# Handy Equation Reference for CSE 166 (Image Processing)

Serge Belongie

Dec. 3, 2002

## 1 binomial filter

(approximation to Gaussian)

Relation between $\sigma^2$ (variance) and $N$ (number of taps):

$$N = 4\sigma^2 + 1$$

1D case:

$$h(x) = 2^{-(N-1)} \begin{pmatrix} N-1 \\ x \end{pmatrix}, \quad x = 0, 1, \ldots, N-1$$

For example, the 5-tap binomial filter is [1,4,6,4,1], with a normalization factor of 16.

2D case (it's separable, like Gaussian):

$$h(x, y) = 4^{-(N-1)} \begin{pmatrix} N-1 \\ x \end{pmatrix} \begin{pmatrix} N-1 \\ y \end{pmatrix}, \quad x, y = 0, 1, \ldots, N-1$$

## 2 histogram comparison: $\chi^2$ test

Chi-squared distance between two normalized histograms $h_i(k)$ and $h_j(k)$:

$$\chi^2(i, j) = \frac{1}{2} \sum_{k=1}^{K} \frac{[h_i(k) - h_j(k)]^2}{h_i(k) + h_j(k)}$$

The value of $\chi^2(i, j)$ must lie between 0 and 1. This can be proven using the fact that $\sum_k h(k) = 1$. To keep the expression from blowing up, we assume that $h_i(k) > 0$ and $h_j(k) > 0$ for all $k$.

## 3 $2\times2$ symmetric matrix: eigenvalues and eigenvectors

Consider the symmetric $2 \times 2$ matrix

$$A = \begin{bmatrix} a & b \\ b & c \end{bmatrix}.$$

The eigenvalues of A are given by

$$\lambda = \frac{\mathrm{tr}(A) \pm \sqrt{\mathrm{tr}(A)^2 - 4\det(A)}}{2}$$

and the angle of the principal eigenvector of $A$ is given by

$$\phi = \frac{1}{2}\tan^{-1}\left(\frac{2b}{a-c}\right)$$

The angle of the other eigenvector is $\phi + \pi/2$.

## 4 $2 \times 2$ matrix: inverse

Consider the $2 \times 2$ matrix

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}.$$

The inverse is given by

$$A^{-1} = \frac{1}{\det(A)} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}.$$

## 5 scatter matrix

Given $K$ points on a 2D shape (e.g. on the boundary) with coordinates

$$\mathbf{x}_k = \begin{bmatrix} x_k \\ y_k \end{bmatrix}$$

the centroid (or center of mass) is the following $2 \times 1$ vector

$$\mathbf{m} = \frac{1}{K}\sum_{k=1}^{K} \mathbf{x}_k$$

and the scatter matrix (or covariance matrix) is the following $2 \times 2$ symmetric matrix

$$C = \frac{1}{K}\sum_{k=1}^{K}(\mathbf{x}_k - \mathbf{m})(\mathbf{x}_k - \mathbf{m})^T = \frac{1}{K}\sum_{k=1}^{K}\mathbf{x}_k\mathbf{x}_k^T - \mathbf{m}\mathbf{m}^T$$

Now for simplicity assume that we have subtracted the centroid from each coordinate in the shape (this is called "centering"). In this case the entries of $C$ are given by

$$C = \frac{1}{K} \begin{bmatrix} \sum_k x_k^2 & \sum_k x_k y_k \\ \sum_k y_k x_k & \sum_k y_k^2 \end{bmatrix}$$

We can diagonalize the scatter matrix as follows:

$$C = U \Lambda U^T = \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \begin{bmatrix} \cos\phi & \sin\phi \\ -\sin\phi & \cos\phi \end{bmatrix}$$

Here, $U$ is the matrix of eigenvectors (it is orthogonal, so $UU^T = U^T U = I$) and $\Lambda$ is the diagonal matrix of eigenvalues. $C$ is positive semidefinite, so $\lambda_1, \lambda_2 \geq 0$, and we take the convention that $\lambda_1 \geq \lambda_2$, without loss of generality.

The columns of $U$ (i.e. the eigenvectors) give the *principal axes*, which are unit vectors with angles of $\phi$ and $\phi + \pi/2$ with respect to the original coordinate axes.

The angle $\phi$ is also known as the "axis of least inertia."

The "aspect ratio" for the shape, which measures the elongation of the shape along its (first) principal axis, is given by $\sqrt{\lambda_1/\lambda_2}$.

# 6 affine transform

We are given two sets of $K$ points in corresponding order on a pair of 2D shapes

$$\mathbf{x}_k = \begin{bmatrix} x_k \\ y_k \end{bmatrix}, \quad \mathbf{x}'_k = \begin{bmatrix} x'_k \\ y'_k \end{bmatrix}$$

Call the first point set the "model" and the second (primed) point set the "target."

The affine transformation is a linear transformation on the 2D coordinates involving 6 degrees of freedom:

$$\mathbf{x}'_k = A\mathbf{x}_k + \mathbf{t}, \qquad A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}, \quad \mathbf{t} = \begin{bmatrix} t_1 \\ t_2 \end{bmatrix}$$

The $A$ matrix accounts for rotation, shear, and scaling, while $\mathbf{t}$ takes care of translation.

In general, an affine transform maps a square into a parallelogram. In particular, if two lines are parallel before an affine transform, they will remain parallel after the affine transform.

To get the least squares solution, first center both point sets. (The optimal $\mathbf{t}$ is given by the difference vector between the centroids of the two pointsets.) Now express the affine transform in matrix form $Y = AX$ or

$$\begin{bmatrix} x'_1 & x'_2 & \cdots & x'_K \\ y'_1 & y'_2 & \cdots & y'_K \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x_1 & x_2 & \cdots & x_K \\ y_1 & y_2 & \cdots & y_K \end{bmatrix}$$

Then to find the $A$ that minimizes $\sum_k \|\mathbf{x}'_k - A\mathbf{x}_k\|^2$ we compute

$$A = YX^+ = YX^T(XX^T)^{-1}$$

where $X^+$ is the *pseudoinverse* of $X$.

# 7 windowed image second moment matrix

The windowed image second moment matrix is used for corner detection. To compute it, first compute the gradient (after smoothing the image, if necessary). The components of the gradient $\nabla I$ are

$$\nabla I = \begin{bmatrix} I_x \\ I_y \end{bmatrix}$$

and it is a vector field in which each vector points in the direction of greatest change, from dark to light.

The windowed image second moment matrix corresponding to a generic pixel $i$ in the image is given by

$$C = \sum_{j \in \mathcal{N}(i)} \nabla I(j) \nabla I(j)^T$$

where $\mathcal{N}(i)$ denotes a small neighborhood around pixel $i$ and $\nabla I(j)$ denotes the value of the gradient at pixel $j$, i.e. $\nabla I(j) = \nabla I(x_j, y_j)$. As an example, we might define $\mathcal{N}(i)$ to be a $5 \times 5$ window centered on pixel $i$, with the pixel index $i$ ranging from 1 through 25 going from top to bottom, left to right.

Note that the mean is not subtracted here, as it was for the scatter matrix. (Some authors do subtract the mean.)

If one wants to give more weight to the pixels near the center, one solution is to define $C$ as

$$C = \sum_{j \in \mathcal{N}(i)} w(j) \nabla I(j) \nabla I(j)^T$$

where $w_j = w(x_j, y_j)$ is a Gaussian evaluated over $\mathcal{N}(i)$.

We can diagonalize $C$ as before:

$$C = U \Lambda U^T$$

When $\mathcal{N}(i)$ is centered on a corner-like image neighborhood, we get $\lambda_1 \approx \lambda_2 \gg 0$. In that case, $\phi$ is not meaningful.

When $\mathcal{N}(i)$ is centered on an edge-like image neighborhood, we get $\lambda_1 \gg \lambda_2$. In that case, $\phi$ is perpendicular to the dominant edge orientation.

Finally, when $\mathcal{N}(i)$ is centered on a neighborhood of constant gray value, we have $\lambda_1, \lambda_2 \approx 0$.

# 8 covariance matrix trick for PCA

Suppose we are given an $n \times K$ matrix of centered feature vectors

$$X = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_K]$$

and we would like to find the eigenvectors of $C = XX^T$. Assuming that $n \gg K$, where $n$ is dimensionality of the feature vector (e.g. the number of pixels) and $K$ is the number of examples, the matrix $C$ will be huge ($N \times N$).

The following trick allows us to get the leading $K$ eigenvectors of $C$ by diagonalizing a smaller ($K \times K$) matrix. Our goal is to solve

$$C\mathbf{e} = \lambda\mathbf{e} \qquad \text{or} \qquad XX^T\mathbf{e} = \lambda\mathbf{e}$$

Consider the following problem instead:

$$X^TX\mathbf{f} = \lambda\mathbf{f}$$

wherein the matrix $X^TX$ is only $K \times K$. Now multiply both sides from the left by $X$:

$$XX^TX\mathbf{f} = \lambda X\mathbf{f}$$

Notice that if we group $X\mathbf{f}$ as follows,

$$XX^T(X\mathbf{f}) = \lambda(X\mathbf{f})$$

this matches our desired eigenvalue problem, with $\mathbf{e} = X\mathbf{f}$. Note: when using this method, $\mathbf{e}$ must be re-normalized (to have norm 1), which is accomplished by reassigning it as $\mathbf{e} \leftarrow \mathbf{e}/\sqrt{\lambda}$.

# 9 distance matrix computation

We are given $N$ feature vectors $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$ and $k$ prototype vectors $\{\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_k\}$. Assume all of the vectors are of size $n \times 1$. The $N \times k$ matrix of squared Euclidean distances between the feature vectors and the prototype vectors can be obtained by computing the following matrix:

$$T = \mathbf{E}\mathbf{1}_k^T + \mathbf{1}_N\mathbf{F}^T - 2X^TY$$

where $\mathbf{E}$ is an $N \times 1$ vector with entries $E_i = \|\mathbf{x}_i\|^2$, $\mathbf{F}$ is a $k \times 1$ vector with entries $F_i = \|\mathbf{m}_i\|^2$, $\mathbf{1}$ is a column vector of ones whose length is given by its subscript, $X$ is an $n \times N$ matrix of feature vectors, and $Y$ is an $n \times k$ matrix of prototype vectors. In other words, the entries of $T$ are given by

$$T_{ij} = \|\mathbf{x}_i - \mathbf{m}_j\|^2 = \|\mathbf{x}_i\|^2 + \|\mathbf{m}_j\|^2 - 2\mathbf{x}_i^T\mathbf{m}_j$$