Engineering Tripos Part IIB FOURTH YEAR

Module 4F12: Computer Vision and Robotics

Solutions to Examples Paper 1

1. *Introduction to geometric constraints*

   Techniques to recover shape form texture assume that the texture is both homogeneous and isotropic. Shape from shading techniques assume that the light source is Lambertian and the reflectance properties of surfaces in the scene are isotropic: it is also necessary to assume that the scene is top-lit. Structure from motion techniques assume that the scene being viewed is rigid (ie. no independently moving bodies). Both structure from motion and stereo vision assume that the camera is performing a perspective projection onto a CCD array. Shape from contour also assumes perspective projection onto a CCD array: the silhouette defines a set of tangent planes to the observed surface, from which the local surface orientation can be derived.

2. *Images*

   Each frame requires $512 \times 512 \times 1 = 2.62 \times 10^5$ Bytes. A 25Hz stereo image stream requires $2.62 \times 10^5 \times 25 \times 2 = 1.3 \times 10^7$ Bytes/s. Assuming an average A4 page of text contains 50 lines, with about 80 characters on each line, and that a character is represented (using an ASCII code) as a single byte, a page of text requires $80 \times 50 \times 1 = 4000$ Bytes. So, instead of one second of stereo video, we could alternatively store $1.3 \times 10^7 / 4000 \approx 3000$ pages of text — enough for a small encyclopaedia!

3. *Smoothing by convolution with a Gaussian*

   Consider smoothing an image, first with a Gaussian of standard deviation σ_1, then with a Gaussian of standard deviation σ_2:

   $$s(x) = g_{\sigma 2}(x) * (g_{\sigma 1}(x) * I(x))$$

   Since convolution is associative, we can write this as the convolution of the image with the kernel $g_{\sigma 2}(x) * g_{\sigma 1}(x)$:

   $$s(x) = (g_{\sigma 2}(x) * g_{\sigma 1}(x)) * I(x)$$

   The easiest way to evaluate the convolution of two Gaussians is to find their Fourier transforms and then multiply the transforms in the frequency domain. If $g_\sigma(x) \leftrightarrow G_\sigma(\omega)$, then:

   $$\begin{aligned} G_\sigma(\omega) &= \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{\infty} \exp\left(-\frac{x^2}{2\sigma^2}\right) e^{-j\omega x} dx \\ &= \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{\infty} \exp\left[-\left(\frac{x^2}{2\sigma^2} + j\omega x\right)\right] dx \end{aligned}$$

$$
\begin{aligned}
&= \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{\infty} \exp\left[-\frac{1}{2\sigma^2}\left(x^2 + 2j\omega\sigma^2 x\right)\right] dx \\
&= \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{\infty} \exp\left[-\frac{1}{2\sigma^2}\left((x + j\omega\sigma^2)^2 - j^2\omega^2\sigma^4\right)\right] dx \\
&= \exp\left(-\frac{\omega^2\sigma^2}{2}\right) \times \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{\infty} \exp\left(-\frac{(x + j\omega\sigma^2)^2}{2\sigma^2}\right) dx \\
&= \exp\left(-\frac{\omega^2\sigma^2}{2}\right) \quad \text{(since the integral is a standard Gaussian)}
\end{aligned}
$$

Hence

$$
\begin{aligned}
g_{\sigma 2}(x) * g_{\sigma 1}(x) &\leftrightarrow G_{\sigma 2}(\omega) \times G_{\sigma 1}(\omega) = \exp\left(-\frac{\omega^2\sigma_2^2}{2}\right) \times \exp\left(-\frac{\omega^2\sigma_1^2}{2}\right) \\
\Leftrightarrow g_{\sigma 2}(x) * g_{\sigma 1}(x) &\leftrightarrow \exp\left(-\frac{\omega^2(\sigma_2^2 + \sigma_1^2)}{2}\right)
\end{aligned}
$$

The expression on the right is the Fourier transforms of a Gaussian with standard deviation $\sqrt{\sigma_2^2 + \sigma_1^2}$. So the convolution of two Gaussians with variances σ_1^2 and σ_2^2 is a Gaussian with variance $\sigma_1^2 + \sigma_2^2$. It follows that consecutive smoothing with a series of 1D Gaussians, each with a particular standard deviation σ_i, is equivalent to a single convolution with a Gaussian of variance $\sum_i \sigma_i^2$.

Spatial domain convolution

Alternatively, we can convolve in the spatial domain. The trick, once again, is to complete the square:

$$
g_{\sigma 2}(x) * g_{\sigma 1}(x) = \frac{1}{2\pi\sigma_1\sigma_2} \int_{-\infty}^{\infty} \exp\left(-\frac{u^2}{2\sigma_2^2}\right) \exp\left(-\frac{(x-u)^2}{2\sigma_1^2}\right) du
$$

$$
\begin{aligned}
&= \frac{1}{2\pi\sigma_1\sigma_2} \int_{-\infty}^{\infty} \exp\left(\frac{-u^2\sigma_1^2 - x^2\sigma_2^2 - u^2\sigma_2^2 + 2ux\sigma_2^2}{2\sigma_1^2\sigma_2^2}\right) du \\
&= \frac{1}{2\pi\sigma_1\sigma_2} \int_{-\infty}^{\infty} \exp\left(\frac{-(\sigma_1^2 + \sigma_2^2)\left(u - \frac{x\sigma_2^2}{\sigma_1^2+\sigma_2^2}\right)^2 + \frac{x^2\sigma_2^4}{\sigma_1^2+\sigma_2^2} - x^2\sigma_2^2}{2\sigma_1^2\sigma_2^2}\right) du \\
&= \frac{1}{2\pi\sigma_1\sigma_2} \int_{-\infty}^{\infty} \exp\left(\frac{-\left(u - \frac{x\sigma_2^2}{\sigma_1^2+\sigma_2^2}\right)^2}{\frac{2\sigma_1^2\sigma_2^2}{\sigma_1^2+\sigma_2^2}}\right) \exp\left(\frac{-x^2\sigma_1^2\sigma_2^2}{2(\sigma_1^2 + \sigma_2^2)\sigma_1^2\sigma_2^2}\right) du \\
&= \frac{1}{2\pi\sigma_1\sigma_2} \exp\left(\frac{-x^2}{2(\sigma_1^2 + \sigma_2^2)}\right) \int_{-\infty}^{\infty} \exp\left(\frac{-\left(u - \frac{x\sigma_2^2}{\sigma_1^2+\sigma_2^2}\right)^2}{2\left(\frac{\sigma_1\sigma_2}{\sqrt{\sigma_1^2+\sigma_2^2}}\right)^2}\right) du
\end{aligned}
$$

$$= \frac{1}{\sqrt{2\pi}\sqrt{\sigma_1^2+\sigma_2^2}} \exp\left(\frac{-x^2}{2(\sigma_1^2+\sigma_2^2)}\right) \frac{1}{\sqrt{2\pi}\left(\frac{\sigma_1\sigma_2}{\sqrt{\sigma_1^2+\sigma_2^2}}\right)} \int_{-\infty}^{\infty} \exp\left(\frac{-\left(u-\frac{x\sigma_2^2}{\sigma_1^2+\sigma_2^2}\right)^2}{2\left(\frac{\sigma_1\sigma_2}{\sqrt{\sigma_1^2+\sigma_2^2}}\right)^2}\right) du$$

$$= \frac{1}{\sqrt{2\pi}\sqrt{\sigma_1^2+\sigma_2^2}} \exp\left(\frac{-x^2}{2(\sigma_1^2+\sigma_2^2)}\right) \quad \text{(since the integral is a standard Gaussian)}$$

This expression is a Gaussian with standard deviation $\sqrt{\sigma_2^2+\sigma_1^2}$.

4. *Generating the Gaussian filter kernel*

   In general, if we discard the sample $(n+1)$ pixels from the center of the kernel, the size of the kernel will be $2n+1$ pixels. We can find n by solving:

$$\begin{aligned} \exp\left[-\frac{(n+1)^2}{2\sigma^2}\right] &< \frac{1}{1000} \\ \Leftrightarrow n &> 3.7\sigma - 1 \end{aligned}$$

   So n must be the nearest integer to $3.7\sigma - 0.5$.

   (a) Applying this formula for $\sigma = 1$ gives $n = 3$ and a kernel size of $2n+1 = 7$ pixels. The filter coefficients can be found by sampling the 1D Gaussian $g_1(x)$ at the points $x = \{-3, -2, -1, 0, 1, 2, 3\}$. The sum of the coefficients is one, so regions of uniform intensity are unaffected by smoothing.

   (b) For $\sigma = 5$ we get $n = 18$ and a kernel size of 37 pixels.

   (c) The choice of σ depends on the *scale* at which the image is to be analysed. Modest smoothing (a Gaussian kernel with small σ) brings out edges at a fine scale. More smoothing (larger σ) identifies edges at larger scales, suppressing the finer detail. There is no right or wrong size for the kernel: it all depends on the scale we're interested in. Another factor is image noise: the smoothing suppresses noise. It may be difficult to detect fine scale edges, since a kernel large enough to suppress the noise may also suppress the fine detail. Finally, computation time may be an issue: large σ means a large kernel and computationally expensive convolutions.

5. *Discrete convolution*

   The image and filter kernels are discrete quantities and convolutions are performed as truncated summations:

$$s(x) = \sum_{u=-n}^{n} g_\sigma(u) I(x-u)$$

   Applying this to the pixel with intensity 118, which is the 11th pixel in the row, we obtain

$$s(x) = \sum_{u=-3}^{3} g_\sigma(u) I(11-u)$$

$$
\begin{aligned}
&= 0.004 \times 57 + 0.054 \times 77 + 0.242 \times 99 + 0.399 \times 118 \ldots \\
&\quad +0.242 \times 130 + 0.054 \times 133 + 0.004 \times 134 \\
&= 115 \qquad \text{(to the nearest integer)}
\end{aligned}
$$

6. *Derivative of convolution theorem*

   (a) This is easily proved by interchanging the order of differentiation and integration:

$$
\begin{aligned}
s'(x) &= \frac{d}{dx}\left[g_\sigma(x) * I(x)\right] = \frac{d}{dx}\left[\int_{-\infty}^{\infty} g(x-u)I(u)\,du\right] \\
&= \int_{-\infty}^{\infty} \frac{d}{dx}\left[g(x-u)\right] I(u)\,du \\
&= \int_{-\infty}^{\infty} g'(x-u)I(u)\,du = g'_\sigma(x) * I(x)
\end{aligned}
$$

   (b) Edges are localised at the maxima and minima of $\frac{d}{dx}[g_\sigma(x) * I(x)]$. These occur when

$$\frac{d^2}{dx^2}[g_\sigma(x) * I(x)] = 0$$

   The derivate of convolution theorem tells us that

$$\frac{d^2}{dx^2}[g_\sigma(x) * I(x)] = g''_\sigma(x) * I(x)$$

   Hence edges can be localised at the zero-crossings of $g''_\sigma(x) * I(x)$.

7. *Differentiation and 1D edge detection*

   An approximation to the first-order spatial derivative of $I(x)$ mid-way between the nth and $(n+1)$th sample is $I(n+1) - I(n)$. This can be computed by convolving with the kernel | 1 | -1 | (remember that the kernel is flipped before the multiply and accumulate operation).

   Applying this kernel to the smoothed row of pixels gives

2	3	3	8	15	19	17	11	6	1	0	- 1

   The intensity discontinuity is at the maximum of the first-order spatial derivative. The maximum derivative (19) occurs between the pixel with intensity 79 and the pixel with intensity 98[1].

8. *Decomposition of 2D convolution*

[1] If you want to be more precise, you can localise the discontinuity to sub-pixel accuracy by calculating the second order derivatives and then interpolating to find the zero-crossing. You'll find that the intensity discontinuity is two thirds of the way between the pixel with intensity 79 and the pixel with intensity 98.

The 2D convolution can be decomposed into two 1D convolutions as follows:

$$
\begin{aligned}
G_\sigma(x,y) * I(x,y) &= \frac{1}{2\pi\sigma^2} \int\int I(x-u, y-v)\ \exp - \left(\frac{u^2+v^2}{2\sigma^2}\right)\ du\ dv \\
&= \frac{1}{\sqrt{2\pi}\sigma} \int \exp - \left(\frac{u^2}{2\sigma^2}\right) \left[\frac{1}{\sqrt{2\pi}\sigma} \int I(x-u, y-v) \exp - \left(\frac{v^2}{2\sigma^2}\right) dv\right] du \\
&= \frac{1}{\sqrt{2\pi}\sigma} \int \exp - \left(\frac{u^2}{2\sigma^2}\right) [g_\sigma(y) * I(x-u, y)]\, du \\
&= g_\sigma(x) * [g_\sigma(y) * I(x,y)]
\end{aligned}
$$

Performing two 1D convolutions is much quicker than performing a single 2D convolution. A discrete 1D convolution with a kernel of size n requires n multiply and add operations. A discrete 2D convolution with a kernel of size $n \times n$ requires n^2 multiply and add operations. The speed-up offered by decomposing the 2D convolution is $n^2/2n = n/2$.

9. *Isotropic and directional edge finders*

The Laplacian of a Gaussian is given by

$$
\begin{aligned}
\nabla^2 G_\sigma(x,y) &= \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right) \left[\frac{1}{2\pi\sigma^2}\ \exp - \left(\frac{x^2+y^2}{2\sigma^2}\right)\right] \\
&= \frac{1}{2\pi\sigma^2} \left(\frac{x^2}{\sigma^4} + \frac{y^2}{\sigma^4} - \frac{2}{\sigma^2}\right) \exp - \left(\frac{x^2+y^2}{2\sigma^2}\right) \\
&= \frac{1}{2\pi\sigma^2} \left(\frac{r^2}{\sigma^4} - \frac{2}{\sigma^2}\right) \exp - \left(\frac{r^2}{2\sigma^2}\right) \quad \text{in polar coordinates}
\end{aligned}
$$

Since $\nabla^2 G_\sigma$ is independent of θ, it follows that the Laplacian of a Gaussian is an isotropic operator.

Consider rotating the edge until it is aligned with the image's y axis (since the Marr-Hildreth operator is isotropic, this does not entail a loss of generality): a step edge running along the line $x = 10$ is shown in Figure 1. Convolving with $\nabla^2 G_\sigma$ is equivalent to first smoothing the edge to produce a gentle edge (call this $S(x,y)$ — see Figure 1) and then computing its Laplacian $\partial^2 S/\partial x^2 + \partial^2 S/\partial y^2$.

Since S is constant in the y direction, $\partial^2 S/\partial y^2$ is zero everywhere. Also, since $\partial S/\partial x$ peaks at $x = 10$, $\partial^2 S/\partial x^2$ is zero at $x = 10$. Thus the response of the Marr-Hildreth operator is zero at a step edge. As we move away from $x = 10$ along the x-axis, $\partial^2 S/\partial x^2$ becomes non-zero, positive in one direction and negative in the other. Hence the operator produces zero-crossings along an ideal step edge.

The Canny edge detector searches for a maximum of $|\nabla (G_\sigma * I)|$ in the direction $\hat{\mathbf{n}}$. If s is a length parameter in the direction $\hat{\mathbf{n}}$, then the Canny operator looks for places

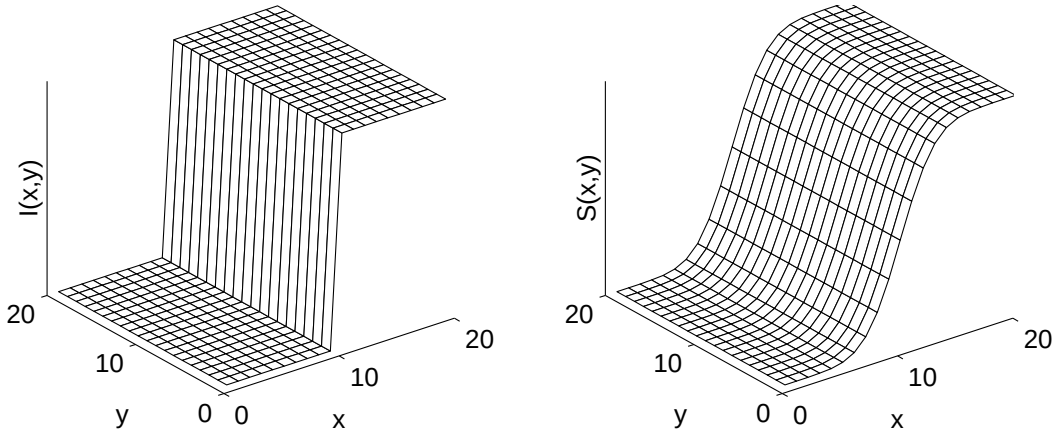

Figure 1: Smoothing an ideal step edge.

where

$$\frac{\partial}{\partial s}\left|\nabla\left(G_\sigma * I\right)\right| = 0 \tag{1}$$

From the definition of grad, we know that the maximum attainable value of a directional derivative $\partial\phi/\partial s$ is given by $|\nabla\phi|$. Hence

$$\frac{\partial}{\partial s}\left(G_\sigma * I\right) = \left|\nabla\left(G_\sigma * I\right)\right| \tag{2}$$

since s is in the direction of maximum intensity gradient. Substituting (2) into (1) gives

$$\frac{\partial}{\partial s}\left(\frac{\partial}{\partial s}\left(G_\sigma * I\right)\right) = 0$$

Hence the Canny edge detector finds zero-crossings in the directional second derivative of $(G_\sigma * I)$.

The principle advantage of the isotropic operator is computational efficiency: edge detection requires only a single convolution and the detection of zero-crossings. Conversely, the directional operator requires an additional, costly search for a local maximum normal to the gradient direction. The advantage of the directional operator is enhanced robustness to noise. Any differential operator amplifies noise. The Canny operator computes only first derivatives and then searches for a local maximum (which is equivalent to a zero-crossing of the second derivative) normal to the gradient. The Marr-Hildreth operator computes second derivatives both along and normal to the edge. Computation of the second derivative along the edge emphasizes noise in that direction while serving no purpose in edge detection.

10. *Corner detection*

(a) Since the eigenvectors $\mathbf{u}_1 \ldots \mathbf{u}_n$ of the real, symmetric matrix A form an orthonormal basis, we can decompose any vector $\mathbf{n}$ as follows:

$$\mathbf{n} = \sum_{i=1}^{n} c_i \mathbf{u}_i$$

If the corresponding eigenvalues are $\lambda_1 \ldots \lambda_n$, then

$$\mathrm{A}\mathbf{n} = \sum_{i=1}^{n} c_i \lambda_i \mathbf{u}_i$$

and

$$\mathbf{n}^T \mathrm{A}\mathbf{n} = \sum_{i=1}^{n} c_i^2 \lambda_i$$

Also

$$\mathbf{n}^T \mathbf{n} = \sum_{i=1}^{n} c_i^2$$

Putting all this together, we get

$$C = \frac{\mathbf{n}^T \mathrm{A}\mathbf{n}}{\mathbf{n}^T \mathbf{n}} = \frac{\sum_{i=1}^{n} c_i^2 \lambda_i}{\sum_{i=1}^{n} c_i^2}$$

If λ_1 is the minimum eigenvalue of A, then

$$\frac{\sum_{i=1}^{n} c_i^2 \lambda_i}{\sum_{i=1}^{n} c_i^2} \geq \frac{\sum_{i=1}^{n} c_i^2 \lambda_1}{\sum_{i=1}^{n} c_i^2}$$

Likewise, if λ_n is the maximum eigenvalue of A, then

$$\frac{\sum_{i=1}^{n} c_i^2 \lambda_i}{\sum_{i=1}^{n} c_i^2} \leq \frac{\sum_{i=1}^{n} c_i^2 \lambda_n}{\sum_{i=1}^{n} c_i^2}$$

Hence we conclude that

$$\lambda_1 \leq C \leq \lambda_n$$

(b) A corner detector needs to find points in the image where local values of $\nabla I(x, y).\mathbf{n}$ are not zero (or small) in any direction $\mathbf{n}$. First we calculate the change in intensity in direction $\mathbf{n}$:

$$I_n \equiv \nabla I(x, y).\hat{\mathbf{n}} \;\Rightarrow\; I_n^2 = \frac{\mathbf{n}^T \nabla I \, \nabla I^T \mathbf{n}}{\mathbf{n}^T \mathbf{n}} = \frac{\mathbf{n}^T \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \mathbf{n}}{\mathbf{n}^T \mathbf{n}}$$

where $I_x \equiv \partial I/\partial x$, etc. The directional derivatives I_x and I_y are estimated by convolving with kernels like the one in question 7. Next we smooth I_n^2 by convolution with a Gaussian kernel:

$$C_n(x,y) \equiv G_\sigma(x,y) * I_n^2 = \frac{\mathbf{n}^T \begin{bmatrix} \langle I_x^2 \rangle & \langle I_x I_y \rangle \\ \langle I_x I_y \rangle & \langle I_y^2 \rangle \end{bmatrix} \mathbf{n}}{\mathbf{n}^T \mathbf{n}}$$

where $\langle\ \rangle$ is the smoothed value. The smoothed values are obtained by discrete convolution with a truncated Gaussian kernel, as illustrated in question 5.

The smoothed change in intensity in direction $\mathbf{n}$ is therefore given by

$$C_n(x,y) = \frac{\mathbf{n}^T \mathrm{A} \mathbf{n}}{\mathbf{n}^T \mathbf{n}}$$

where A is the 2×2 matrix

$$\begin{bmatrix} \langle I_x^2 \rangle & \langle I_x I_y \rangle \\ \langle I_x I_y \rangle & \langle I_y^2 \rangle \end{bmatrix}$$

The result proved in (a) tells us that

$$\lambda_1 \leq C_n(x,y) \leq \lambda_2$$

where λ_1 and λ_2 are the eigenvalues of A. So, if we try every possible orientation $\mathbf{n}$, the maximum change in intensity we will find is λ_2, and the minimum value is λ_1.

We can therefore classify image structure at each pixel by looking at the eigenvalues of A:

No structure: (smooth variation) $\lambda_1 \approx \lambda_2 \approx 0$

1D structure: (edge) $\lambda_1 \approx 0$ (direction of edge), λ_2 large (normal to edge)

2D structure: (corner) λ_1 and λ_2 both large

It is necessary to calculate A at every pixel and mark corners where the quantity $\lambda_1\lambda_2 - \kappa(\lambda_1 + \lambda_2)^2$ exceeds some threshold ($\kappa \approx 0.04$ makes the detector a little edge-phobic, overcoming the "staircase effect" which makes corner detectors respond to discretised edges). Note that $\det \mathrm{A} = \lambda_1\lambda_2$ and trace $\mathrm{A} = \lambda_1 + \lambda_2$.

Roberto Cipolla and Andrew Gee
October 2006