CSE 252B: Computer Vision II

Lecturer: Serge Belongie
Scribes: Louka Dlagnekov, Brian Fulkerson

LECTURE 13
Interest Point Detection

13.1. Introduction

The algorithms we have studied thus far have required us to provide two sets of interest points and, by virtue of the indexing of the two sets of coordinates, correspondences. So far we have done this by manually clicking on points in the images, but, in general, we will need to automate this process, and that will be the subject of this and the following lecture.

Figure 1 shows the left and right view from a wide baseline stereo pair and contains two examples of correspondences. The circle (intersection on the checkerboard) indicates a somewhat straightforward correspondence, and the square (corner of the mouth) is slightly more difficult. The correspondence problem increases in difficulty with wider baselines. For the human visual system, finding these correspondences is a trivial task.

In this lecture we will discuss how to find these interest points in an image.

[1]Department of Computer Science and Engineering, University of California, San Diego.

May 17, 2004

Figure 1. An example of two correspondences. (From MaSKS)

13.2. Interest Point Detection

We would like interest points to be:

- Sparse
- Informative
- Reproducible

13.2.1. Terminology

In the computer vision community, interest point detection is often called *corner detection*, even though not all features need be corners. What we usually mean by corners are actually L junctions, but there are also Y junctions, X junctions, Ψ junctions, etc. each having different levels if informativeness.

13.3. An Example of Corner Detection

It is desirable for a corner detector to fire when there exists a real corner. The area circled in green in Figure 2 is an example of a real corner in the physical world, and the area circled in red is an example of a window in the image that is not a real corner. Something like the latter is not a good interest point because leads to a false correspondence. This type of junction is called an *occlusion junction*, since it is formed by one surface occluding another surface at a different depth.

Figure 2. Example of good corner (green) and a not so good corner (red)

13.4. Properties of Corners

If we think of each point on the image plane as having a certain brightness, then we can imagine the image to be a 3D surface with light points being high and dark points being low. This enables us to use basic calculus on the surface and the simplest thing we can do is compute its gradient.

Definition 13.1. The gradient of an image I at a point (x, y) is denoted ∇I and points in the direction of greatest change from dark to light.

$$\nabla I = \begin{bmatrix} \frac{\partial I}{\partial x} \\ \frac{\partial I}{\partial y} \end{bmatrix} = \begin{bmatrix} I_x \\ I_y \end{bmatrix}$$

Corner features occur where there is a sharp change in the angle of the gradient. Figure 3 shows what a corner might look like in a window of 10 by 16 pixels. The arrows point in the direction of the gradient.

13.5. Finding Corners

We would like to know whether each part of the image is a corner and if so whether that corner is a good corner. To do so we can analyze the pixels of the image in a small neighborhood, $\mathcal{N}$, and assign the location of a corner to the point with minimum *weighted* distance to all the tangent lines through each pixel in $\mathcal{N}$ (see Figure 3b). A typical size for $\mathcal{N}$ is 5×5 pixels, though in practice it should be set as a percentage of the image dimensions, and its width should be odd, for reasons of symmetry.

13.5.1. Formulating the Cost Function

Consider a neighborhood of an image containing an ideal corner. For the image in Figure 4(a), the tangent lines intersect exactly at the location of the corner.

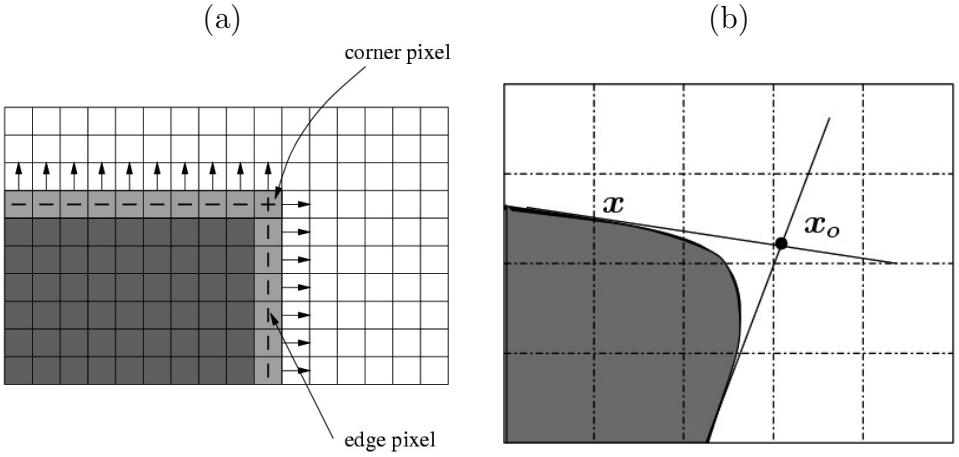

Figure 3. (a) A closeup (of neighborhood $\mathcal{N}$) of a corner in an image (b) Another closeup with tangent lines to the curve (perpendicular to the gradient) shown.

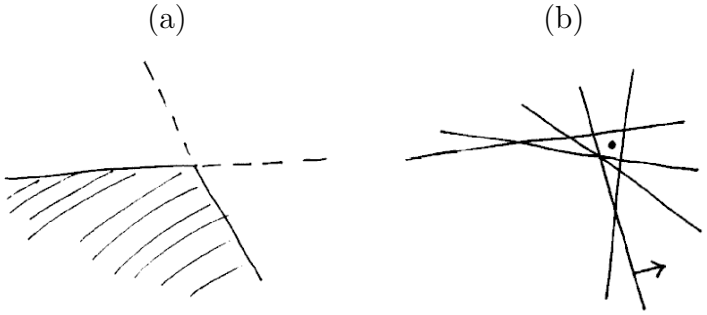

Figure 4. (a) Ideal case: tangent lines intersect exactly at corner. (b) In real images, we can only hope to find an approximation to this point of intersection, indicated here by the dot. The arrow indicates a gradient vector at a sample pixel.

In real images, the corners aren't this clean and we need an approximate solution. We will seek a least-squares solution for the point of intersection of the tangent lines, as suggested in Figure 4(b).

The equation for the tangent line $\boldsymbol{l}_{\boldsymbol{x}'}$ passing through a pixel at location $\boldsymbol{x}'$ is

$$D_{\boldsymbol{x}'}(\boldsymbol{x}) = \nabla I(\boldsymbol{x}')^\top (\boldsymbol{x} - \boldsymbol{x}') = 0$$

where $\nabla I(\boldsymbol{x}')$ denotes the gradient of the image I at $\boldsymbol{x}'$, indicated by the small arrow in Figure 4(b). Our goal is to find the point $\boldsymbol{x}_o$ with the minimum

perpendicular distance to all the lines in this neighborhood:

$$\boldsymbol{x}_o = \arg\min_{\boldsymbol{x}\in\mathbb{R}^2} \int_{\boldsymbol{x}'\in\mathcal{N}} D_{\boldsymbol{x}'}(\boldsymbol{x})^2 d\boldsymbol{x}'$$

where $\mathcal{N}$ denotes the neighborhood around a candidate corner location. This expression is the weighted integral of the squares of the distances from $\boldsymbol{x}$ to all lines in $\mathcal{N}$. $D_{\boldsymbol{x}'}(\boldsymbol{x})$ describes the distance from $\boldsymbol{x}$ to the line $\boldsymbol{l}_{\boldsymbol{x}'}$ multiplied by the gradient magnitude. In this way we give more weight to lines that pass through strong edge pixels. (Recall that pixels in roughly constant regions have small gradient magnitudes with arbitrary gradient directions.)

13.5.2. The least-squares solution

We can find $\boldsymbol{x}_o$ in closed form as follows:

$$\begin{aligned}
\boldsymbol{x}_o &= \arg\min_{\boldsymbol{x}\in\mathbb{R}^2} \int \left(\nabla I(\boldsymbol{x}')^\top (\boldsymbol{x}-\boldsymbol{x}')\right)^2 d\boldsymbol{x}' \\
&= \arg\min_{\boldsymbol{x}} \int (\boldsymbol{x}-\boldsymbol{x}')^\top \nabla I(\boldsymbol{x}') \nabla I(\boldsymbol{x}')^\top (\boldsymbol{x}-\boldsymbol{x}') d\boldsymbol{x}' \\
&= \arg\min_{\boldsymbol{x}} \boldsymbol{x}^\top A \boldsymbol{x} - 2\boldsymbol{x}^\top \boldsymbol{b} + c \qquad (13.2)
\end{aligned}$$

where

$$\begin{aligned}
A &= \int \nabla I(\boldsymbol{x}') \nabla I(\boldsymbol{x}')^\top d\boldsymbol{x}' \\
\boldsymbol{b} &= \int \nabla I(\boldsymbol{x}') \nabla I(\boldsymbol{x}')^\top \boldsymbol{x}' d\boldsymbol{x}' \\
c &= \int \boldsymbol{x}'^\top \nabla I(\boldsymbol{x}') \nabla I(\boldsymbol{x}')^\top \boldsymbol{x}' d\boldsymbol{x}'
\end{aligned}$$

Note: A is 2×2, $\boldsymbol{b}$ is 2×1, and c is a scalar.

The arg min in Equation (13.2) can be found by taking its derivative w.r.t. $\boldsymbol{x}$ and setting it equal to zero. This gives

$$2A\boldsymbol{x} - 2\boldsymbol{b} = \boldsymbol{0} \quad \Longrightarrow \quad A\boldsymbol{x} = \boldsymbol{b}$$

at the minimum. Therefore the subpixel coordinates of the corner are given by $\boldsymbol{x}_o = A^{-1}\boldsymbol{b}$.

The above method is known as Förstner corner detection (1987). A variant of it is called the Harris corner detector (1988).

The solution $\boldsymbol{x}_o = A^{-1}\boldsymbol{b}$ will be possible only when the rank of A is 2.

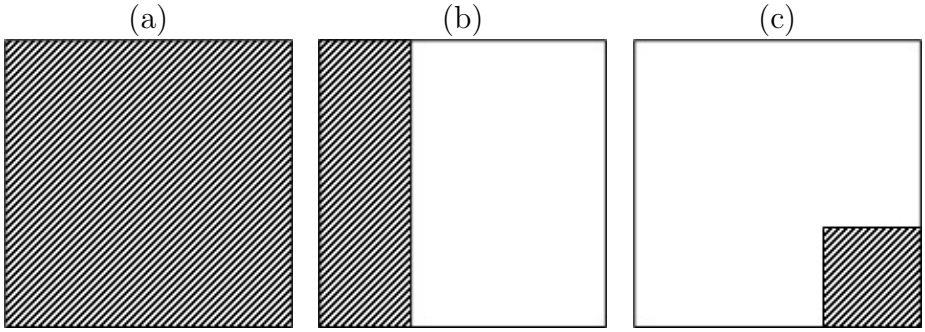

Figure 5. Some example image neighborhoods: (a) constant, (b) edge, (c) corner

Since real images are discrete, A's contents are obtained using sums in place of integrals,

$$A = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix} \tag{13.3}$$

where each sum is over the neighborhood $\mathcal{N}$. (And similarly for $\boldsymbol{b}$ and c.) Matlab's discrete approximation to the gradient is implemented in `gradient.m`.

13.5.3. Discussion

- Different neighborhoods around a true corner location will each yield different estimates of $\boldsymbol{x}_o$. Keep the one from the neighborhood whose center is closest to $\boldsymbol{x}_o$.
- The symmetric positive semidefinite matrix A is known as the "windowed image second moment matrix" or just "second moment matrix." The same matrix appears in Lucas and Kanade's method of recovering optical flow.
- Note the similarity between A and the covariance matrix for a collection of n gradient vectors.
- By substituting normal lines for tangents, the above least-squares formulation can be modified in a straightforward way to find the center of circular features instead of corners.
- A spatial weighting function $w(\boldsymbol{x}')$ (e.g. a Gaussian) is sometimes used in the above integrals to give more weight to pixels near the center of the neighborhood.

13.5.4. Possible Forms of the A Matrix

Three example image neighborhoods are shown in Figure 5. These scenarios will result in the following forms of the A matrix:

$$(a): \quad A = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

$$(b): \quad A \propto \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$$

$$(c): \quad A \propto \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

One can notice that if half of the gradient vectors point up and the other half point to the side, then A will take the form of (c) and have rank 2 – a "healthy feature" (a Serge term). Also note that a small disk or square in the neighborhood as the sole feature could also give a second moment matrix in the form of (c).

In practice we look at the two eigenvalues of A (call them $\lambda_1\,\lambda_2$, with $\lambda_1 \leq \lambda_2$) to get a soft measure of the rank for a neighborhood. E.g., you could threshold $\lambda_1/\text{trace}(A) = \lambda_1/(\lambda_1 + \lambda_2)$. Sometimes people just use λ_1. Harris suggests thresholding $\det(A) + k \cdot \text{trace}^2(A)$ with $k \approx 0.04$ to obtain features that are somewhere in between corners and edges.

13.6. Conclusion

We have described a framework for extracting points to feed to the algorithms we studied earlier in the class. We have not yet described how to find correspondences, i.e., how to match interests points between two images. That will be the subject of next lecture.