C18 Computer Vision

David Murray

david.murray@eng.ox.ac.uk
www.robots.ox.ac.uk/~dwm/Courses/4CV

Michaelmas 2015

Computer Vision: This time ...

1. Introduction; imaging geometry; camera calibration
2. **Salient feature detection – edges, line and corners**
3. Recovering 3D from two images I: epipolar geometry.
4. Recovering 3D from two images II: stereo correspondence algorithms; triangulation.

Lecture 2

2.1 Cameras as photometric devices (just a note)

2.2 Image convolution

2.3 Edge detection, Line fitting

2.4 Corner detection

2.1 Cameras as photometric devices

In Lecture 1, we considered the camera as a geometric abstraction grounded on the rectilinear propagation of light.

But they are also photometric devices.

Important to consider the way image formation depends on:

- the nature of scene surface (reflecting, absorbing ...)
- the relative orientations surface, light source and camera
- the power and spectral properties of source
- the spectral properties of the imaging system.

The important overall outcome (eg, Forsyth & Ponce, p62) is that

image irradiance is proportional to the scene radiance

A relief! This means the image really can tell us about the scene.

Cameras as photometric devices /ctd

But the study of photometry (often called physics-based vision) requires *detailed* models of the reflectance properties of the scene and of the imaging process itself — the sort of models that underpin photo-realistic graphics too.

Eg, understanding how light scatters on water droplets allowed this image to be de-fogged

Original

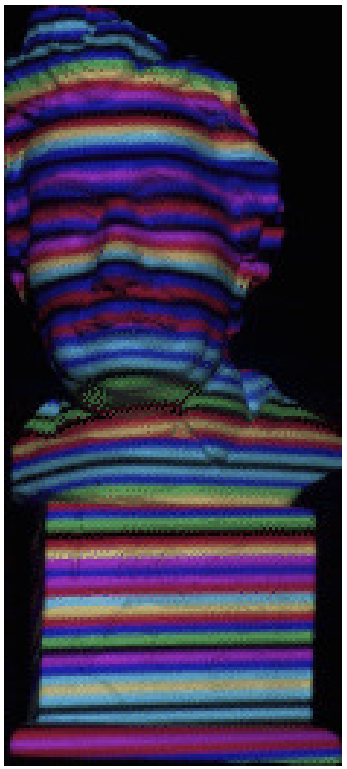
De-fogged

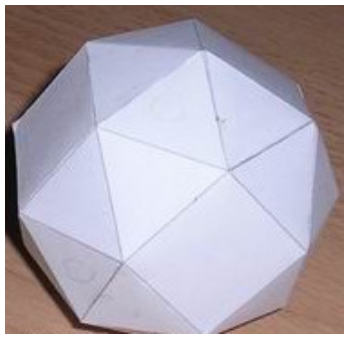
Can we avoid going into such detail? ... Yes — by considering aspects of scene geometry that are revealed in **step changes** in image irradiance.

Step irradiance changes are due to ...

1. Changes in Scene Radiance.
Natural (eg shadows) or deliberately introduced via artificial illumination: light stripes are generated either by a mask or from a laser.

2. Changes in the scene reflectance at sudden changes in surface orientation. These arise at the intersection of two surfaces, and thus represent geometrical entities fixed on the object in the scene.





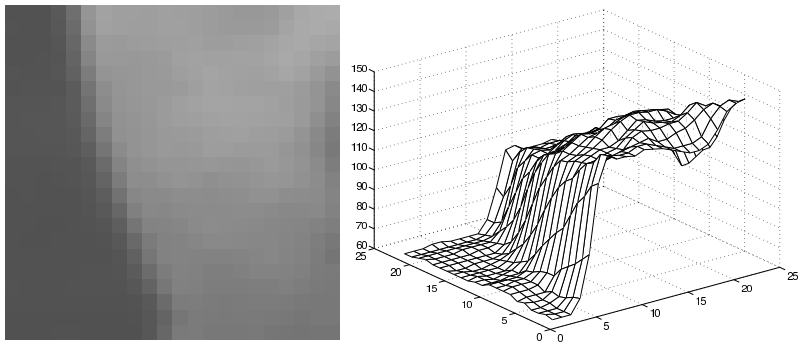
3. Changes in reflectance properties due to changes in surface albedo.
The reflectance properties are scaled by a changing albedo arising from surface markings. Again these are fixed to the object.

2.2 Feature detection

We are after step spatial changes in image irradiance because
(i) they are likely to be tied to scene geometry; and
(ii) they are likely to be salient (have high info content)

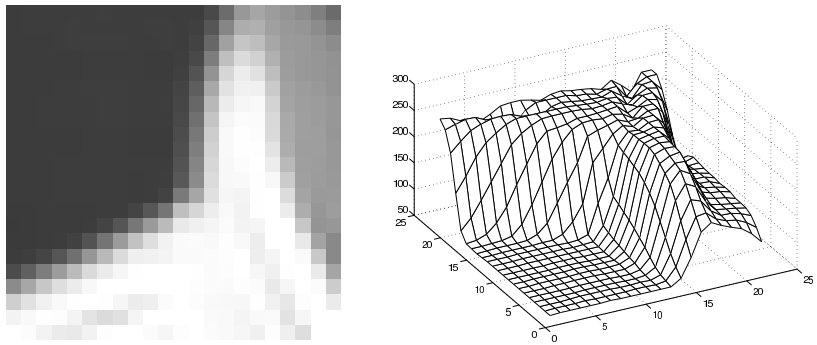
A simple classification of changes in image irradiance $I(x, y)$ is into areas that, locally, have

1D structure



$\Rightarrow$ **Edge Detectors**

2D structure



$\Rightarrow$ **Corner Detectors**

Image operations for Feature detection

Feature detection should be a **local** operation, working without knowledge of higher geometrical entities or objects ...
We should use pixel values $I(x, y)$ and derivatives $\partial I/\partial x$ and $\partial I/\partial y$, and so on.

It would be useful to have a non-directional combination of these, so that a feature map of a rotated image is identical to the rotated feature map of the original image: $\mathcal{F}(\boldsymbol{R}I) = \boldsymbol{R}\mathcal{F}(I)$

Considering edge detection, two possibilities are:

- Search for maxima in the gradient magnitude
  $\sqrt{(\partial I/\partial x)^2 + (\partial I/\partial y)^2}$ — 1st order, but non-linear

- Search for zeroes in the Laplacian
  $\nabla^2 I = \partial^2 I/\partial x^2 + \partial^2 I/\partial y^2$ — linear, but 2nd order

Which to choose?

The gradient magnitude is attractive because it is first order in the derivatives. Differentiation enhances noise, and the 2nd derivatives in the Laplacian operator introduces even more.

The Laplacian is attractive because it is linear, which means it can be implemented by a succession of fast linear operations – effectively matrix operations as we are dealing with a pixelated image.

Both approaches have been used.

For both methods we need to consider
☛ how to compute gradients, and
☛ how to suppress noise
so that insignificant variations in pixel intensity are not flagged up as edges.

This will involve spatial convolution of the image with an impulse response function that smooths the image and computes the gradients.

Preamble: spatial convolution

You are familiar with the 1D **convolution integral** in the time domain between a input signal $i(t)$ and **impulse response function** $h(t)$

$$o(t) = i(t) * h(t) = \int_{-\infty}^{+\infty} i(t-\tau)h(\tau)\mathrm{d}\tau = \int_{-\infty}^{+\infty} i(\tau)h(t-\tau)\mathrm{d}\tau \ .$$

The second equality reminds us that convolution commutes $i(t) * h(t) = h(t) * i(t)$. It also associates.

In the frequency domain we would write $O(s) = H(s)I(s)$.

Now in a continuous 2D domain, the **spatial convolution integral** is

$$o(x,y) = i(x,y) * h(x,y) = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} i(x-a,y-b)h(a,b)\mathrm{d}a\mathrm{d}b$$

In the spatial domain you'll often see $h(x,y)$ referred to as the **point spread function** or the **convolution mask**.

Spatial convolution /ctd

For pixelated images $I(x, y)$ we need a **discrete convolution**

$$O(x, y) = I(x, y) * h(x, y) = \sum_i \sum_j I(x - i, y - j) h(i, j)$$

for x, y ranging over the image width and height respectively, and i, j ensuring that access is made to any and all non-zero entries in h.

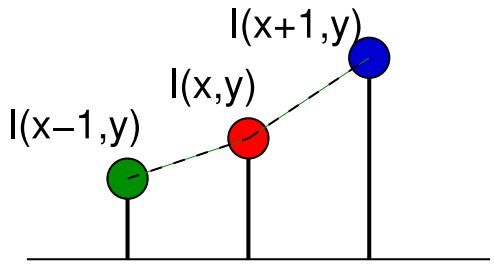
Many authors rewrite the convolution by replacing $h(i, j)$ with $\bar{h}(-i, -j)$

$$\begin{aligned} O(x, y) &= \sum_i \sum_j I(x - i, y - j) h(i, j) = \sum_i \sum_j I(x + i, y + j) h(-i, -j) \\ &= \sum_i \sum_j I(x + i, y + j) \bar{h}(i, j) \end{aligned}$$

This looks more like the expression for a cross-correlation but, confusingly, it is still called a convolution.

Computing partial derivatives using convolution

We can approximate $\partial I/\partial x$ at image pixel (x, y) using a central difference



$$\begin{aligned} \partial I/\partial x &\approx (1/2)\{[I(x+1,y) - I(x,y)] + [I(x,y) - I(x-1,y)]\} \\ &= (1/2)\{I(x+1,y) - I(x-1,y)\} \end{aligned}$$

Writing this as a "proper" convolution we would set

$$h(-1) = +\frac{1}{2} \quad h(0) = 0 \quad h(1) = -\frac{1}{2}$$

$$D(x,y) = I(x,y) * h(x) = \sum_{i=-1}^{1} I(x-i,y)h(i)$$

Notice how the "proper" mask is reversed from what we might naïvely expect from the expression.

Computing partial derivatives using convolution /ctd

Now, as ever,

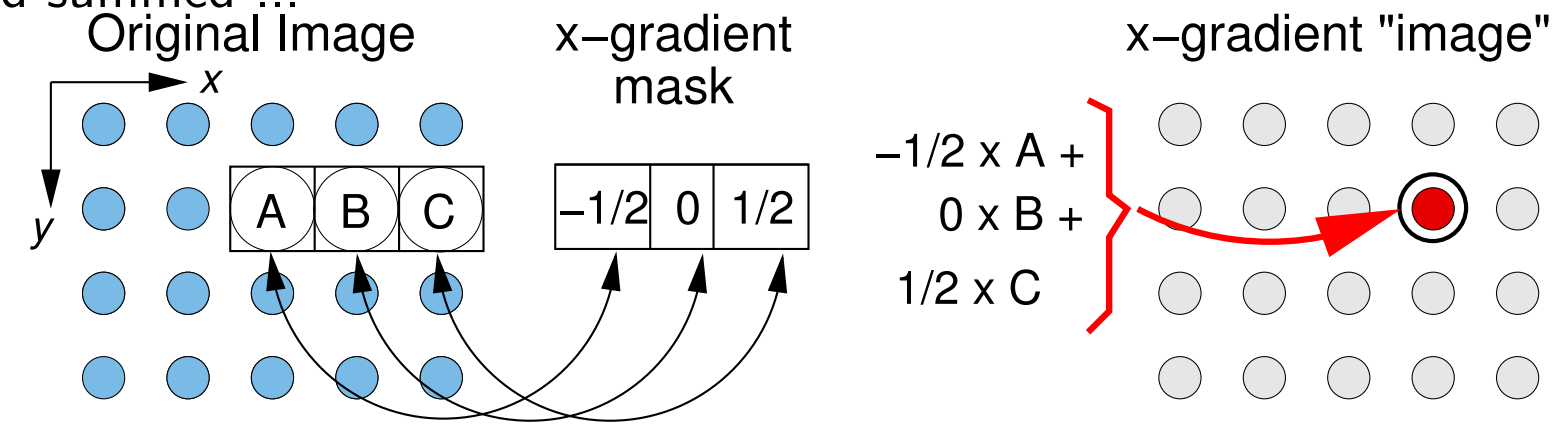
$$(\partial I/\partial x) \approx (1/2)[I(x+1,y) - I(x-1,y)]$$

Writing this as a "sort of correlation"

$$\bar{h}(-1) = -1/2 \quad \bar{h}(0) = 0 \quad \bar{h}(1) = +1/2$$

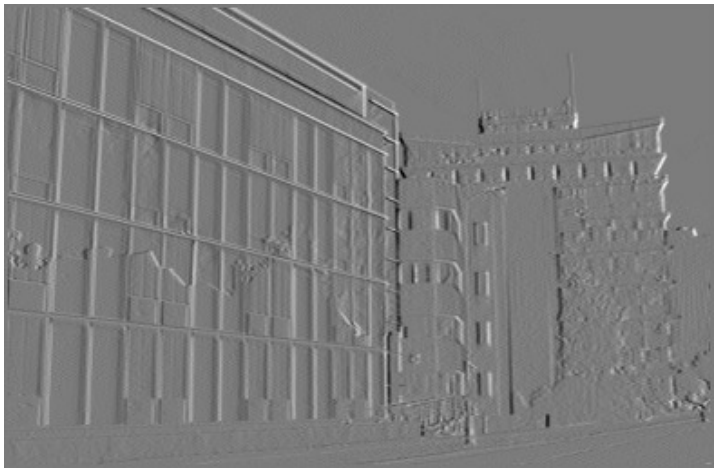
$$D(x,y) = I(x,y) * \bar{h}(x) = \sum_{i=-1}^{1} I(x+i,y)\bar{h}(i)$$

Note how we can just lay this mask directly on the pixels to be multiplied and summed ...

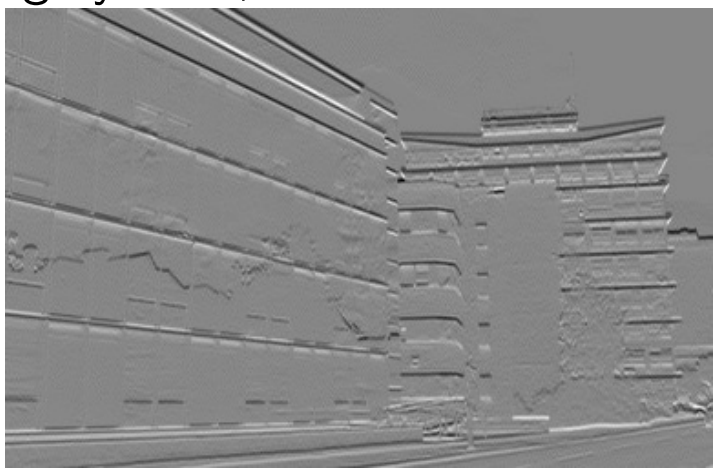


Example Results

The actual image used is grey-level, not colour



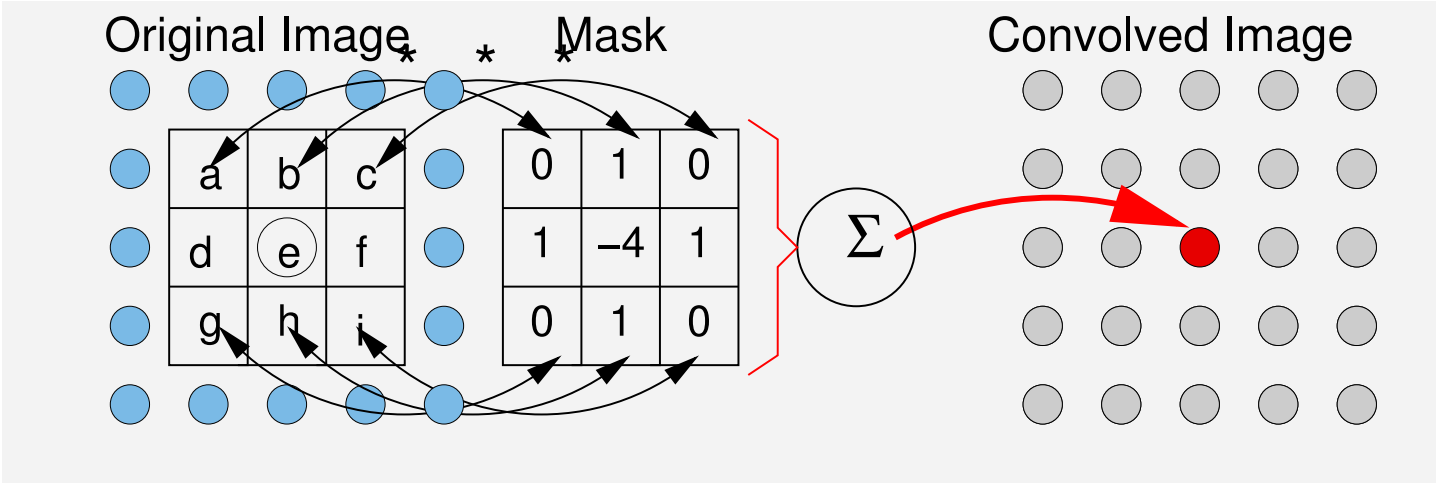
x-gradient "image"



y-gradient "image"

In 2 dimensions ...

As before, one imagines the flipped "correlation-like" mask centred on a pixel, and the sum of products computed.



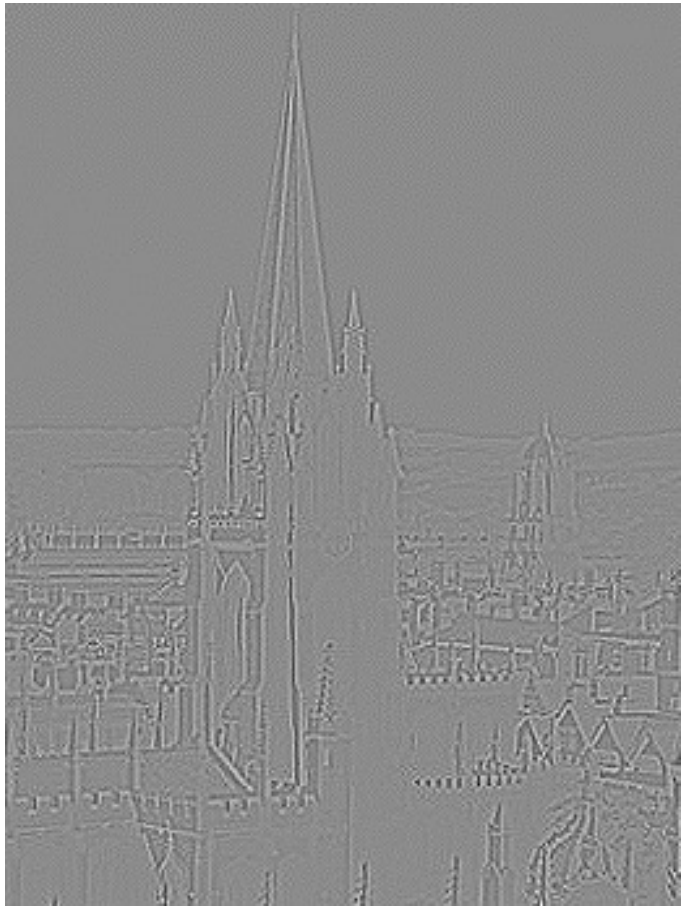
Often a 2D mask is "separable" in that it can be broken up in to two separate 1D convolutions in x and y

$$O = h_{2d} * I = f_y * g_x * I$$

The computational complexity is lower — but intermediate storage is required, and so for a small mask it may be cheaper to use it directly.

Example result — Laplacian (non-directional)

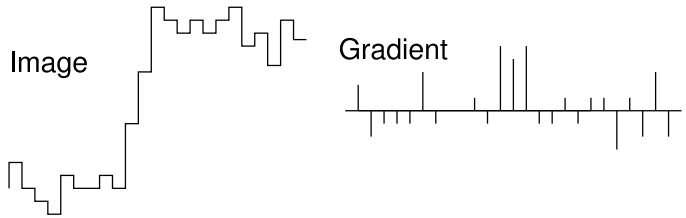
The actual image used is grey-level, not colour



Laplacian

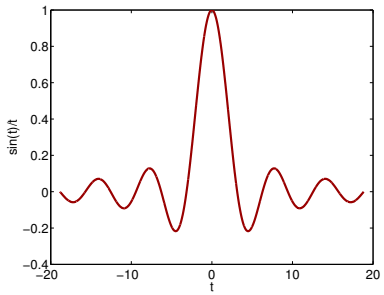
Noise and smoothing

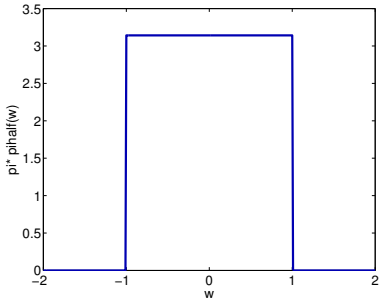
Differentiation enhances noise — the edge appears clear enough in the image, but less so in the gradient map.



If we knew the noise spectrum, we might find an optimal brickwall filter $g(x, y) \Leftrightarrow G(s)$ to suppress noise edges outside the signal edge band.

But a sharp cut-off in spatial-frequency, requires a wide spatial $g(x, y)$ — and Infinite Impulse Response (IIR) filter. Expensive to compute.





$$\operatorname{sinc} t) \quad \Leftrightarrow \quad \pi\Pi_{\frac{1}{2}}(\omega/2)$$

Conversely, if $g(x, y)$ has finite size (an FIR filter) it will not be band-limited, spatially blurring of the "signal" edges.

Can we compromise spread in space and spatial-frequency in some optimal way?

Compromise in space and spatial-frequency

Suppose our IR function is $h(x)$ and $h \rightleftharpoons H$ is a Fourier transform pair.

Define the spreads in space and spatial-frequency as X and Ω where

$$X^2 = \frac{\int (x - x_m)^2 h^2(x) dx}{\int h^2(x) dx} \quad \text{with} \quad x_m = \frac{\int x h^2(x) dx}{\int h^2(x) dx}$$

$$\Omega^2 = \frac{\int (\omega - \omega_m)^2 H^2(\omega) d\omega}{\int H^2(\omega) d\omega} \quad \text{with} \quad \omega_m = \frac{\int \omega H^2(\omega) d\omega}{\int H^2(\omega) d\omega}$$

Now vary h to minimize the product of the spreads $U = X\Omega$.
An uncertainty principle indicates that $U_{min} = 1/2$ when

$$h(x) = \textbf{A Gaussian function} = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{x^2}{\sigma^2}\right)$$

2.3 The Canny Edge Detector (JF Canny 1986)

2D Gaussian smoothing is applied to the image

$$S(x,y) = G(x,y) * I = \frac{1}{2\pi\sigma^2} \exp\left[-\frac{(x^2+y^2)}{2\sigma^2}\right] * I(x,y)$$

But separated into two separable masks

$$S(x,y) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{(x^2)}{2\sigma^2}\right] * \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{(y^2)}{2\sigma^2}\right] * I(x,y)$$

Then 1st derivatives in x and y are found by two further convolutions

$$\begin{aligned} D_x(x,y) &= \partial S/\partial x = h_x * S \\ D_y(x,y) &= \partial S/\partial y = h_y * S. \end{aligned}$$

Interestingly Canny's decision to use a Gaussian is made from a separate study of how to maximize robustness, localization, and uniqueness in edge detection.

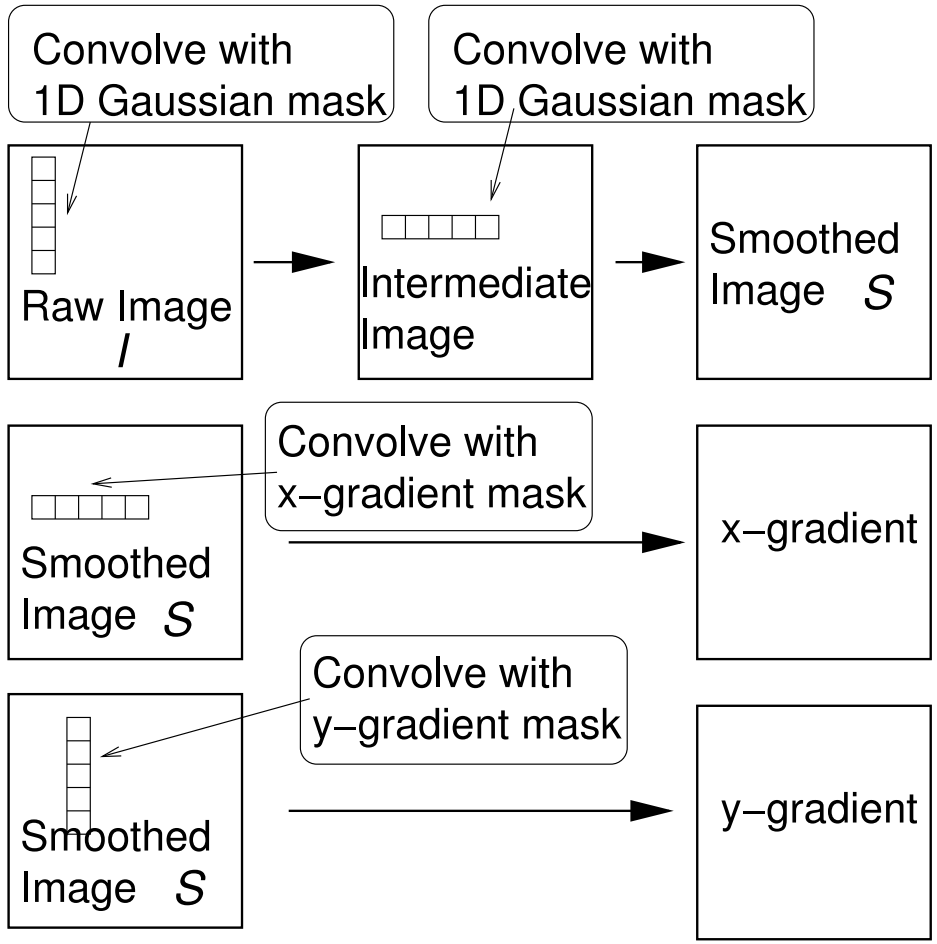
Canny/ Implementation

Step 1: Gaussian smooth image $S = G * I$

This is separable
$S = G_x * G_y * I$

Often 5×1 or 7×1 masks

Step 2: Compute gradients $\partial S/\partial x$, $\partial S/\partial y$ using derivative masks.



Canny/ Implementation

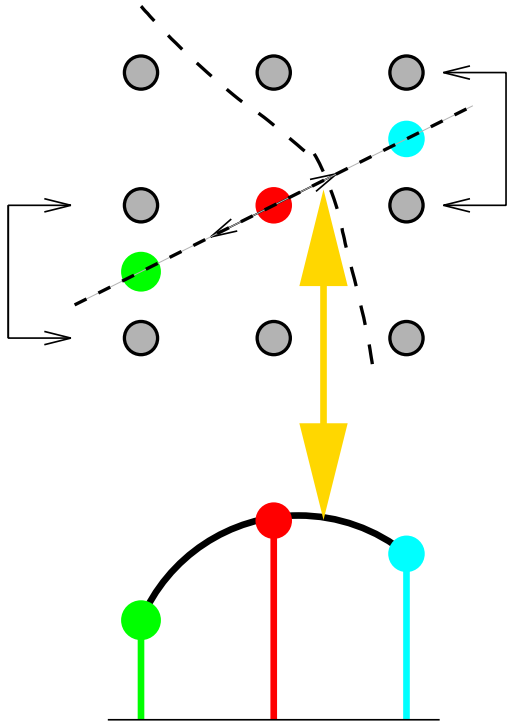
Step 3: Non-maximal suppression

a. Use $\partial S/\partial x$ and $\partial S/\partial y$ to find gradient magnitude and direction at a pixel.

b. Track along direction to marked positions on neighbourhood perimeter.

c. Linearly interpolate gradient magnitude at positions using magnitudes at arrow pixels.

d. If gradient magnitude at central pixel greater than both interpolated values, **declare an edgel**, and fit parabola to find (x, y) to sub-pixel acuity along the edge direction.

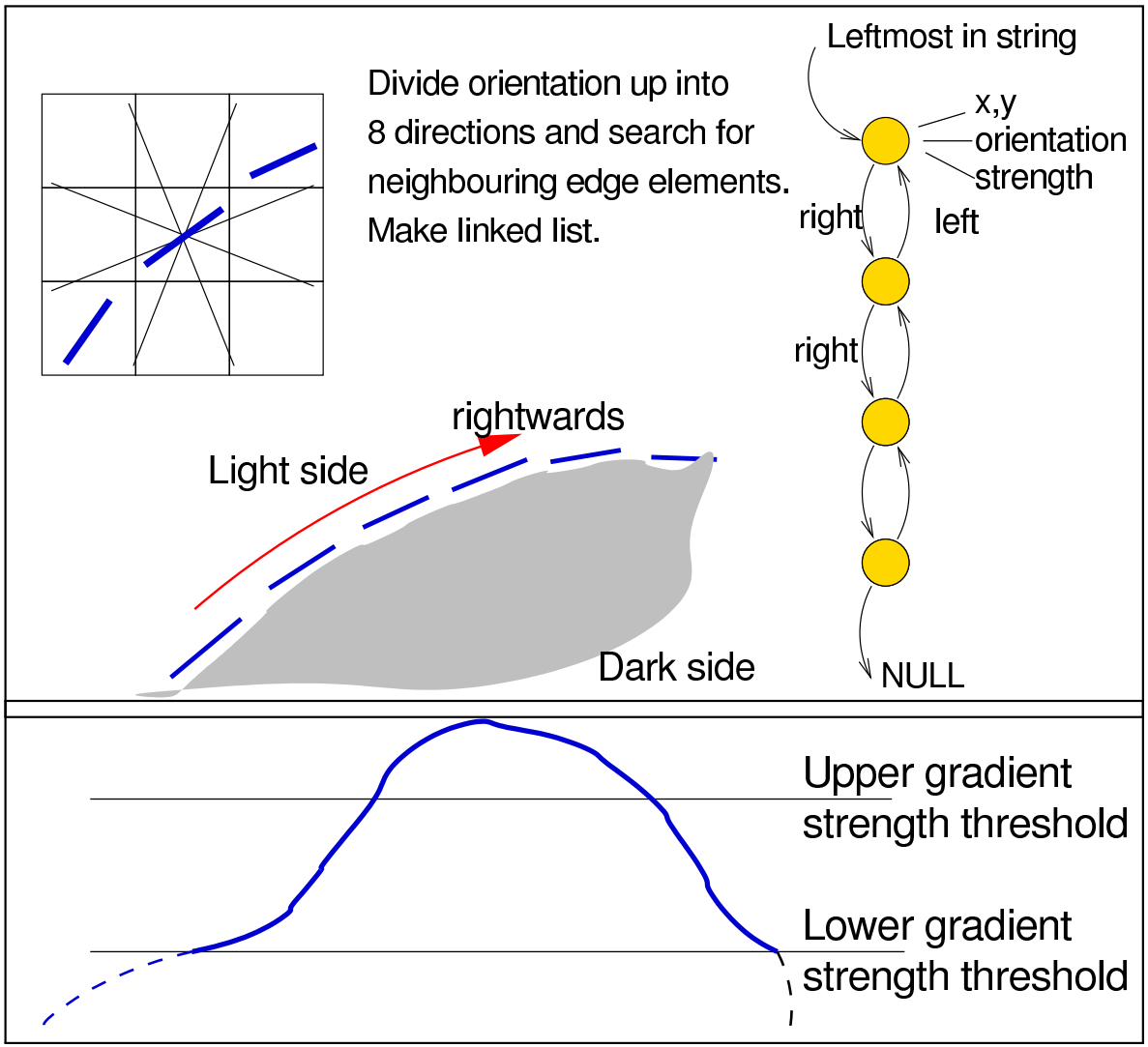


Step 4:

Store edgel's **sub-pixel position** (x, y); **gradient strength**; and **orientation** (full 360°as light/dark distinguished),

Canny/ Implementation: Hysterical Thresholding

Step 5:
Link edges together into strings using orientation to aid search for neighbouring edgels.



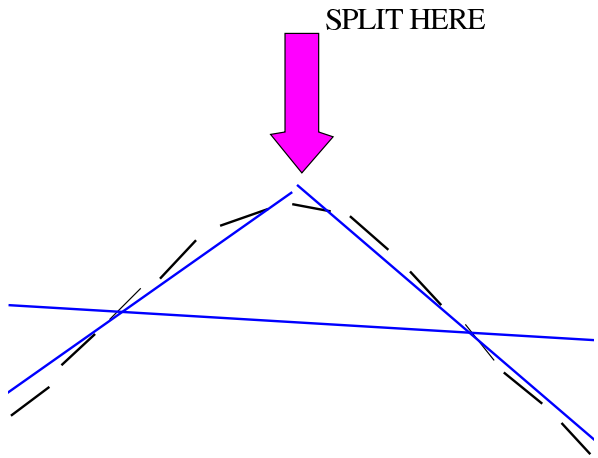
Step 6:
Perform thresholding with hysteresis, by running along the linked strings

Example of Edgel and String Outputs

From Strings to Straight Lines

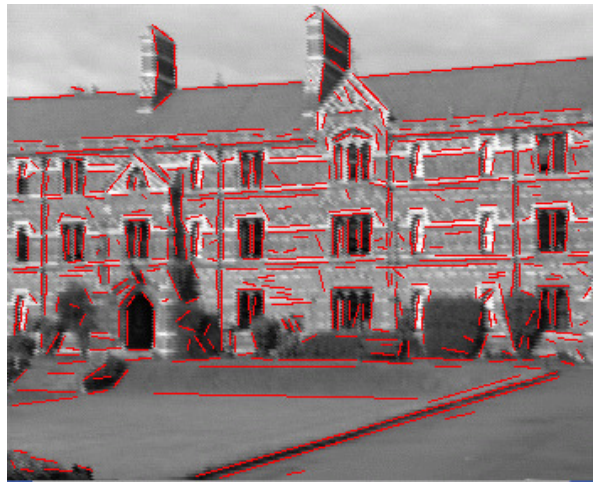
Split algorithm for lines:

1. Fit a straight line to all edgels in string
2. If RMS error less than threshold accept and stop.
3. Otherwise find point of highest curvature on edge string and split into two. Repeat from 1 for each substring.



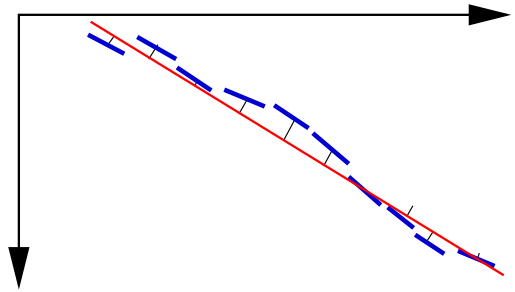
Merge algorithm for lines:

1. Fit straight lines to each pair of consecutive edgels in a string
2. Compute RMS error for each potential merger of an adjacent pair of lines into a single line, and find the pair for which the RMS error is minimum
3. If RMS error less than threshold, merge and repeat from 2.



Actually fitting lines and finding that RMS error

Consider n "edgels" that have been linked into an edge (sub-)string.



Fitting a line requires **orthogonal regression**.
See B14, or the variant here ...

Find

$$\min E(a, b, c) = \sum_{i=1}^{n} (ax_i + by_i + c)^2$$

subject to $a^2 + b^2 = 1$

Hence a 2-dof problem.

From edge elements to lines /ctd

Now $\min E(a,b,c) = \min \sum_{i=1}^{n} (ax_i + by_i + c)^2$ requires $\partial E/\partial c = 0$.

$$\Rightarrow 2\sum_{i=1}^{n} (ax_i + by_i + c) = 0 \quad \Rightarrow c = -(a\bar{x} + b\bar{y}) \ .$$

Therefore E becomes

$$E = \sum_{i=1}^{n} [a(x_i - \bar{x}) + b(y_i - \bar{y})]^2 = \begin{bmatrix} a & b \end{bmatrix} \boldsymbol{U}^T \boldsymbol{U} \begin{bmatrix} a \\ b \end{bmatrix}$$

where the 2nd-moment matrix is

$$\boldsymbol{U}^T \boldsymbol{U} = \begin{bmatrix} \sum_{i=1}^{n} x_i^2 - n\bar{x}^2 & \sum_{i=1}^{n} x_i y_i - n\bar{x}\bar{y} \\ \sum_{i=1}^{n} x_i y_i - n\bar{x}\bar{y} & \sum_{i=1}^{n} y_i^2 - n\bar{y}^2 \end{bmatrix},$$

Solution a, b given by the unit eigenvector of $\boldsymbol{U}^T \boldsymbol{U}$ corresponding to smaller eigenvalue. (Eigen-decomposition or SVD.)

RMS error given by the eigenvalue itself.

Example of String and Line Outputs

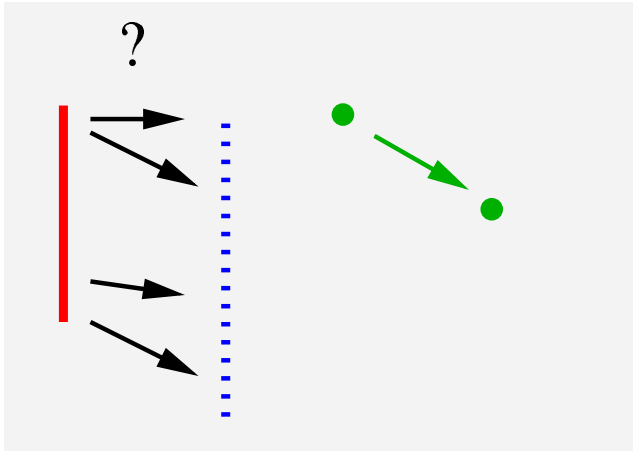
Problems using 1D image structure for Geometry

Computing edges make the feature map sparse but interpretable. Much of the salient information is retained.

If the camera motion is known, feature matching is a 1D problem, for which edges are very well suited (see Lecture 4).

However, matching is much harder when the camera motion is unknown: known as the Aperture problem.

End points are unstable, hence line matching is largely uncertain. (Indeed only line *orientation* is useful for detailed geometrical work.)

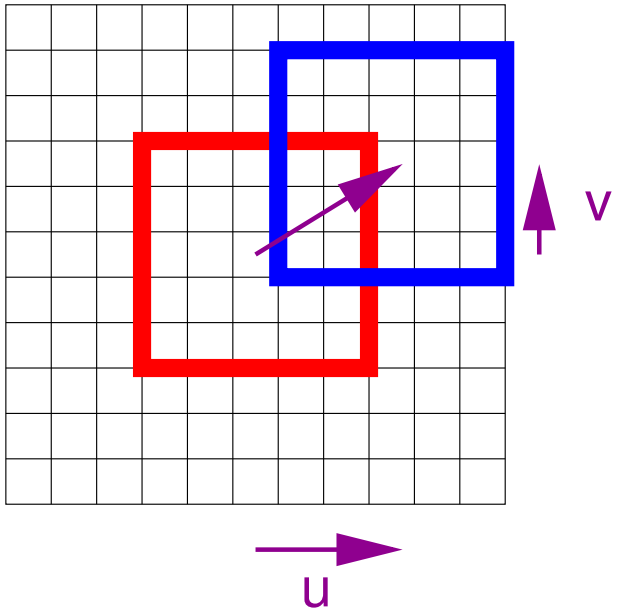


In general, matching requires the unambiguity of 2D image features or "corners".

Corners defined as sharp peaks in the 2D autocorrelation of local patches in the image.

2.5 Corner detection: preamble on auto-correlation

Suppose that we are interested in correlating a $(2n+1)^2$ pixel patch at (x, y) in Image I with a similar patch displaced from it by (u, v).



We would write the correlation between the patches as

$$C_{uv}(x, y) = \sum_{i=-n}^{n} \sum_{j=-n}^{n} I(x+i, y+j) I(x+u+i, y+v+j)$$

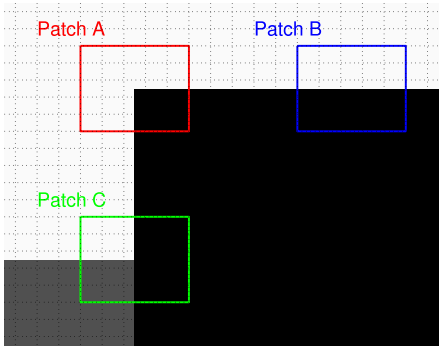
As we keep (x, y) fixed, but change (u, v), we build up the **auto-correlation surface** around (x, y).

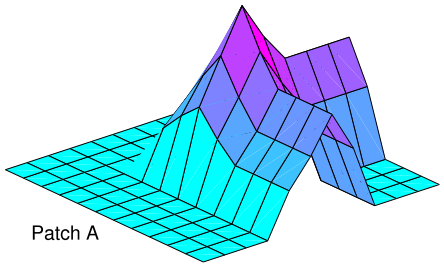
Preamble: Auto-correlation

Reminder **correlation:**

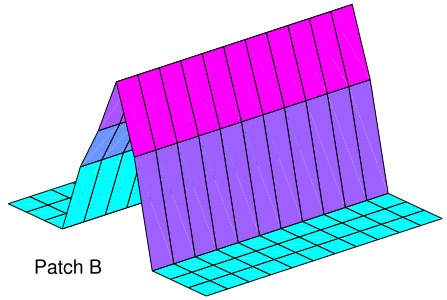
$$C_{uv}(x,y) = \sum_{i=-n}^{n} \sum_{j=-n}^{n} I(x+i, y+j) I(x+u+i, y+v+j)$$

Here are the surfaces around 3 different patches

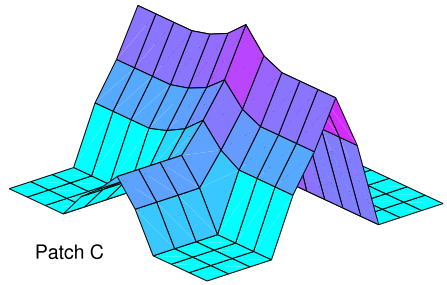




Plain corner



Straight edge



3-way corner

A pixel in a uniform region will have a flat autocorrelation

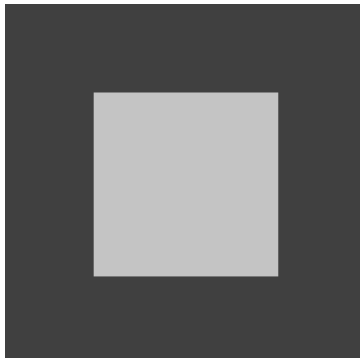
A pixel on an edge will have a ridge-like autocorrelation, but

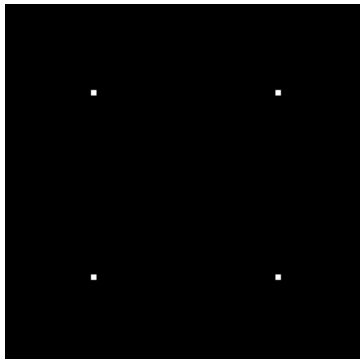
A pixel at a corner has a peak.

Results ...

So a simple corner detector might have the following steps at each pixel

a. Determine the auto-correlation $C_{uv}(x, y)$ around the pixel position x, y.
b. Find positions x, y where $C_{uv}(x, y)$ is maximum in two directions.
c. If $C_{uv}(x, y) > \text{threshold}$ mark as a corner.





There is an expression which can be computed more cheaply than C_{uv} which gives comparable qualitative results.
This is the **sum of squared differences**

$$E_{uv}(x, y) = \sum_{i=-n}^{+n} \sum_{j=-n}^{+n} [I(x + u + i, y + v + j) - I(x + i, y + j)]^2$$

Harris Corner Detector (Chris Harris, 1987)

Earlier we estimated the gradient from the pixels values.
Now assume we know the the gradients, and estimate a pixel difference using a 1st order Taylor expansion ...

$$I(x+u, y+v) - I(x,y) \approx u\frac{\partial I(x,y)}{\partial x} + v\frac{\partial I(x,y)}{\partial y}$$

So the sum of squared differences can be approximated as

$$\begin{aligned} E_{uv}(x,y) &= \sum_{i=-n}^{+n}\sum_{j=-n}^{+n}\left(I(x+u+i, y+v+j) - I(x+i, y+j)\right)^2 \\ &\approx \sum_i\sum_j\left(u\frac{\partial I}{\partial x} + v\frac{\partial I}{\partial y}\right)^2 \\ &= \sum_i\sum_j\left(u^2\left(\frac{\partial I}{\partial x}\right)^2 + 2uv\frac{\partial I}{\partial x}\frac{\partial I}{\partial y} + v^2\left(\frac{\partial I}{\partial y}\right)^2\right) \end{aligned}$$

Note, $\partial I/\partial x$ etc are computed at the relevant $(x+i, y+j)$.

Harris Corner Detector /ctd

The double sum over i,j is replaced by a convolution with

$$W = \begin{array}{|c|c|c|c|c|}\hline 1&1&1&1&1\\\hline 1&1&1&1&1\\\hline 1&1&1&1&1\\\hline 1&1&1&1&1\\\hline 1&1&1&1&1\\\hline\end{array}$$

$$\begin{aligned} E_{uv}(x,y) &= \sum_i \sum_j (u^2 \left(\frac{\partial I}{\partial x}\right)^2 + 2uv \frac{\partial I}{\partial x}\frac{\partial I}{\partial y} + v^2 \left(\frac{\partial I}{\partial y}\right)^2) \\ &= u^2 (W * \left(\frac{\partial I}{\partial x}\right)^2) + 2uv (W * \frac{\partial I}{\partial x}\frac{\partial I}{\partial y}) + v^2 (W * \left(\frac{\partial I}{\partial y}\right)^2) \\ &= \begin{pmatrix} u & v \end{pmatrix} \begin{pmatrix} p & r \\ r & q \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} \end{aligned}$$

where

$$\begin{aligned} p(x,y) &= W * (\partial I/\partial x)^2 \\ q(x,y) &= W * (\partial I/\partial y)^2 \\ r(x,y) &= W * (\partial I/\partial x)(\partial I/\partial y) \end{aligned}$$

Harris Corner Detector /ctd

We can introduce smoothing by replacing W with a Gaussian G, so that

$$E_{uv}(x,y) = \begin{pmatrix} u & v \end{pmatrix} \begin{pmatrix} p & r \\ r & q \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix}$$

where

$$\begin{aligned} p(x,y) &= G * (\partial I/\partial x)^2 \\ q(x,y) &= G * (\partial I/\partial y)^2 \\ r(x,y) &= G * (\partial I/\partial x)(\partial I/\partial y) \end{aligned}$$

The quantities p, q and r computed at each (x,y) define the shape of the auto-correlation function $E_{uv}(x,y)$ at (x,y)

Harris Corner Detector /ctd

Recall that

$$E_{uv}(x,y) = \begin{pmatrix} u & v \end{pmatrix} \begin{pmatrix} p & r \\ r & q \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix}$$

Now (u, v) defines a direction, so an estimate of E_{uv} a unit distance away from (x, y) along a vector direction $\mathbf{n}$ is then

$$E \approx \frac{\mathbf{n}^\top \boldsymbol{S} \mathbf{n}}{\mathbf{n}^\top \mathbf{n}} \quad \text{where} \quad \boldsymbol{S} = \begin{pmatrix} p & r \\ r & q \end{pmatrix}$$

Now recall (eg, from A1 Eng Comp notes) that if λ_1 and λ_2 are the larger and smaller eigenvalues of $\boldsymbol{S}$, respectively, that

$$\lambda_2 \leqslant \frac{\mathbf{n}^\top \boldsymbol{S} \mathbf{n}}{\mathbf{n}^\top \mathbf{n}} \leqslant \lambda_1 \quad \Rightarrow \quad \lambda_2 \leqslant E \leqslant \lambda_1$$

/ctd Harris Corner Detector

This allows a classification of image structure:

- Both $\lambda_1, \lambda_2 \approx 0$ — the autocorrelation is small in all directions: image must be flat.
- $\lambda_1 \gg 0, \lambda_2 \approx 0$ — the autocorrelation is high in just one direction $\Rightarrow$a 1D edge
- $\lambda_1 \gg 0, \lambda_2 \gg 0$ — the autocorrelation is high in all directions $\Rightarrow$a **2D corner**.

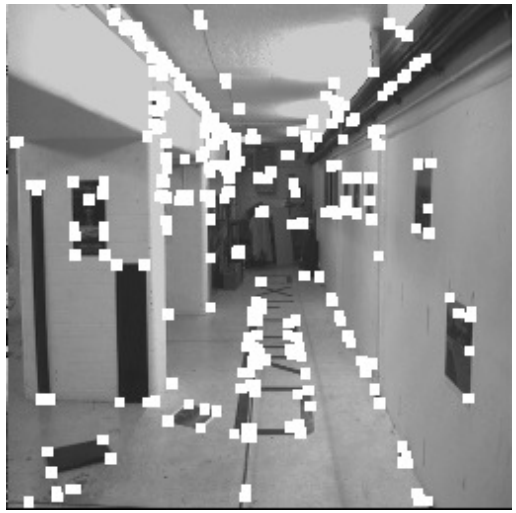
Harris' original "interest" score was later modified by Harris and Stephens

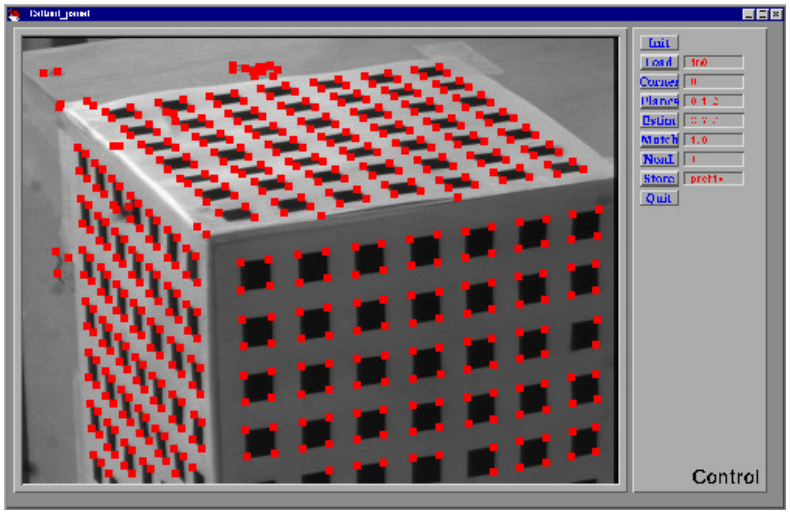
$$S_{Harris} = \frac{\lambda_1 \lambda_2}{\lambda_1 + \lambda_2} \qquad S_{HS} = \lambda_1 \lambda_2 - \alpha \frac{(\lambda_1 + \lambda_2)^2}{4}$$

α is a positive constant $0 \leqslant \alpha \leqslant 1$ which decreases the response to edges, sometimes called the "edge-phobia".

With $\alpha > 0$ edges give negative scores, and corners positive. Scores close to zero indicate flat surfaces or "T" features which are intermediary between edges and corners. The size of α determines how much edges are penalised.

Examples




Exit
Load
Corner
Planes
Optical
Match
NonL
Store
Quit
Control

Summary

In this lecture we have considered

- As enabling techniques, we have described convolution and correlation.
- Described how to smooth imagery and detect gradients
- Described how to recover 1D structure (edges) in an image
  In particular, have developed the Canny Edge detector.
- Considered how to join edgels into strings, and to fit lines.
- Described how to recover 2D structure (corners) in an image.
  Have developed the Harris corner detector