CT336/CT404
Graphics & Image Processing

Section 10

Segmentation

(including some image sequence / video techniques)

Typical Workflow in Machine Vision

1. Clean up
   - reduce noise
   - remove geometric/radiometric distortion
   - emphasise desired aspects of image
2. Segmentation
   - identify/extract objects of interest
   - sometimes the entire image is ‘of interest’ and the task is to separate it into disjoint (non-overlapping) regions
   - probably leverages domain-specific knowledge
3. Measurement
   - quantify appropriate measurements on segmented objects
4. Classification:
   - assign segmented objects to classes
   - make decisions etc.

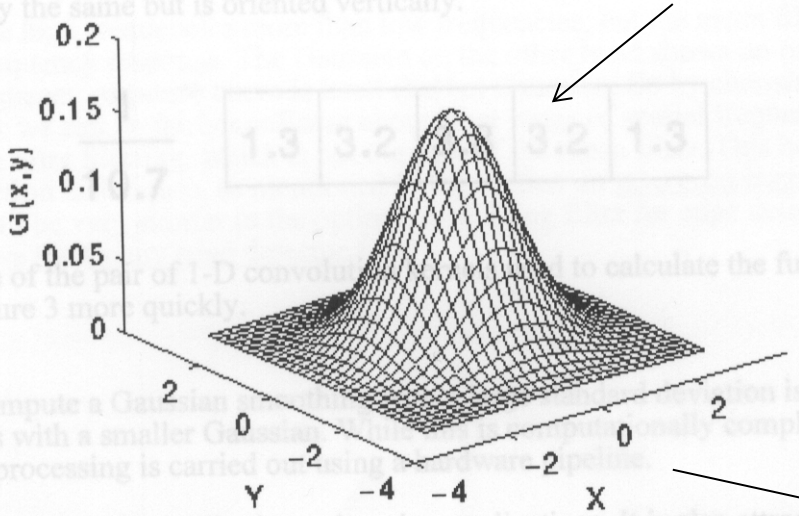
Noise Reduction

- Typically use convolution filters to smooth the image, i.e. dampen high-frequency noise without unduly damaging larger (low-frequency) objects
- Blur - averages a pixel and its neighbours
- Median - replaces a pixel with the median, rather than the mean, of the pixel and its neighbours
- Gaussian - a filter that produces a smooth response (unlike blur/'box' and median filtering) by weighting more towards the centre

Gaussian Convolution Kernel

Blur/box filter (viewed in 1D)

Gaussian (viewed in 2D)



A kernel for Gaussian smoothing with standard deviation of 1.4 pixels:

$$\frac{1}{115}$$

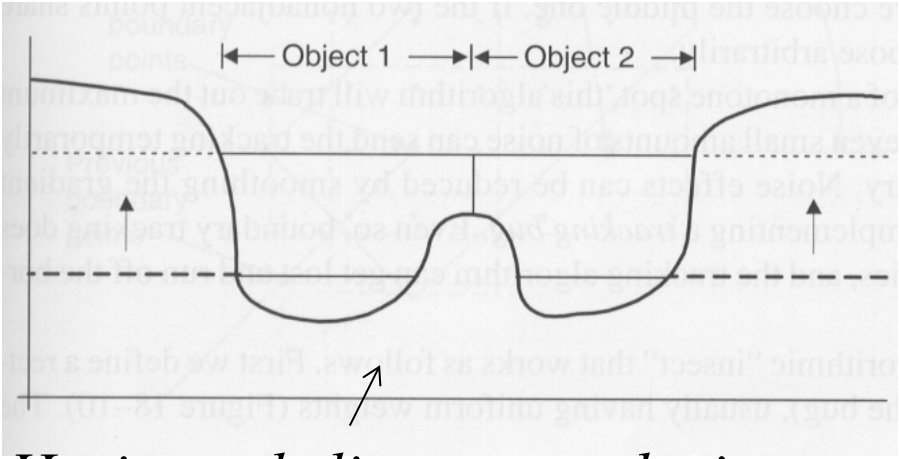
2	4	5	4	2
4	9	12	9	4
5	12	15	12	5
4	9	12	9	4
2	4	5	4	2

Thresholding

- May be an appropriate segmentation technique where the objects of interest lie on a sharply contrasting background
- A bi-modal greyscale histogram may indicate this
- Some post-thresholding cleanup may be useful (e.g. using morphological techniques such as erosion and dilation)
- Thresholding is a global/point operation, and like other global operations it may be employed in a *locally adaptive* form:
    - E.g. if the background of an image varies slowly from dark to bright, it may still be possible to segment objects of interest
    - divide the image into small rectangular sections, which may well display cleaner bimodal histograms than the entire image. Different threshold values may then be chosen for each section. Sections that don't exhibit bimodal histograms may have their threshold levels interpolated from neighbouring sections

A Watershed Algorithm with Thresholding

- Deals with objects that are too close together for simple thresholding to tell them apart
- 'Desired grey level' often chosen as either the lowest level between histogram peaks, or halfway between the peaks



Horizontal slice across the image. We start with a low grey level (dashed line), and gradually rise to the desired grey level (dotted line). The boundary between the 2 objects will remain identified.

Edge Detection

- A common early step in segmentation tasks (usually preceded by noise reduction – smoothing, etc.)
- Determines how different pixels are from their neighbours: abrupt changes in brightness are interpreted as the edges of objects
- The aim is to provide evidence for the automatic delineation of objects in a scene, which are assumed to be characterised by their edges
- Convolution kernels for edge detection are classified as *gradient magnitude* type or *gradient direction (compass)* type.

Gradient Magnitude Edge Detection

- Convolves the image with either 1 or 2 kernels
- In the latter case, one kernel estimates the 1st derivative in the *x*-direction, and the other in the *y*-direction: the gradient magnitude (edge strength) is calculated by combining the results of the two.
- E.g. the Sobel operator

$$X = \frac{1}{4}\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

$$Y = \frac{1}{4}\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

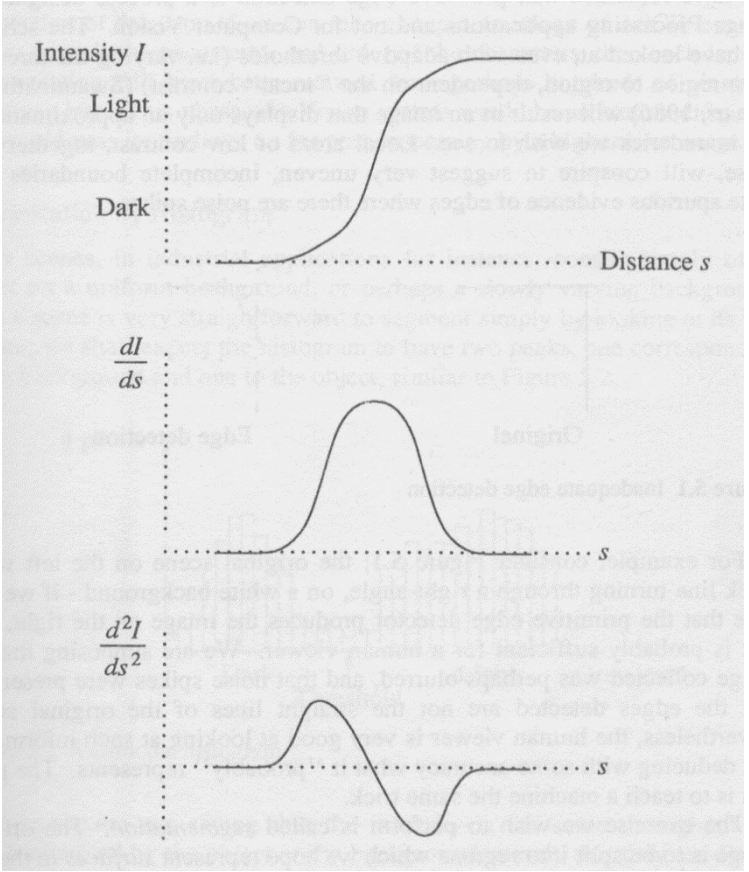
- The 'X' kernel is maximised by dark pixels on the left and bright pixels on the right; the opposite gives a large negative value, while no edge evidence is implied by a value in the middle, close to zero
- The edge magnitude calculation combines the two kernel responses, and also removes negative values

The edge magnitude is taken as:

$$\sqrt{X^2 + Y^2}$$

Edge Detection Filters as Mathematical Derivatives

- Consider a horizontal slice across the image
- Edge detection filters are essentially performing differentiation of grey level with respect to distance
    - i.e. ‘how different is a pixel to its neighbours’?
- Some filters are akin to first derivatives, while others are akin to second derivatives



Laplacian Edge Detection

- The Laplacian is a gradient filter that has the advantage that it is an isotropic (omnidirectional) approximation of the 2nd derivative of an image, i.e. the edge magnitude is obtained independently of the edge orientation through the application of a single kernel
- It was developed from Marr's theoretical work on human vision systems ('Vision', c.1980: psychology/physiology)
- The Laplacian can be approximated at different sizes, e.g. this 3x3 convolution kernel:

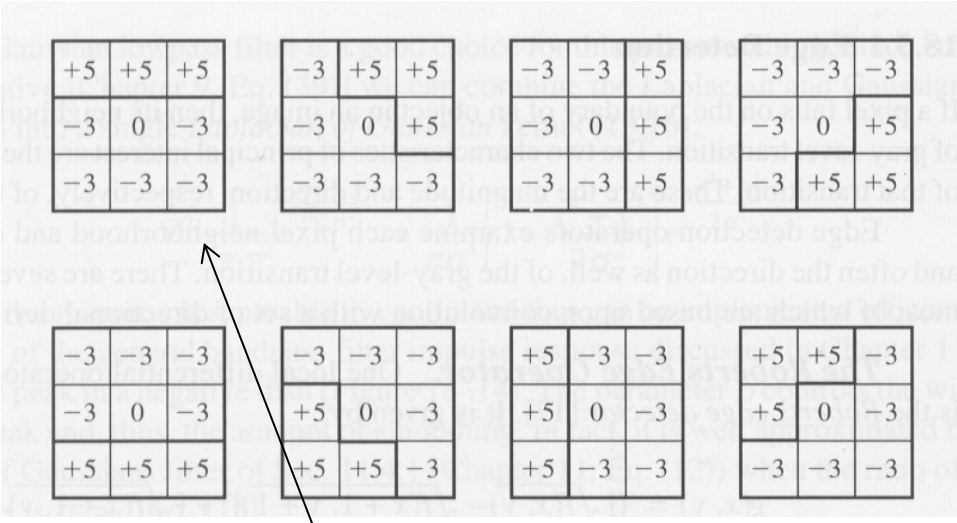
-1	-1	-1
-1	8	-1
-1	-1	-1

Laplacian of Gaussian..

- Since the Laplacian is a second derivative measurement, it is highly sensitive to noise: therefore, its application is normally preceded by Gaussian smoothing.
- Since convolution is an associative operation, a hybrid filter that approximates both operations - the LoG - may be generated.
- Due to its smoothing function, and to the fact that it responds to increasing *and* decreasing image intensity, the LoG filter produces edge maps with multiple responses to each 'true' edge - characterised by multiple-pixel wide, inexact edges.
- But it works quite well at identifying objects at reasonably low contrast.. insofar as a simple convolution filter can do

Gradient Direction (Compass) Edge Detection

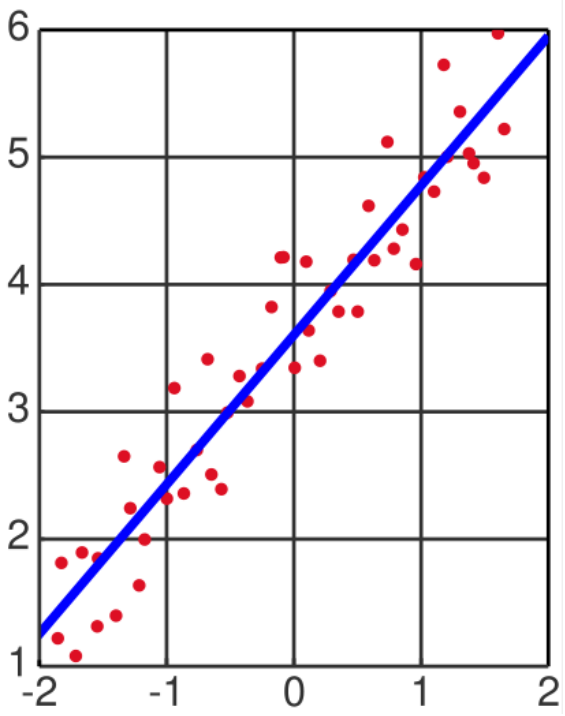
- Compass edge detection convolves the image with (usually 8) kernels, each of which is sensitive to a different edge orientation
- The kernel producing the maximum response at each pixel determines the edge magnitude *and direction* at that pixel.
- Edge direction may give useful evidence for further processing (e.g. boundary tracking)



E.g. with the Kirsch operator, the top-left kernel, for example, is maximised by a bright object vertically above a dark object.

Least-Squares approach to estimating Gradient Direction

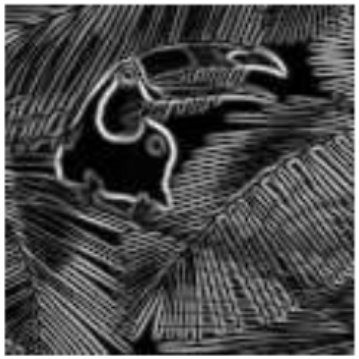
- Another approach, which may more accurately locate edge directions, begins with a standard gradient magnitude filter and theshold operation, and then locates and orients edges using least-squares fitting in a window (e.g. 5x5 pixels) around each edge point



Limitations of Edge Detection Operators

Original

Sobel



Threshold of Sobel

Typical problems:

- Arbitrary threshold value
- Thick edges (caused by low threshold and/or effects of smoothing step)
- Broken edges (caused by low contrast, or occlusion)
- Noise points
- Clutter (due to having no control over precisely what's in the scene)

Boundary Tracking / Canny Algorithm

- Locate an edge pixel on the boundary of an object, and iteratively search for neighbouring pixels that are assumed to be also on the boundary.
- Canny algorithm:
  - Apply smoothing (e.g. Blur) and edge enhancement (e.g. Sobel)
  - Use non-maximal suppression to thin the extracted edges to single pixel width by tracking ‘gradient magnitude ridges’, and setting all pixels in their neighbourhoods with lower values to zero
  - The tracking process only starts at points with an edge strength greater than a preset threshold, but tracks ridges as long as their edge strength does not fall below a lower preset threshold.
  - This 'hysteresis' process helps to ensure that noisy edges are not broken up into multiple edge fragments.

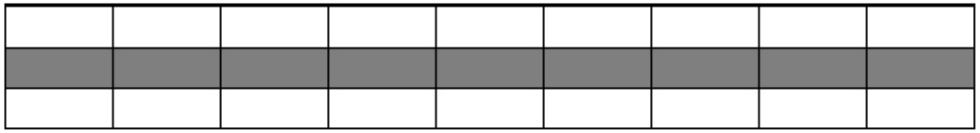
Hysteresis

With thresholding, this is an ‘ill-posed problem’, i.e. a correct threshold value does not exist

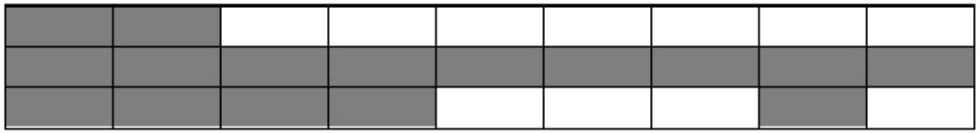
E.g. consider this edge map, in which a horizontal line is surrounded by noise:

8	9	4	5	6	5	2	3	4
10	10	9	8	8	7	8	9	10
8	8	7	7	6	5	4	7	3

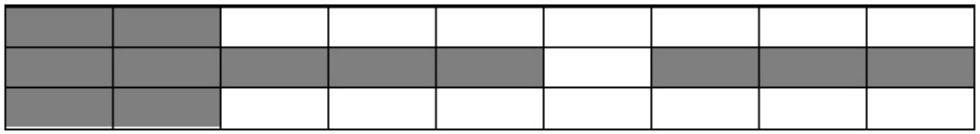
The Canny algorithm, applied with upper and lower hysteresis values of 10 and (say) 5, produces:



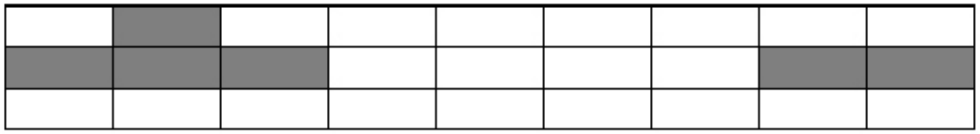
By comparison, thresholding of the original edge map at level 7 produces:



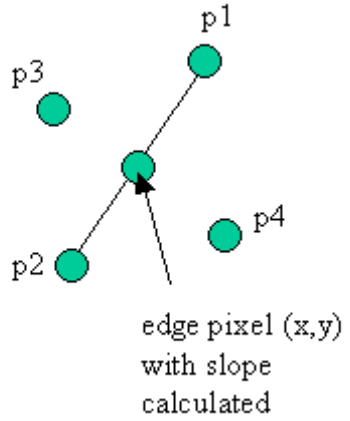
Thresholding of the original edge map at level 8 produces:



Thresholding of the original edge map at level 9 produces:



The Canny Algorithm



- Given: Greyscale 2-dimensional array P, upper hysteresis threshold value H_U, lower hysteresis threshold value H_L
- Apply Gaussian smoothing to P
- Compute Sobel edge detection for x and y directions, yielding two arrays of directional gradients, Gx and Gy
- Compute aggregate edge gradient array G and edge slope array S, where the gradient at each pixel is and the slope at each pixel is $90^o+arctan(Gy/Gx)$
- For each pixel (x,y) in G whose value is $> H_U$, perform edge tracing:
    - Interpolate greyscale values $p1$ and $p2$, in each direction along the line of slope at (x,y), and greyscale values $p3$ and $p4$ at points perpendicular to this line of slope
    - If $max(p1,p2) > max(p3,p4)$ and $max(p1,p2) > H_L$ then mark (x,y) as an edge pixel, mark pixels perpendicular to line as non-edges, and continue edge tracing along the line of slope
    - Else mark (x,y) as a non-edge pixel and finish edge tracing

The Canny Algorithm

Highly regarded for a number of reasons:

- It has good ability to locate as many edges as possible
- It is relatively insensitive to noise
- There will be a minimal distance between its detected edges and the real edges (important when you want to measure/classify extracted objects)
- It gives only one response to each edge

See demo here:
https://github.com/inspirit/jsfeat#readme
https://inspirit.github.io/jsfeat/sample_canny_edge.html

Edge Detection Post-Processing

- Following smoothing, edge detection, and morphological clean-up and/or non-maximal suppression of an image, it is still required to find and extract the objects of interest.
- Relaxation (line-linking) techniques extrapolate partial lines or line segments in order to "fill in the gaps" and thereby assist the object detection task (goal: form closed loops)
- Simple low-level relaxation techniques attempt to join neighbouring edge fragments without the use of gradient direction information
- More advanced techniques use both gradient strength and direction to improve edge confidence and direct the relaxation effort

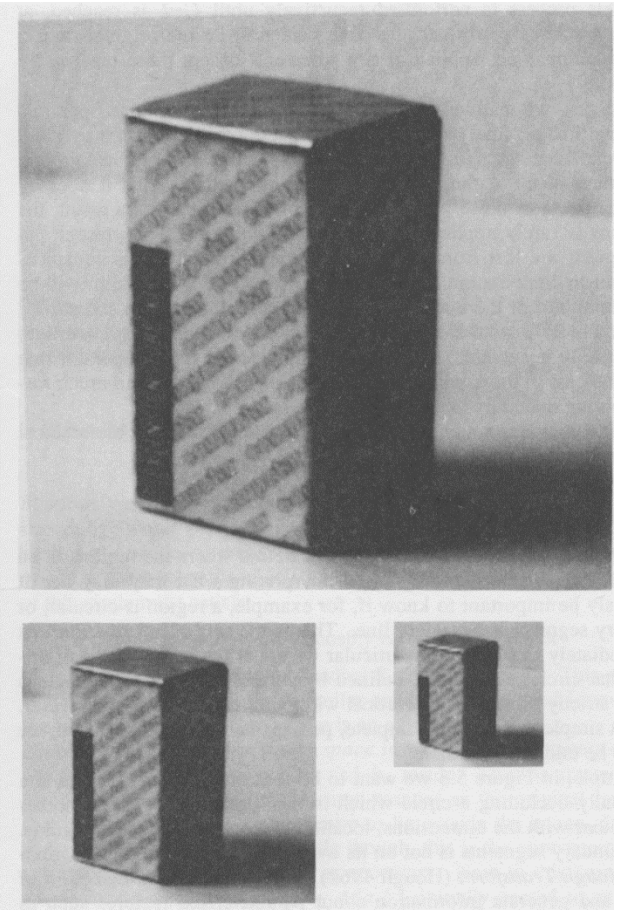
Relaxation Approach 1

- Perform boundary tracking (e.g. Canny alg.)
- Apply relaxation when the edge being tracked comes to an end, that is when the edge gradient becomes too low or the edge direction becomes too different to those at the preceding pixel(s).
- Extrapolate the end of the line being traced for one or more pixels in the same direction as the most recent pixel on the line, to determine whether there is evidence for continuation of the contour
- If there *is* evidence, then fill in the intervening pixel(s) which caused the gap, and continue the boundary tracking process
- **Note:** In order to consider pixels at further and further distances from the original gap pixel, larger and larger neighbourhoods have to be considered, due to the inaccuracy of gradient direction information. This effectively limits the distance of gaps that can be filled in this way, to a few pixels.

Relaxation Approach 2

- Apply a compass edge operator, producing an edge map which can be considered as a map of the 'strength of evidence' that each pixel is an edge pixel
- Allow pixels to influence their neighbours' strengths and directions:
    - Strong neighbouring pixels with similar edge directions will strengthen each other
    - Weak neighbouring pixels or those with contradictory edge directions will weaken each other
    - Apply iteratively until the edge evidence at each pixel is either very strong (255) or very weak (0)

[Aside] Image Pyramids



- Many image processing techniques, including edge detection, can make use of 'image pyramids' (hierarchies of increasingly high resolution images)
- The information available at different resolutions is different and often complementary (high resolution: more fine detail but also more noise; low resolution: larger structural detail becomes amenable to low-level image processing operations)
- Typically the image processing task starts with the lowest resolution image, providing useful information allowing a more directed effort on the image at the next resolution in the pyramid.
- It is believed that the human visual system operates in this way
- This is a ‘coarse-to-fine’ approach

e.g. Edge Detection with Image Pyramids

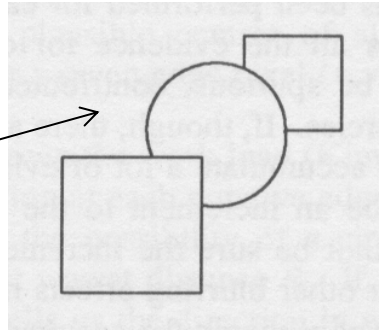
- Perform edge detection on the lowest resolution (i.e. most blurred) image: this will tend to find the major features and be insensitive to noise
- At each pixel for which this produces a strong edge, move to the next image on the hierarchy and apply the edge operator to the 4 relevant pixels
- The overall result is both faster and less affected by noise

The need for Template Matching

- Template matching leverages higher-level domain-specific information to assist the task of segmentation:
  - which edge pixels are likely to be due to the desired objects (do they form shapes that adhere to a template version of what we're looking for?)
  - which edges are likely to be due to **noise** and **clutter?**
  - where is it likely that **occlusion** has stopped edges from being detected at all?

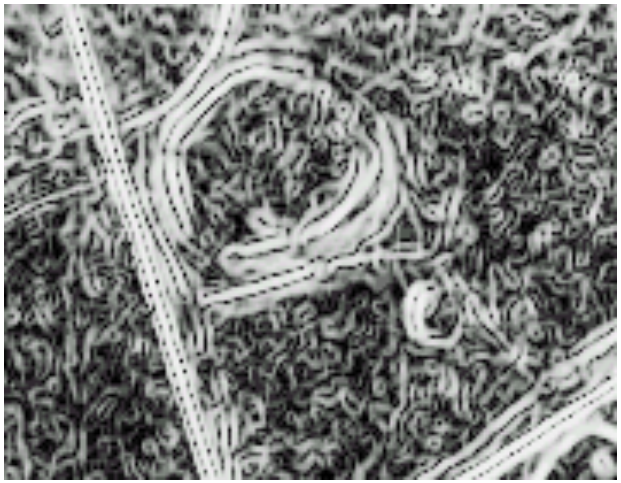
noise and clutter →

occlusion →



The need for Template Matching

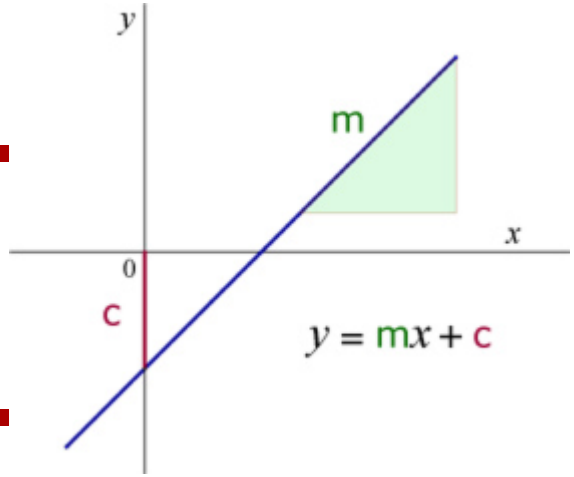
- E.g. automatic segmentation of sub-circular objects (ringforts)
- Laplacian-of-Gaussian
- Manual thresholding of LoG image
- Canny (lower=50,upper=200,sobelsize=3)



The Hough Transform

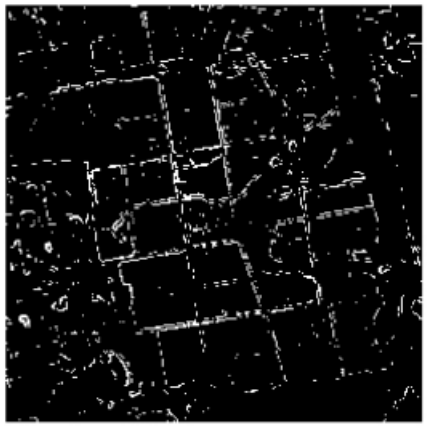
- A template approach that detects simple geometric shapes in edge-enhanced (typically binary/thresholded) images
- More complex shapes may be characterised as specific combinations of simpler shapes (necessary since storage and computation requirements grow alarmingly with increased feature space dimensions).. E.g. combinations of circles and lines in particular patterns/orientations
- Approach:
  - transform points in the spatial feature space into a parameter space, with 1 dimension of parameter space used to represent 1 parameter: e.g. a circle has two parameters: centre point and radius
  - search for peaks in the parameter space
- The HT approach is particularly **insensitive to noise and breaks in an object's boundary**, though it can be very computationally expensive (i.e. brute force approach)

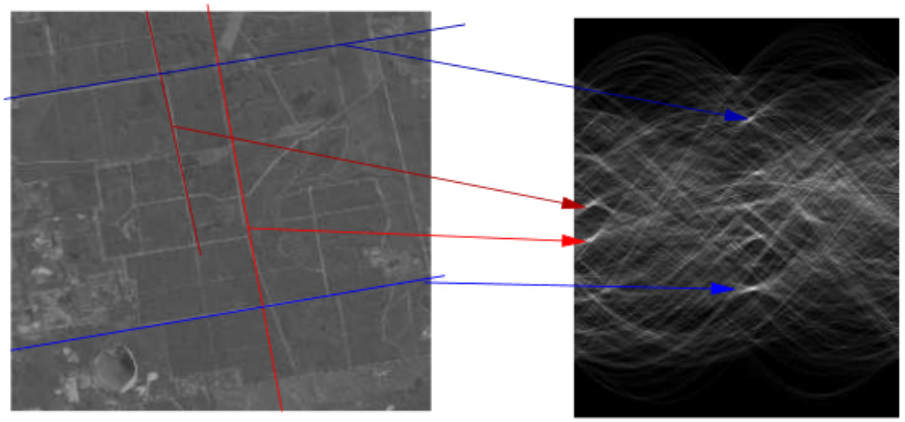
Hough Transform for Lines



- 'Votes' are gathered in an accumulator array, with one array dimension required for each parameter of the shape being sought
- For a straight line, 2 dimensions are required: slope (m) and y-axis intercept (c) : (y=**m**x+**c**)
- Each edge pixel casts a single 'vote' for each of the lines that it *may be on* (given a reasonable number of slope values, e.g. 360 different slopes)
- Peaks in the accumulator array are sought, and the lines with the most 'votes' are thereby identified and extracted, while broken edges and 'noise' edges will have relatively little effect on the result.

Hough Transform Example: finding roads in aerial image



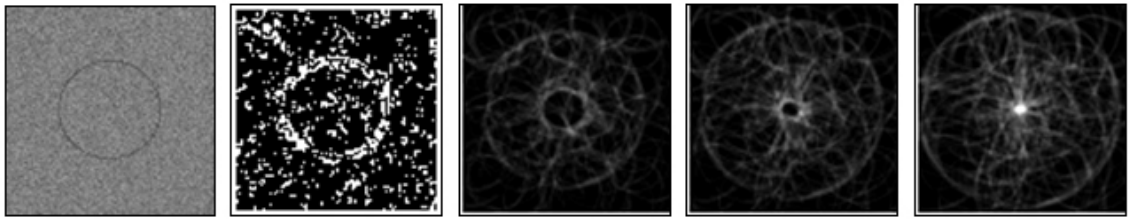


- Sobel filter
- Threshold
- Yields broken lines

- Hough transform 2D accumulator array shown with *c* on vertical axis and *m* on horizontal axis
- Highest peaks are identified

Hough Transform: Circles

- A 3-dimensional accumulator array is required
- The dimensions represent *x* location of centre, *y* location of centre, and *radius*.

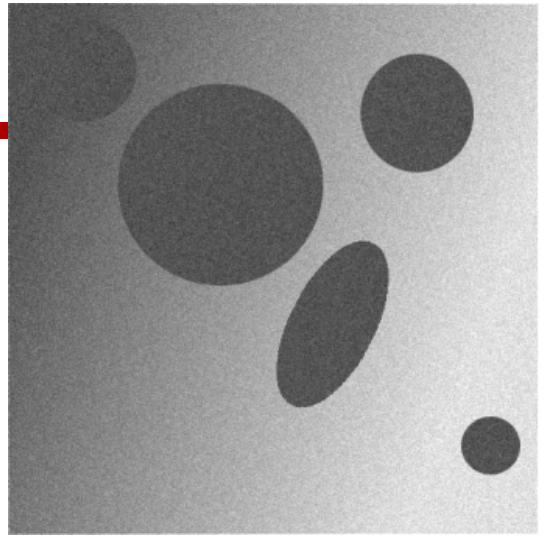


1 2 3 4 5

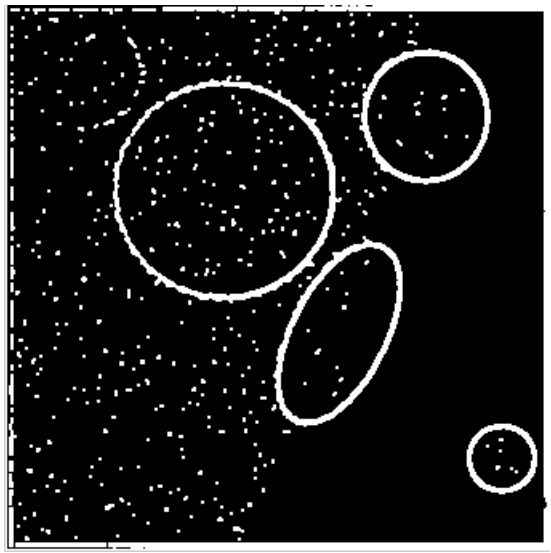
The images show the application of the circle HT to a noisy image containing a low-contrast circle of radius 24 pixels:
(1) the original image,
(2) the edge map used by the HT (following smoothing, edge detection, thresholding)
(3) contents of accumulator array at radius 15
(4) accumulator array at radius 20
(5) Despite the presence of noise, and fragmentation in the object's detected boundary, the accumulator array shows a strong peak at the centre of the circle at radius 24.

"Apples & pears.."

- Task: identify the 'apples' and 'pears' in a somewhat noisy image
- Original image

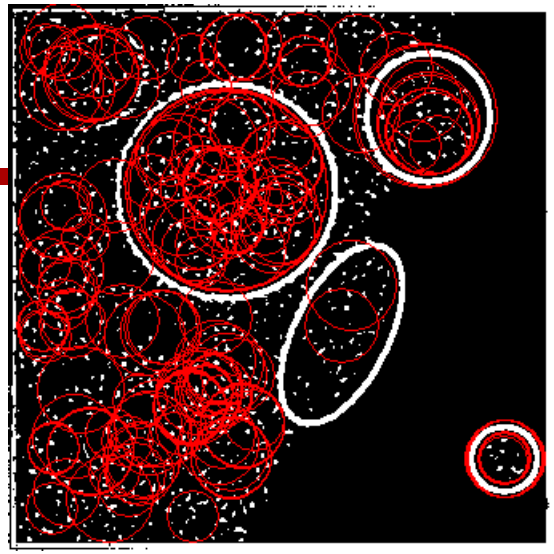


- Gaussian smooth, Laplace 7x7 edge detect, threshold at 150, Opening with 2x2 structuring element
- The 'apple' at the top left is going to cause problems…

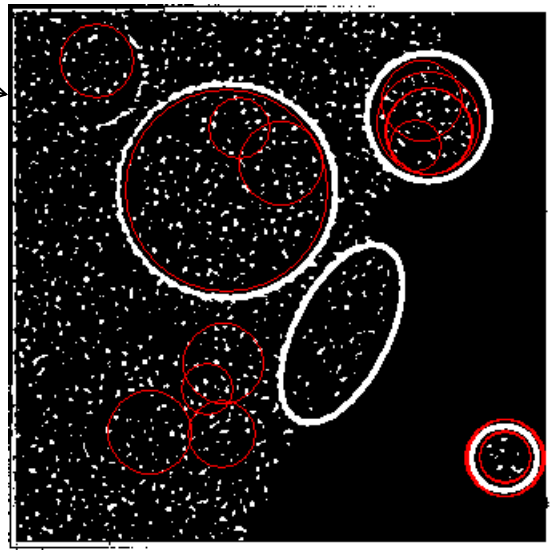


"Apples & pears.."

- Application of the HT for circles
- Accept all peaks in the accumulator where magnitude > 0.5*highestPeak*$2\prod r$



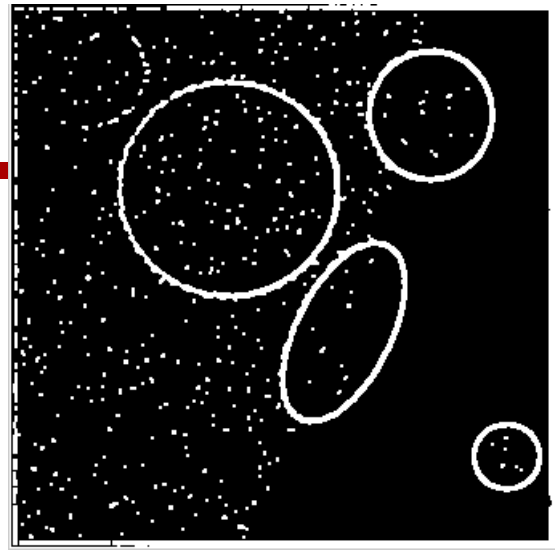
- Accept all peaks in the accumulator where magnitude > 0.7*highestPeak*$2\prod r$



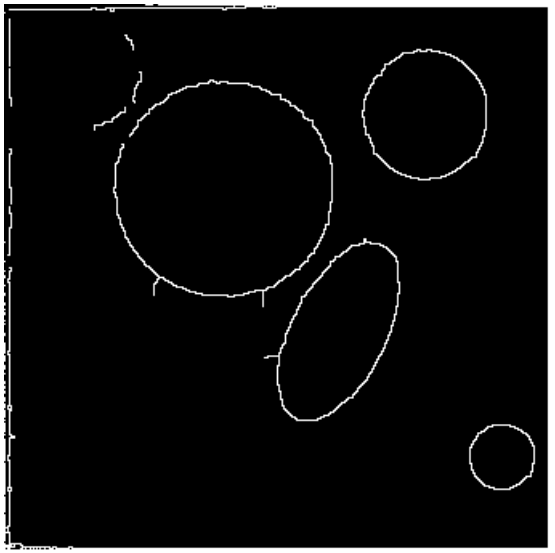
- *We need a morphological clean-up step that's more effective than opening.. should be strict on noise reduction while preserving connectivity where appropriate (see below: 'flood reject')*

"Apples & pears.."

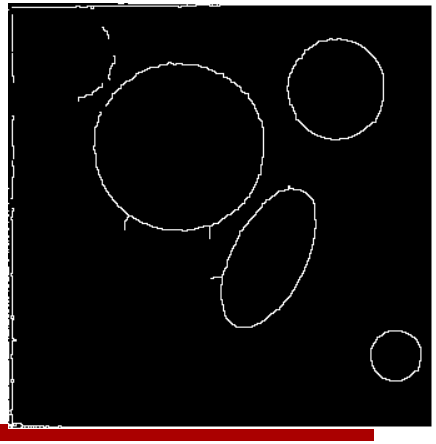
- Gaussian smooth, Laplace 7x7 edge detect, threshold at 150, Opening with 2x2 structuring element



- Gaussian smooth, Laplace 7x7, threshold at 127, skeletonise/thin, prune spurs at length 8, **flood reject** any connected regions less than 10 pixels in area (see next slide)



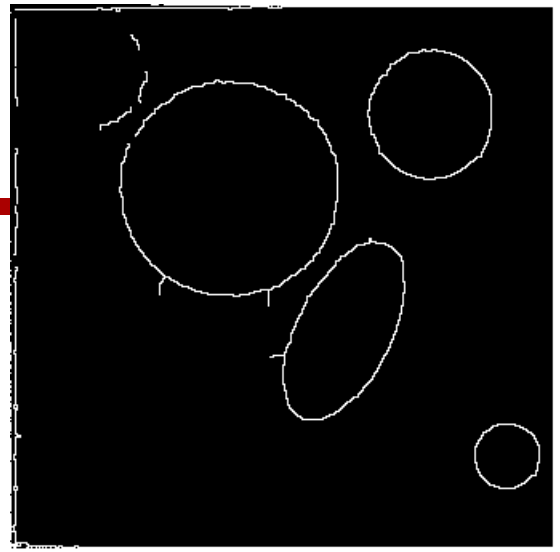
"Flood Reject" morphological algorithm (pseudo code)



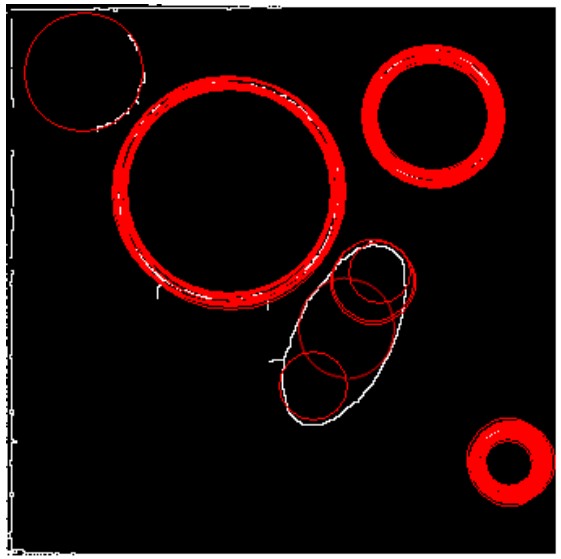
```
/* Determine the size of the patch that pixel at (x,y) belongs to */
Procedure Fix(x,y)  /* (x,y) is an edge pixel */
  Call FloodFix(x,y)
  If a value of less than the predetermined threshold patch size was
returned, then reject all pixels stored during the recursive FloodFix
calls.
End Procedure

Procedure FloodFix(x,y)
  Set the value at (x,y) to some temporary value, so that it is not
repeatedly visited
  Check all neighbours (a,b) of (x,y), i.e. in the range (x-1,y-1) to
(x+1,y+1)
    If any (a,b) are edge pixels then recursively call FloodFix(a,b)
  Increment a counter of overall patch size
  Store (x,y) if the patch size so far is less than the threshold for
a "good" patch.
Return the patch size
End Procedure
```

"Apples & pears.."

- Edge map used by the Hough Transform

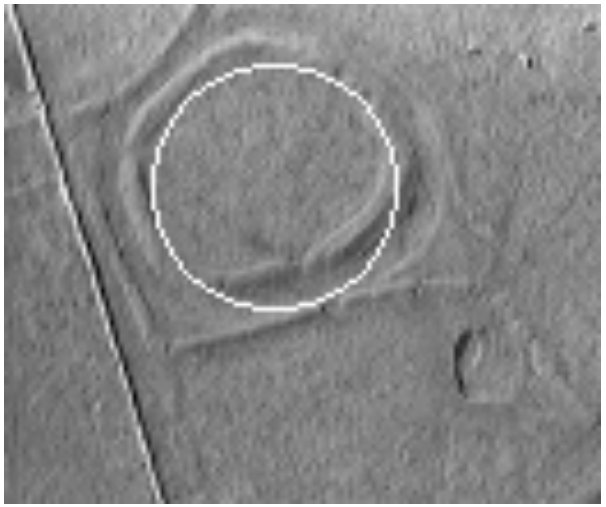


- Accept all peaks in the accumulator where magnitude > 0.4*highestPeak*$2\prod r$
- *Interpret specific combination/pattern of circles as 'pear'*
  - *Close/overlapping circles*
  - *Circle centres linearly aligned*
  - *Tendency to be larger radii towards centre of group*



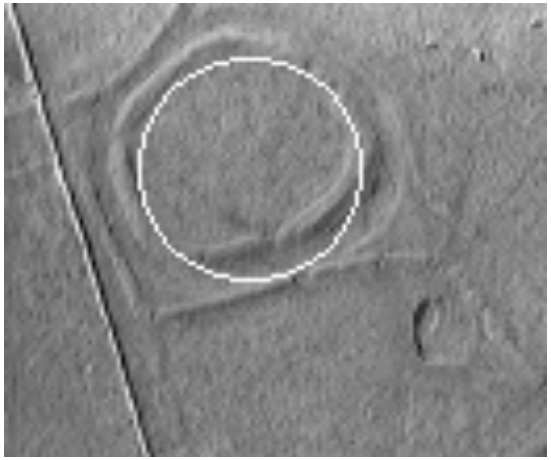
Recall: low contrast shape in the presence of clutter and occlusion

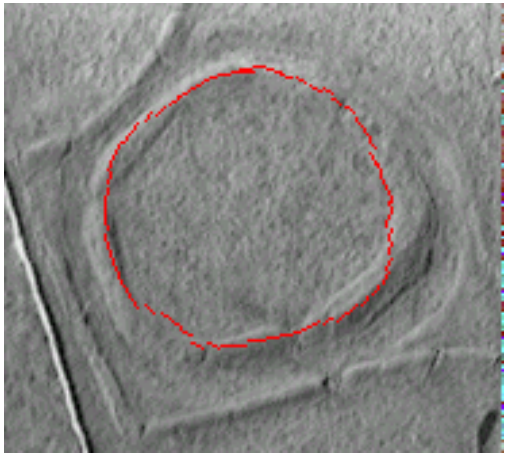
LoG, Thresholding



HT (Circles)

Archaeological Monument Example





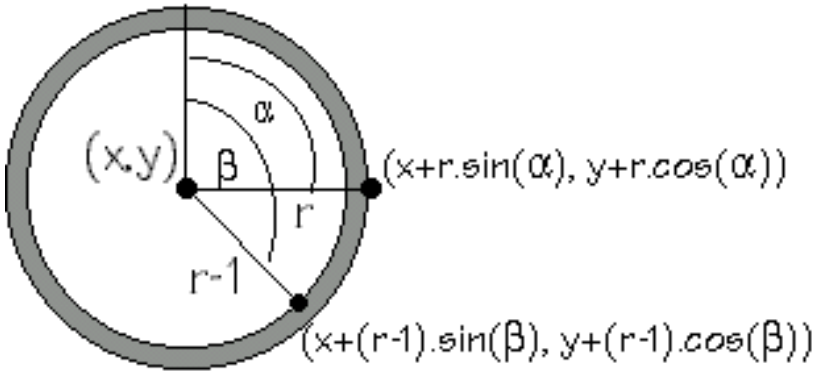
- Hough Transform as first step in a multi-step coarse-to-fine algorithm
- HT used as robust but approximate segmentation tool
- HT seeds the region for a detailed, problem-specific approach:
- Sub-pixel accuracy directional edge detection and the use of domain-specific object primitives (circular arcs)
- Highly directed search for edges allows us to use very sensitive edge detection operator without getting swamped by noise and high-contrast clutter

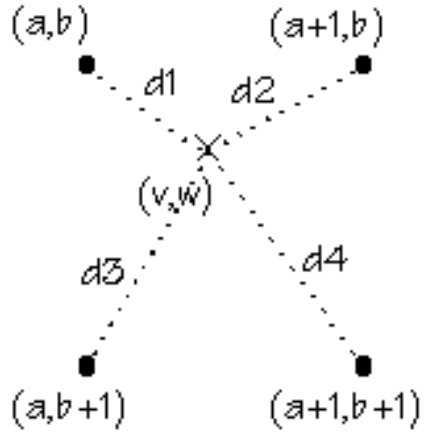
Sub-pixel directional edge detection for circular arcs

- The 1st derivative (edge strength) at a point on a circle's circumference can be estimated by:

$$\Delta I = abs(I(x+r.sin(\theta),y+r.cos(\theta)) - I(x+(r-1).sin(\theta),y+(r-1).cos(\theta)))$$

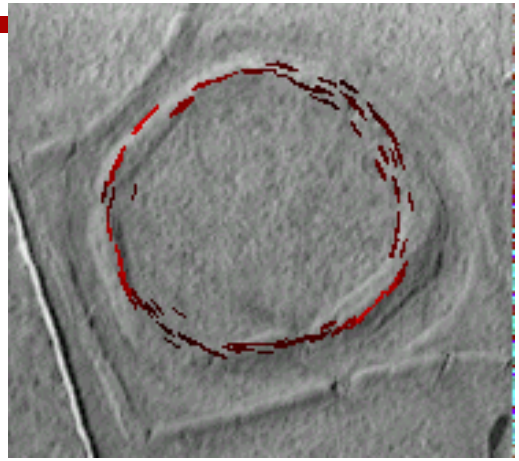
- Where *I(x,y)* is the intensity of the pixel at location *(x,y)*
- Since points on the circle's circumference will not fall exactly upon the pixel grid positions, it is necessary to estimate the intensity *I* at a point *(v,w)* through interpolation.

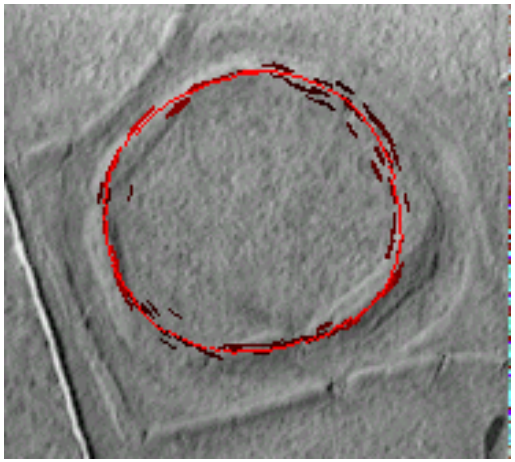




Archaeological Monument Example

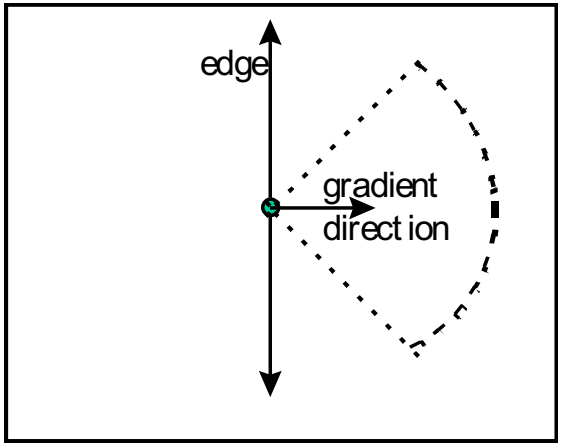
- HT seeds circular arc detection, and provides range of candidate centre points and radii
- For each combination of centre point and radius, 50 discrete 7.2 degree arcs are considered
- Edge strength of each arc is the sum of 12 calculations (cf previous slide) at evenly spaced points on its circumference
- Strongest 8 results at each of the 50 arc positions are accepted, and used to calculate weighted moving average of centre point and radius for final shape
- Edge strength provides the weighting (brighter red arcs in diagram)



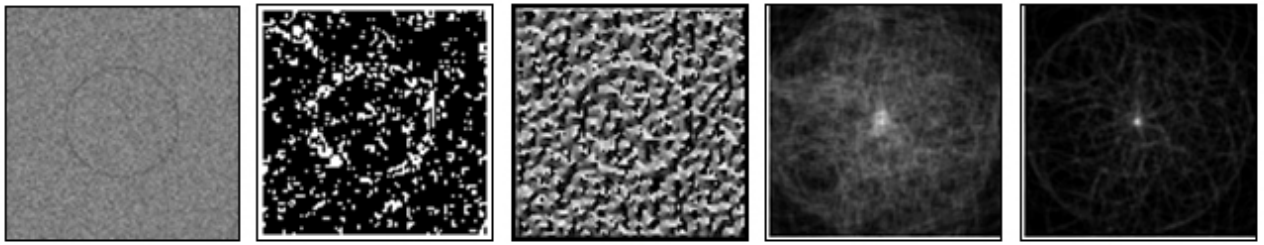


Fast Hough Transforms

- One approach is to use gradient direction information in order to minimise the number of 'votes' that have to be cast by each edge pixel
- E.g. an edge pixel lying on an (estimated) vertical boundary: its gradient direction is perpendicular to this boundary.
- Rather than casting votes for circle centres at each accumulator array location at a given distance from it, the edge pixel needs to cast only 25% as many votes, by directing these votes across a 90^o arc cantered on the estimated gradient direction
- smaller angles are possible if the confidence in edge directions is strong
- Another common approach for FHTs is to allow only a random subset of the edge pixels to cast votes



Gradient-direction assisted FHT

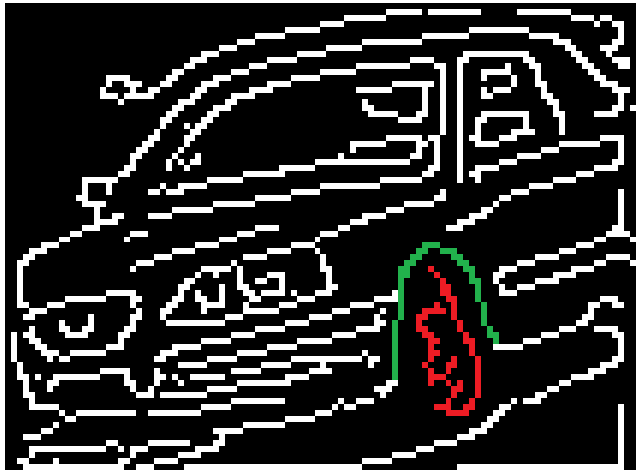


(1) A low contrast circle of radius 24 pixels, in a noisy image
(2) The edge map derived from a compass edge detector's gradient magnitude response
(3) An interpretation of the gradient direction map from the compass edge detector
(4) The accumulator at radius 24 pixels, as calculated by the standard circle HT
(5) The accumulator at radius 24 pixels, as calculated by the FHT under investigation (casting votes in 90 degree arcs, as above)

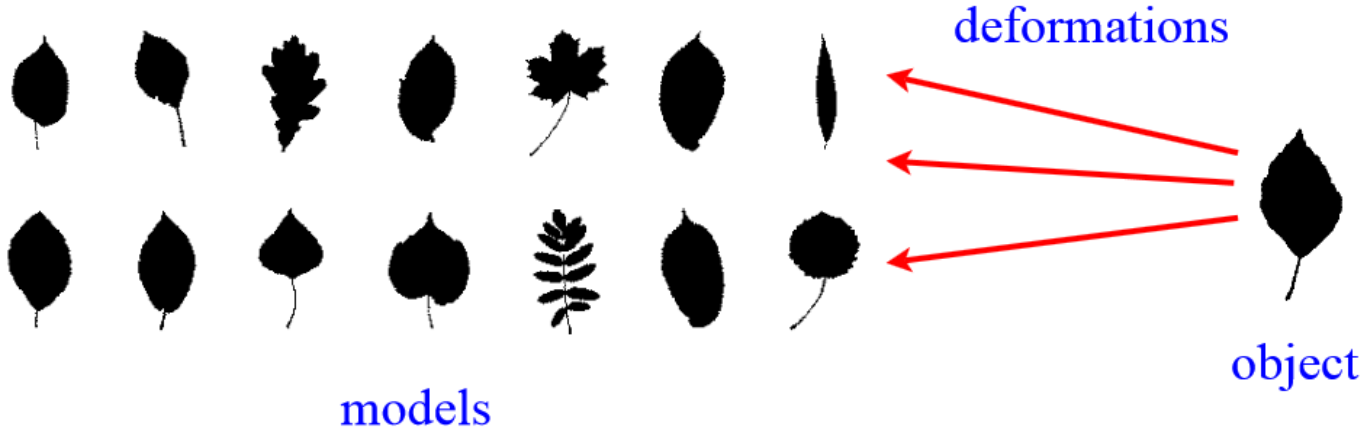
Vehicle Vision System

Circle Hough Transform for wheel detection

Wheel-arch detection based on Canny algorithm and knowledge-driven search for arcs
(Valeo)

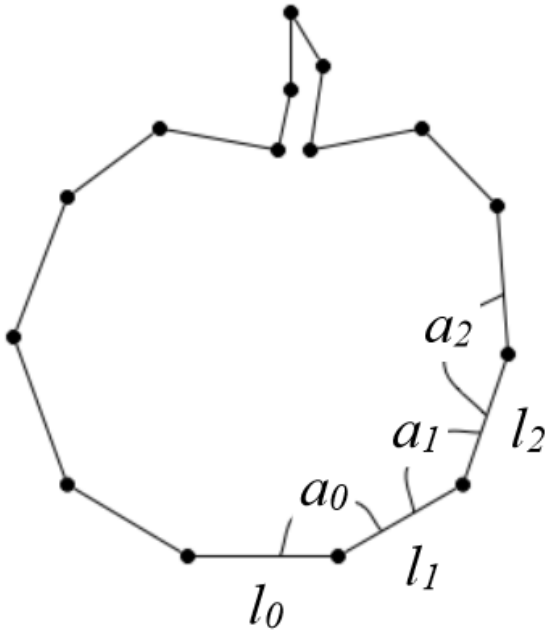


Deformable Template Matching



- Normally used as part of a classification system (see next section of course)
- Find the minimum deformation necessary to transform object into one of the stored template models
- How to encode boundary (+interior?) in such a way that deformation ‘cost’ is quantifiable?

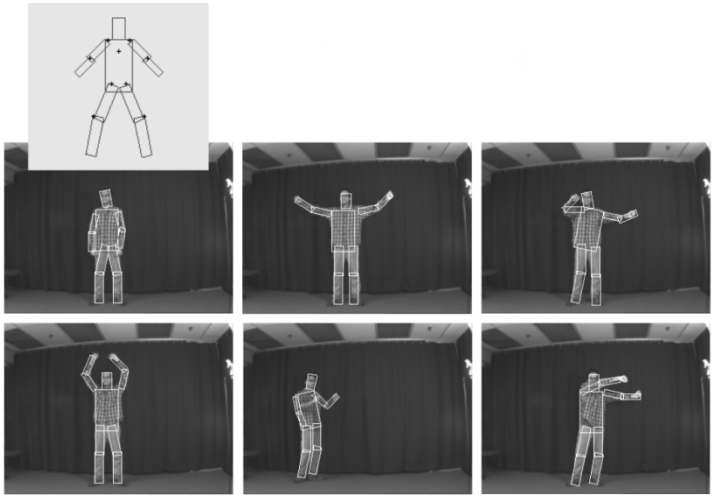
Deformable Boundaries



- E.g. polygonal boundary model
- Consider deformations which preserve local properties such as angles and ratios of lengths
- Measure ‘cost’ of deformation in terms of deviations from template angles/lengths

Part-Based Models

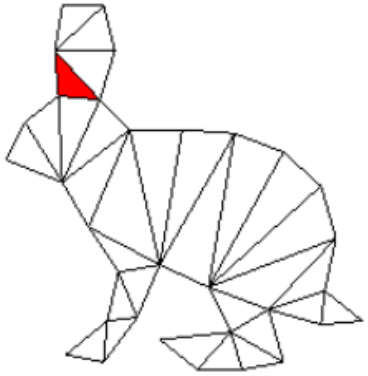
- Constellation of parts
- Rigid parts are arranged in a deformable configuration
- Deform distances, angles within statistically valid ranges



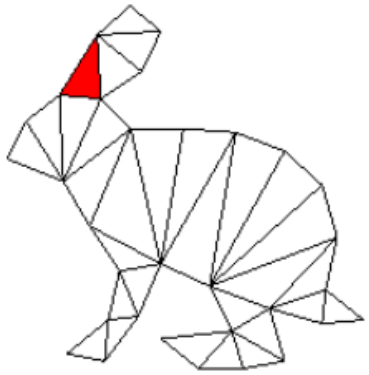
template

Triangulation Approach

- Decompose polygon into triangular parts
- Measure deformation of shape as sum of deformation costs per triangle
- Valid range of deformations for each triangle may be learned via training examples



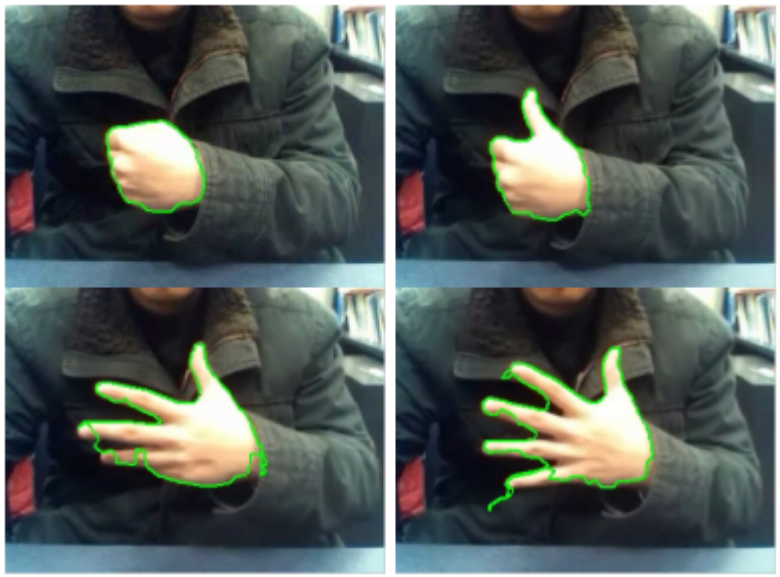
template



deformation

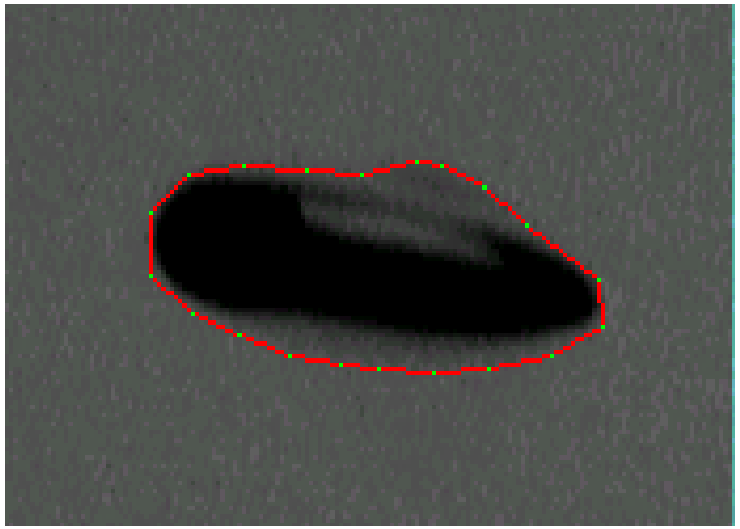
Active Contours ('snakes')

- Useful for objects from an image where approximate location is known
- May be used as part of a coarse-to-fine approach, e.g. seed active contour using Hough Transform or some other robust yet imprecise approach
- Useful for image sequences, since the contour data between neighbouring images will be closely matched



Active Contours

- Active contours conform to features within an image according to pre-defined weights and an energy minimisation approach
- Contour is composed out of a number of ‘snaxels’ (essentially, vertices) and the optimisation algorithm seeks to minimise the total energy, i.e. sum of energy at each snaxel



- An excellent discussion of active contours and their energy factors (although using a different image processing library):

http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/YOUNG/vision7.html

- Also:

http://homepages.inf.ed.ac.uk/cgi/rbf/CVONLINE/entries.pl?TAG709

Energy Factors (Constraints)

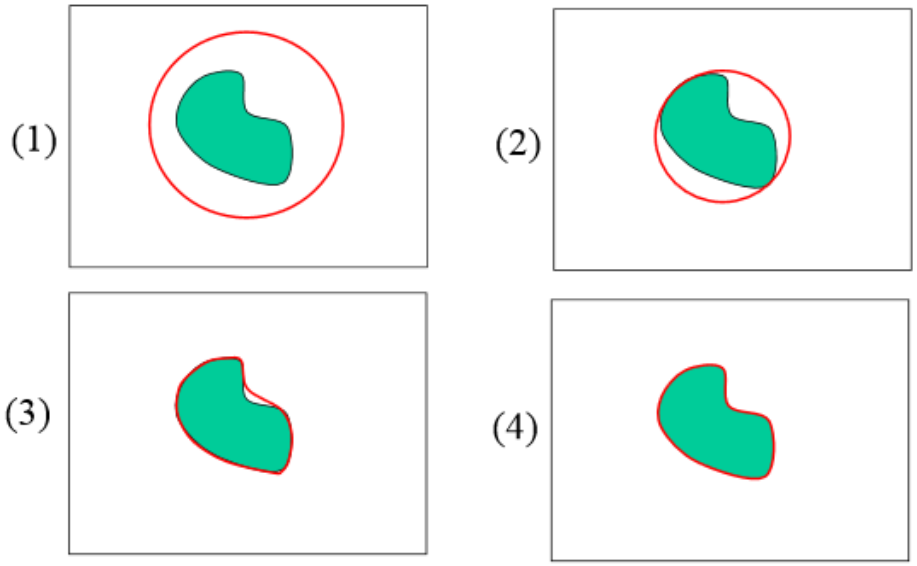
- Optimise a set of internal and external energy factors
- Internal energy factors constrains the shape of the curve
    - E.g. to encourage smoothness (few sharp angles)
- External factors encourage conformance to features in the image itself:
    - E.g. strong edges
- Total energy for the snake with snaxels at a particular position is the sum of all of their internal and external energy factors

Example Energy Factors

Example energy constraints:

- Move towards centroid of object
- Seek out strong edge pixels
- Rigidity (angular continuity) - a snaxel B is penalised if the angle ABC is large (where A and B are its neighbours)
- Distance continuity - a snaxel is penalised if the distance between it and its neighbours is different to the average

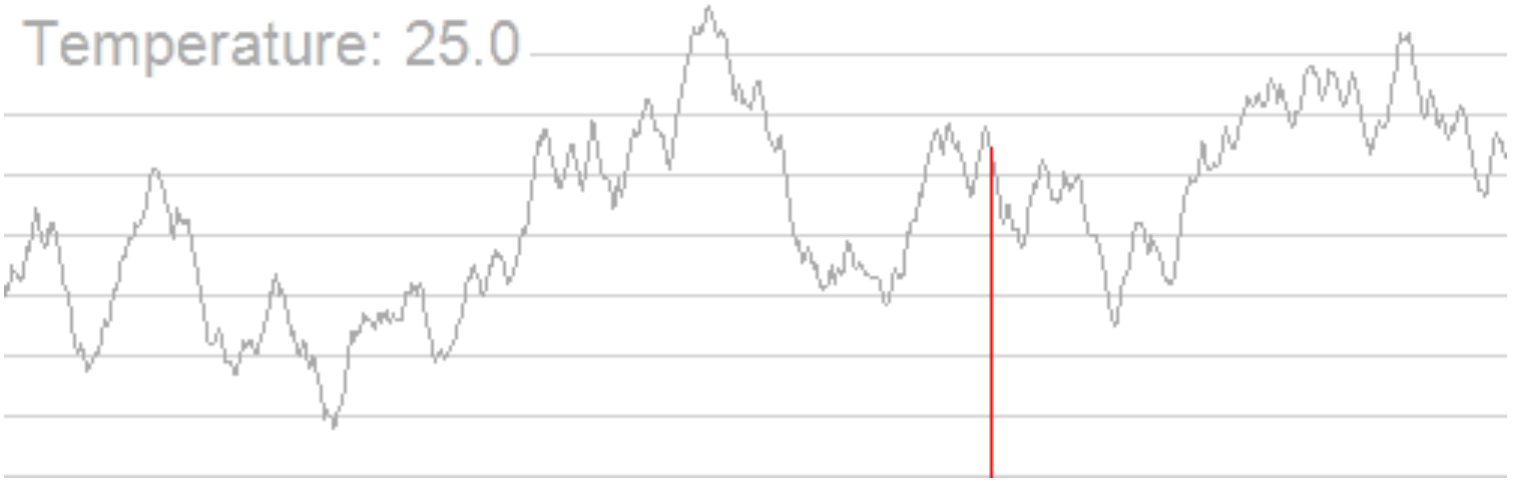
Energy Optimisation



- Optimisation refers to iteratively modifying the snaxels in order to find an optimal set of energy factors
- Also referred to as ‘search’ since it is a search of a problem space

Optimisation

- Local optimisation techniques consider localities of each snaxel as candidates for their new position; e.g. ‘hill climbing’ (consider the energy surface like a hilly terrain)
- Global optimisation techniques may consider positions far away => less likely to get caught in local minima and therefore to find the global optimum values; e.g. simulated annealing (from metallurgy), genetic algorithms (biologically inspired)

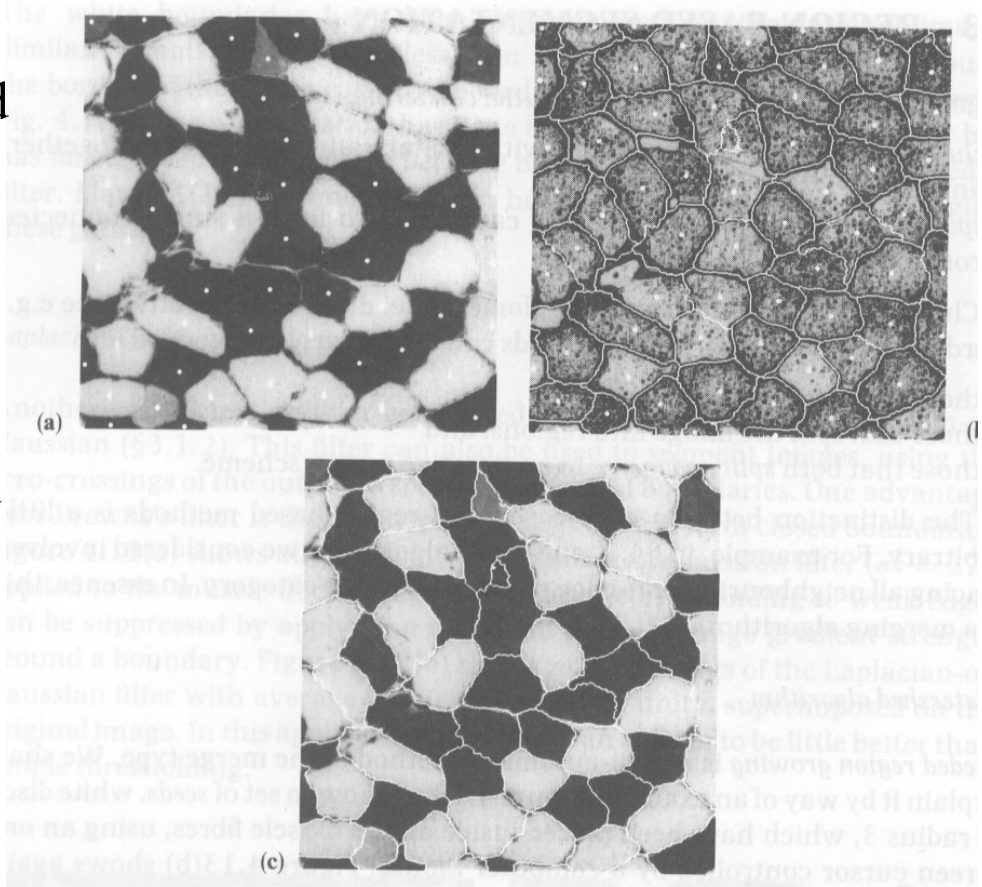


Region Growing

- Another important category of segmentation techniques
- Typically based on spectral information rather than (or as well as) spatial/morphological information.. i.e.:
    - Spectral: what are the actual values in the pixels
    - Spatial: what shapes do strong edge pixels form
- Step 1: divide image into many small regions (even single pixels)
- Step 2: compute properties that define region membership, e.g. average grey level, texture, colour
- Step 3: examine adjacent regions: if their properties are sufficiently similar then merge regions together
- Step 4: return to step 2 and continue until no regions were merged in step 3

E.g. (Muscle Fibres)

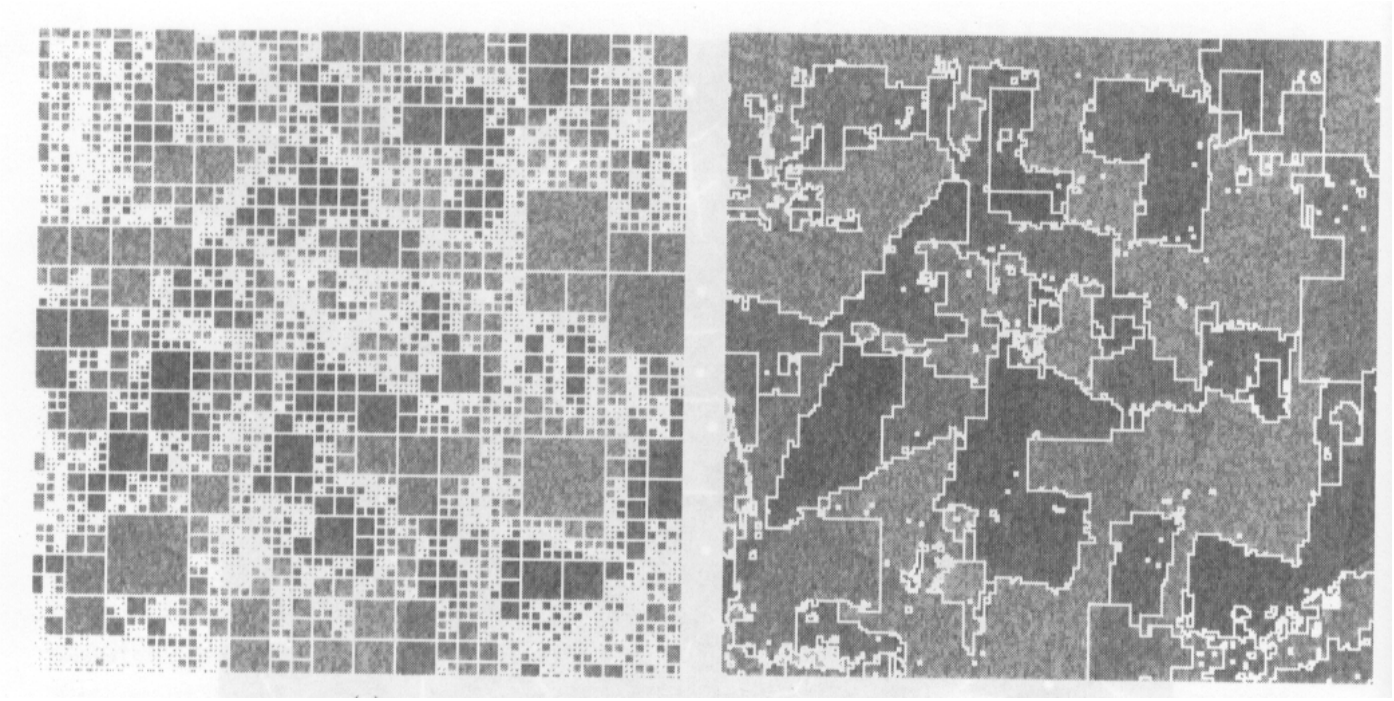
- Seed points for each region could be manually defined, or automated via threshold + distance transform
- (In this approach, most pixels do not yet belong to any region)
- Examine pixels adjacent to regions, and if edge strength is sufficiently low then add to region
- To ensure all pixels belong to a region, use a watershed approach by raising the edge strength threshold periodically



The Split+Merge Algorithm

- The ‘split’ process functions as an inverse of region growing:
  - Start with entire image belonging to one ‘region’
  - Examine properties of region, and if variance (e.g. standard deviation) is high then split region into 4
  - Use recursion to examine each newly formed (smaller) region and split as necessary
- When no more splits are needed, the merge process starts (merge = region growing)
- Initial image should be square, and sized to an exact power of 2 on each side for the above to work

Example



(Left): Split – regions with colour variance of greater than 0.6 are split.. notice how the regions following the split are smaller in non-uniform parts of the image.

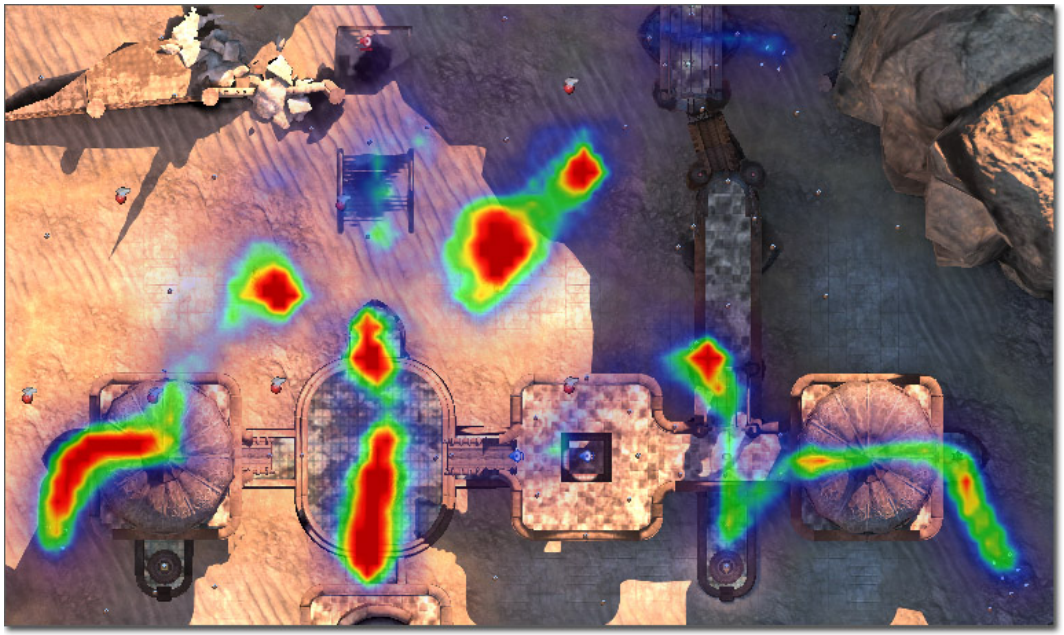
(Right): Merge – regions with colour variance of less than 0.6 are merged

Video: Background Modelling

- Image processing of video sequences is often concerned with identifying foreground/background pixels based on movement
- The rationale in the approach is that of detecting the moving objects from the difference between the current frame and a reference frame, often called "background image", or "background model"
- Frame Differencing/Background Subtraction - which pixels in the current image vary substantially from the same pixels in the background image? These are assumed to be foreground.
- How to generate a background image:
    - Take a picture where foreground is known to be empty!
    - Take the mean/median/mode of each pixel
- What if the background is not truly static? (e.g. tree leaves)
    - Statistical models can help - e.g. identify peaks in histogram of values for each pixel when considered across the whole sequence. Values close to these peaks are likely to be background.

Heatmaps

- Accumulating frame differences over a sequence of frames (e.g 50 frames) can yield a heatmap - can be useful for tracking movement (and therefore foreground objects) in the absence of a stable background model
- See 'frame differencing tests': code on blackboard (but not the images which are 90MB) - also next slide



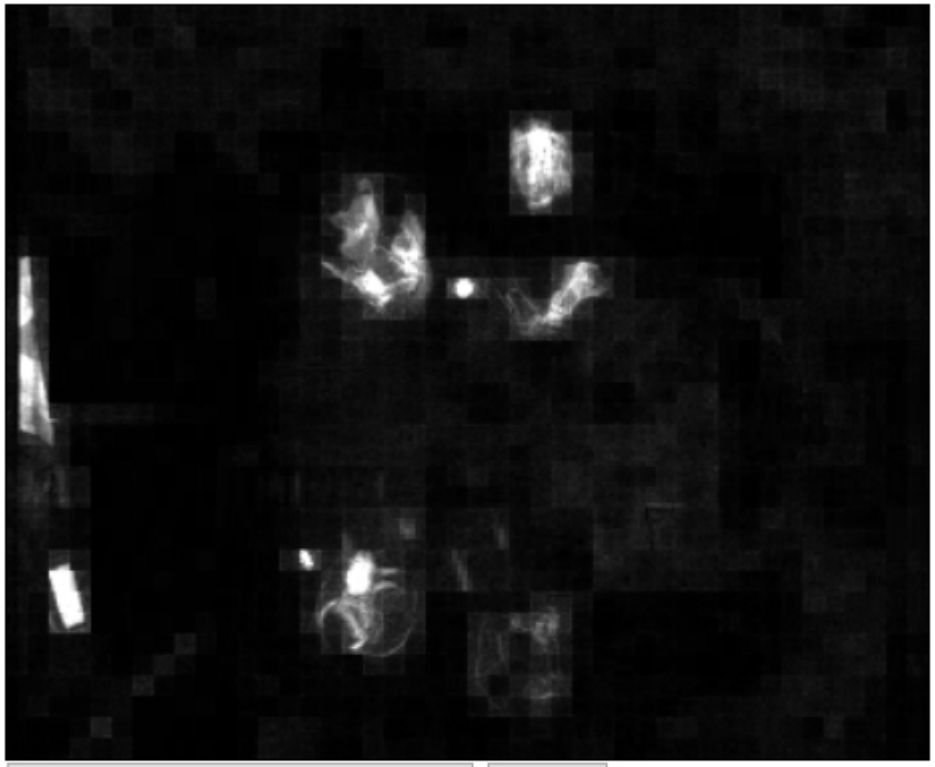
Cycle Smart-Meeting Images | Stop | 9

Heatmap Test 1

Heatmap Test 2

Background Model - Mean

Background Model - StDev



Cycle Smart-Meeting Images | Stop | 9

Heatmap Test 1

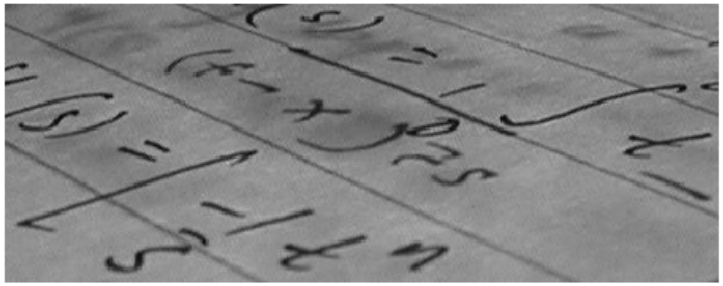
Heatmap Test 2 | 24

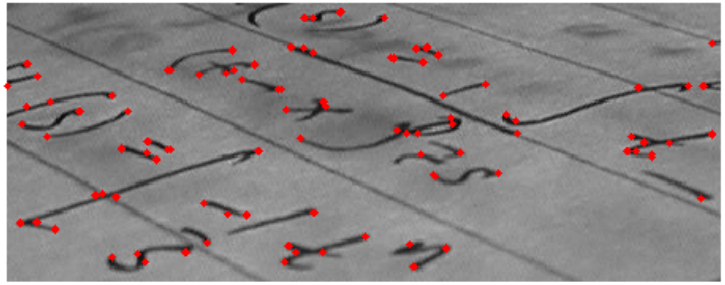
Background Model - Mean

Background Model - StDev

Feature Detection

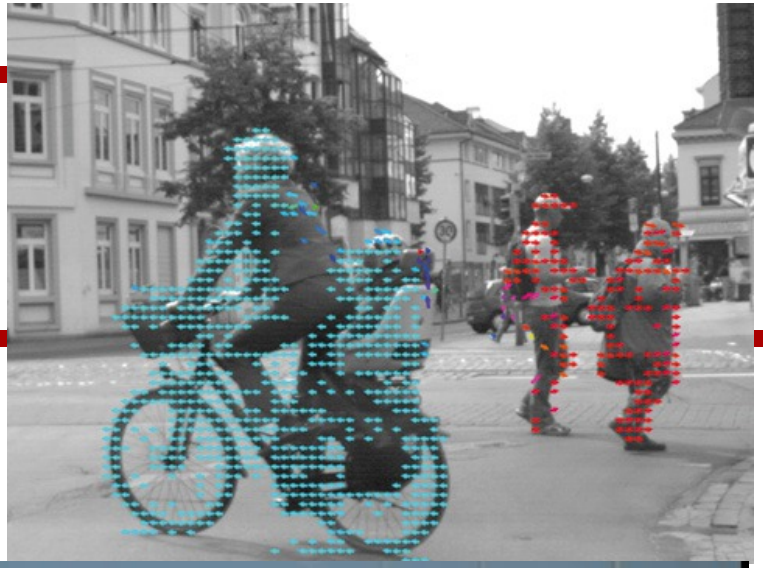
- Feature detection is an approach used within computer vision systems to extract certain kinds of features and infer the contents of an image.
- Frequently used in motion detection, image registration, video tracking, image mocaicing, 3D modelling and object recognition.
- Used for automated 'interest point' detection
- E.g. Harris Corners, Scale-invariant feature transform (SIFT),
- Harris Corner demo:





https://inspirit.github.io/jsfeat/sample_fast_corners.html

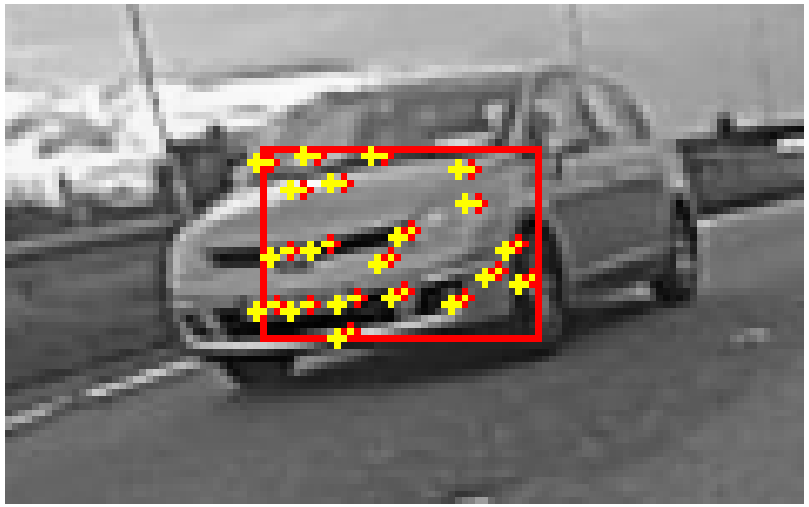
Optical Flow

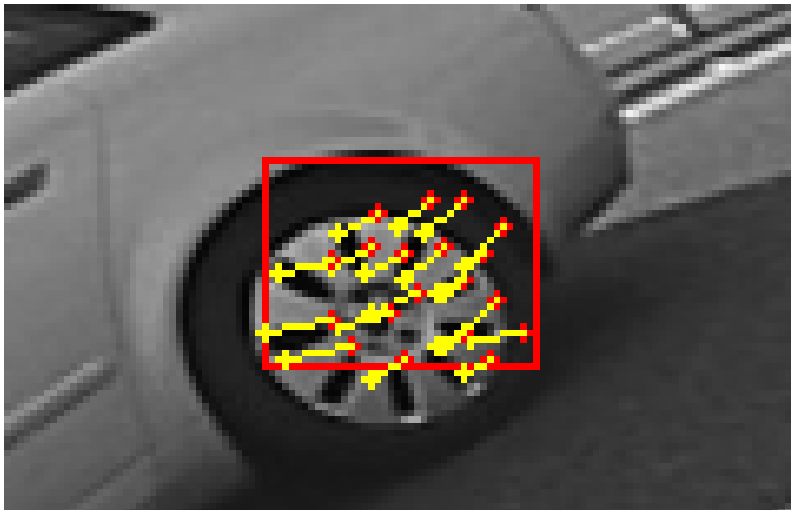


- Optical flow is the pattern of apparent motion of image objects (or pixels) between two consecutive frames caused by the movement of object or camera.
- It is 2D vector field where each vector is a displacement showing the movement of points from first frame to second

See demo of Lucas-Kanade alg.: **https://inspirit.github.io/jsfeat/sample_oflow_lk.html** (also uses image pyramids)

Vehicle Vision System





Optical flow of Harris Corners tracks vehicle movement and identifies wheels by characteristic rotational motion