CSC320 Tutorial Notes

Marcus Brubaker (t4brubak@cdf.toronto.edu)

February 17th, 2006

1 Sobel Mask for Computing Gradients

Recall the 2D Taylor series expansion of $I(x, y)$ about $I(0, 0)$

$$I(x, y) = I(0, 0) + x\frac{\partial I}{\partial x}(0, 0) + y\frac{\partial I}{\partial y}(0, 0) + \dots$$

Like we did in 1D we can use this to estimate the gradient at a point using local image data. If we look at the immediate image pixels we get the following equations

$$\begin{aligned}
I(-1,-1) &= I(0,0) - \frac{\partial I}{\partial x}(0,0) - \frac{\partial I}{\partial y}(0,0) \\
I(-1,0) &= I(0,0) - \frac{\partial I}{\partial x}(0,0) + 0 \\
I(-1,1) &= I(0,0) - \frac{\partial I}{\partial x}(0,0) + \frac{\partial I}{\partial y}(0,0) \\
&\vdots \\
I(1,-1) &= I(0,0) + \frac{\partial I}{\partial x}(0,0) - \frac{\partial I}{\partial y}(0,0) \\
I(1,0) &= I(0,0) + \frac{\partial I}{\partial x}(0,0) + 0 \\
I(1,1) &= I(0,0) + \frac{\partial I}{\partial x}(0,0) + \frac{\partial I}{\partial y}(0,0)
\end{aligned}$$

which can be expressed in matrix forms as

$$Xd = I$$

where

$$X = \begin{bmatrix} 1 & -1 & -1 \\ 1 & -1 & 0 \\ 1 & -1 & 1 \\ & \vdots & \\ 1 & 1 & -1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

$$d = \begin{bmatrix} I(0,0) \\ \frac{\partial I}{\partial x}(0,0) \\ \frac{\partial I}{\partial y}(0,0) \end{bmatrix}$$

and I contains the image data.

We can use the same trick as in 1D and use the psuedo-inverse of X to compute only the terms we're interested in efficiently. However, as in 1D we may also want to smooth out the derivates by weighting things. The Sobel mask corresponds to using weights proportional to the following:

$$\begin{aligned} \Omega(i,j) &= \sqrt{2\pi}e^{-\frac{i^2+j^2}{2}} \\ &\approx \begin{cases} 1 & |i| = |j| = 1 \\ \sqrt{2} & |i| \neq |j| \\ 2.5 & |i| = |j| = 0 \end{cases} \end{aligned}$$

By setting up the system

$$WXd = WI$$

and computing the psuedo-inverse of WX we get the following

$$(WX)^*W = \begin{bmatrix} & & & & - & & & & \\ -\frac{1}{8} & -\frac{2}{8} & -\frac{1}{8} & 0 & 0 & 0 & \frac{1}{8} & \frac{2}{8} & \frac{1}{8} \\ -\frac{1}{8} & 0 & \frac{1}{8} & -\frac{2}{8} & 0 & \frac{2}{8} & -\frac{1}{8} & 0 & \frac{1}{8} \end{bmatrix}$$

which can be used to estimate derivatives using a dot product.

Computation could be made faster by scaling $(WX)^*$ in order to get integer elements which would then make our dot products only use integer arithmetic so we end up with

$$8 * (WX)^*W = \begin{bmatrix} & & & & - & & & & \\ -1 & -2 & -1 & 0 & 0 & 0 & 1 & 2 & 1 \\ -1 & 0 & 1 & -2 & 0 & 2 & -1 & 0 & 1 \end{bmatrix}$$

which is called the Sobel mask.

2 The (single-scale) Lowe Feature Detector

Three steps to the (single-scale) Lowe Feature Detector.

1. Identify extrema of the Laplacian of the image, $S(x, y)$.
2. Eliminate extrema where $|S(x, y)| < threshold$. These points have minimal local image structure and are likely noise.
3. Eliminate extrema which have cylindrical-like local structure. These are places in the image which have a strong edge structure. Unfortunately, these points are poorly localized along the direction of the edge and are generally not good feature points. We can remove these by keeping points where $\frac{\lambda_1}{\lambda_2} < r$ where λ_1 and λ_2 are the eigenvalues of the Hessian H and r is a threshold, usually around 10.

Step 3 might seem to require computing the eigenvalues of H but it turns out that it can done much more efficiently.

$$\frac{\lambda_1}{\lambda_2} \leq r \quad \Longleftrightarrow \quad \lambda_1 \leq r\lambda_2$$

$$\begin{aligned} \frac{Tr(H)^2}{Det(H)} &= \frac{(\lambda_1 + \lambda_2)^2}{\lambda_1 \lambda_2} \\ &\leq \frac{(r\lambda_2 + \lambda_2)^2}{r\lambda_2 \lambda_2} \\ &= \frac{(r+1)^2}{r} \end{aligned}$$

Thus we can replace the test $\frac{\lambda_1}{\lambda_2} \leq r$ with the equivalent test $\frac{Tr(H)^2}{Det(H)} \leq \frac{(r+1)^2}{r}$ which is much more efficient to compute.