I)^T = \left[\begin{array}{cc} \sum(\frac{\partial I}{\partial x})^2 & \sum(\frac{\partial I}{\partial x})(\frac{\partial I}{\partial y}) \\ \sum(\frac{\partial I}{\partial x})(\frac{\partial I}{\partial y}) & \sum(\frac{\partial I}{\partial y})^2 \end{array}\right]$$

is called the *second moment matrix*. Note that each of the four entries in the matrix is a sum of products of gradients over the neighborhood of (x_0, y_0).

[ASIDE: There is a closely related matrix to $\mathbf{M}$, called the *structure tensor*, which is entirely local, i.e., it is defined at a point (x, y) and is given by

$$(\nabla I)\cdot(\nabla I)^T = \left[\begin{array}{cc} (\frac{\partial I}{\partial x})^2 & (\frac{\partial I}{\partial x})(\frac{\partial I}{\partial y}) \\ (\frac{\partial I}{\partial x})(\frac{\partial I}{\partial y}) & (\frac{\partial I}{\partial y})^2 \end{array}\right]$$

If one sums up the structure tensors in a neighborhood of a point, one gets the second moment matrix. Sometimes in the literature the entries of the structure tensor are summed over windows in the image, in which case it is the same thing as the second moment matrix.]

For the second moment matrix, one commonly uses a weighting function $W(\cdot,\cdot))$ to give more weight to the center of the neighborhood. We will do so as well, so when we talk about the second moment matrix, we refer to the following matrix $\mathbf{M}$:

$$\mathbf{M} = \sum_{(x,y)\in Ngd(x_0,y_0)} W(x-x_0, y-y_0)\ (\nabla I)(\nabla I)^T = \left[\begin{array}{cc} \sum_{(x,y)} W()(\frac{\partial I}{\partial x})^2 & \sum_{(x,y)} W()(\frac{\partial I}{\partial x})(\frac{\partial I}{\partial y}) \\ \sum_{(x,y)} W()(\frac{\partial I}{\partial x})(\frac{\partial I}{\partial y}) & \sum_{(x,y)} W()(\frac{\partial I}{\partial y})^2 \end{array}\right]$$

This weighting function is typically a Gaussian.

Think of the weighting function as smoothing the entries of the $\mathbf{M}$ matrix, namely convolving these entries with a Gaussian weighting function $W()$. I wrote "convolution" rather than cross-correlation even though the formula above is for cross-correlation. But the Gaussian is symmetric, so convolution is the same as cross-correlation.)

Do not confuse this smoothing with what is done on the original image prior to taking the intensity gradient. The Gaussian that is used to smooth the image before taking the gradient typically has a small standard deviation, say about $\sigma = 1$ pixel. This σ is often called the *inner* *scale.* The Gaussian that is used to weight (smooth) the elements of the second moment matrix $\mathbf{M}$ has a larger standard deviation, typically at least say $\sigma = 2$ pixels. This larger standard deviation is called the *outer scale.*

The matrix $\mathbf{M}$ provides information about the geometry of the gradient field ∇I in the local neighborhood of (x_0, y_0). $\mathbf{M}$ is symmetric and positive definite and so it has non-negative eigenvalues. It can be written

$$\mathbf{M} = \mathbf{V} \mathbf{\Lambda} \mathbf{V}^T$$

where Λ is a diagonal matrix with elements λ_1, λ_2 and the columns $\mathbf{v_1}$ and $\mathbf{v_2}$ of $\mathbf{V}$ are the eigenvectors of $\mathbf{M}$, and they are orthonormal.

If one orders the eigenvalues λ_1, λ_2 such that $\lambda_1 \geq \lambda_2 \geq 0$ then one has the following qualitative relationship between the image intensity pattern in the neighbhorhood and the eigenvalues and eigenvectors:

- If the image intensity is near constant in the neighborhood, then the gradients will be small and $\lambda_1 \approx \lambda_2 \approx 0$.

- If there is a step edge in intensity in the neighborhood, then the gradients will all be parallel and so $\lambda_1 > 0$ but $\lambda_2 \approx 0$. In this case, $\mathbf{v}_1$ will be in the direction of the image gradient and $\mathbf{v}_2$ is perpendicular to the gradient.

- If there is a corner in the neighborhood (two edges meeting at a point) – or more generally, if there are multiple gradient directions in the neighborhood – then both λ_1 and λ_2 will be greater than 0.

  In this case, we say that the point is *locally distinctive.* The intuition comes from the original formulation back on page 1: shifting the local neighborhood in *any direction* will give a local patch with different point-by-point intensities.

Harris corners

The ideas above have motivated a heuristic method for corner detection by Harris and Stevens.

The above analysis suggests that we can decide if a point is locally distinctive by examining the eigenvalues of $\mathbf{M}$. However, to obtain the eigenvalues requires solving a quadratic equation, $a\lambda^2 + b\lambda + c = 0$, namely the characteristic equation $det(\mathbf{M} - \lambda \mathrm{I}) = 0$. Solving a quadratic requires computing a square root. We will check every pixel (x_0, y_0) in an image to see if it is locally distinctive, and so it seems we need to compute a square root at each pixel, which can be slow.[2]

One trick to avoid computing λ_1, λ_2 explicitly was proposed by Harris and Stevens (1988). It takes advantage of a basic fact from matrix algebra that the determinant of a matrix with eigenvalues λ_1 and λ_2 is the product $\lambda_1 \lambda_2$ and the trace is $\lambda_1 + \lambda_2$. The trick is to convert the conditions "both eigenvalues are large" into a condition on the determinant and trace of $\mathbf{M}$, both of which can easily be computed from $\mathbf{M}$ i.e. no need to compute the λ's explicitly. That is,

$$det(\mathbf{M}) = M_{11} M_{22} - M_{12} M_{21}, \quad tr(\mathbf{M}) = M_{11} + M_{22}.$$

[2]This was more of an issue a few decades ago.

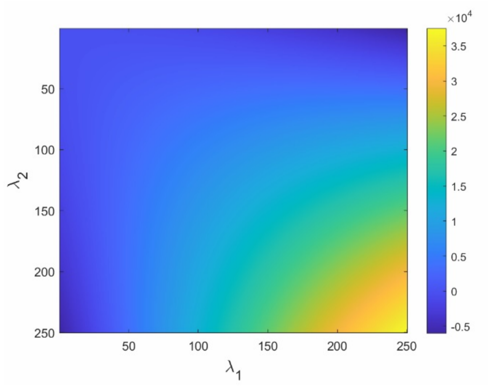
Harris and Stevens operator is is defined (for $k \approx 0.1$) to be

$$det(\mathbf{M}) - k(tr(\mathbf{M}))^2 = \lambda_1 \lambda_2 - k(\lambda_1 + \lambda_2)^2$$

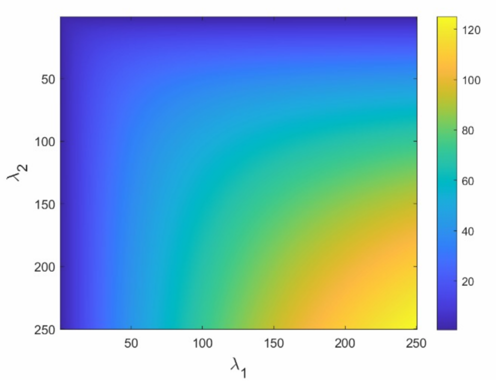
The operator response is negative if one of the eigenvalues is 0, and it is small if both of the eigenvalues are small (see below left). Otherwise this quantity is large. Now known as the *Harris corner* detector or *Harris interest point* detector, it detects points in the image where the above expression takes a value above some given threshold. A similar operator by (Brown, Szeliski, Winder 2004) and shown below right uses

$$\frac{det(\mathbf{M})}{tr(\mathbf{M}) + \epsilon} = \frac{\lambda_1 \lambda_2}{\lambda_1 + \lambda_2 + \epsilon}$$

where ϵ is a small positive number that avoids division by 0. In either case, a threshold on the operator response can be used to detect image points where both eigenvalues are large.



Harris and Stevens



Brown, Szeliski, Winder

Notice that if the determinant is near 0 but the trace is much different from zero, then one of the eigenvalues must be large and the other must be near zero. In this case, there must be strong image intensity gradients in the neighborhood of the point, and the gradients would be in only one direction. Thus, the trace and determinant of the second moment matrix could be used to detect edges as well as corners.

Notice that we will need to do non-maximum suppression to avoid having a clump of points in a neighborhood that are all greater than threshold and hence would all be interest points.

See slides for examples.