Compare & Contrast

Saturday, October 25, 2025 3:10 PM

2024
Compare and contrast:
RANSAC to find lines.
Hough transform for lines.
Contour following followed by recursive boundary splitting.

You must provide a list of the differences and similarities between the techniques. Each of the differences and similarities must be clearly explained. NOTE: Marks will only be awarded for the detailed comparison of techniques. No marks will be awarded for separate descriptions of the techniques

Purpose:
All of these techniques are used to detect or describe straight lines in an image.
RANSAC fits a line model to scattered datapoints, allowing to make these detections in noisy data.
The Hough Transform votes for line points in the parameter space.
Contour following obtains a polygon approximation of an object and recursive boundary splitting takes the boundary and segments it into a number of straight lines to build up the boundary.
Preprocessing:
All of these techniques require preprocessed inputs and will not work on just a raw image.
None of the techniques operate at a pixel-level.
RANSAC requires a set of feature points e.g. Corner detection or SIFT and some model e.g. Line of best fit which to map those feature points to.
Hough requires a set of edge point se.g. From a Canny Edge Detection and a parameter space e.g. A line which to map those edge points to.
Contour following requires a binary image and a measure of connectivity (8-adjacency, 4-adjacency) to define the connectedness. Only then can recursive boundary splitting be used.
Inputs:
RANSAC uses feature points.
Hough uses edge points.
Contour Following uses a binary image.
Recursive Boundary is based on objects/outlines from the binary image, instead of a collection of feature/edge points.
Additional Parameters:
RANSAC needs a model to fit to and also requires a distance threshold to determine how far away a feature point is allowed to be before it should be considered as an outlier i.e. Not in the line we are looking for.
Similarly, recursive boundary splitting also requires a threshold for the deviaton. How much curvature should it observe before splitting into another set of straight lines.
The Hough Transform requires the parameter space which it should map its point sto.
Contour Following requires an adjacency metric which it should use to decide how connected the boundaries of an object are.
Outputs:
RANSAC outputs the model *parameters* to the best fit model. This is a non-deterministic output due to RANSAC's randomness and may produce a slightly different output with the exact same data, depending on what starting points were chosen.
Hough outputs the most likely line(s) as Hough has the capacoity to detect multiple lines at once.
Contour Following & Recursive Boundary Splitting produce a traced boundary around an object whose location is already known. But unlike RANSAC which is robust to noise and Hough which is moderately capable of dealing with noise, recursive boundary splitting will fail ig the contours are noisy or fragmented.
Method:

RANSAC uses randomness. Selecting two points randomly and evaluating against the model to fit. Non-deterministic. Robust to noise. Selects more subsets of the data until convergence.
Hough accumulates votes in the straight line parameter space. Deterministic. Can detect multiple lines at once.
Contour Following + Recursive Boundary Splitting is the only method which is looking for the "least" number of iterations/votes. The more times recursive boundary splitting splits, the more curvature the object has. A greater number of very small straight lines indicates that the boundary is very elliptical.

Compare and contrast:
Robust Object Recognition with a cascade of classifiers.
Scale Invariant Feature Transform.
Principal Components Analysis.

You must provide a list of the differences and similarities between the techniques. Each of the differences and similarities must be clearly explained. NOTE: Marks will only be awarded for the detailed comparison of techniques. No marks will be awarded for separate descriptions of the techniques

Purpose:
Each of these technqiues is used to recognise features/objects in an image.
Cascades recognise directly by iteratively comparing the image to known strong classifiers.
SIFT defines keypoints and their orientations in one image such that they can be found in subsequent images.
PCA reduces the dimensions of features to give a more generic medium for recognition.
Simplificaton/Abstraction:
PCA described above simplifies by creating mean centred data for a feature e.g. A face. PCA's eigenvectors/eigeanvalues describe the feature and on an unknown image, a match is made to the nearest neighbour.
SIFT simplifies by taking a complex image and defining key points by considering the image at different scales (Difference of Gaussian blurs). SIFT drops keypoints which are poor/unstable i.e. Near/on edges.
Cascades simplify by pruning detection. Only images that "pass" on one strong classifier are passed along to the next strong classifier.
Robustness & Outputs:
SIFT is inherently invarient to scale/rotation. It uses a 4D Hough Transform when making (two for location, one for orientation, one for scale). It can output a detected object at different angles without a preprocessed perspective transform. This is facilitated by SIFT needing to find at least three matching features to be considered a valid match.
Cascades are not invarient due to the classifiers (e.g Haar-like rectangles to detect a face). If a face is straight-on, its detection will be output because of the alignment with the rectangles. But a side view will not be detected/outputted as it does not align well with the classifiers. Output is usually bounded with a box.
PCA is not inherently invarient either, but can be made invarient to a degree if a large number of different orientations are used when defining the mean centred data. It produces feature-vectors with reduced dimensionality.
Methods:
Classifiers: train a recognition model using a number of positive and negative samples.
Use very simple features.
Some classifiers are strong and others are weak.
Pass the image along the strong classifiers, pruning as it is passed to quickly reject poor detections.
SIFT: Consider the image at different scales (Difference of Gaussians blur).
Candidate keypoints are identified based on what features stand out compares to their neighbours.
Weak keypoints are dropped from consideration e.g. Edges.

Orientations are assigned to each keypoint by analysing the brightness around the keypoint.
The keypoints are turned into a feature vector, describing the local pattern.
All of the summarised feature vectors are called keypoint descriptors.
This is what makes SIFT scale and rotation invarient.
Keypoint descriptors are compared in Image A and Image B, with poor matches being dropped.
SIFT recognition matches at least 3 features and uses a 4D Hough transform (2 location, 1 orientation, 1 scale).
PCA: Creates a data matrix for features, generates the mean centred data.
Computes the covarince matrix and solves for the eigenvalues and eigenvectors.
The normalised eigenvectors which describe the feature are compared with the unknown image, with a nearest enighbour matching being the recongition.
Inputs:
PCA treats each image as a vector.
SIFT and Cascades operate on greyscale images.
SIFT also has the capacity to operate on colour/RGB images.

Explain the following terms, the context in which they are used and their purpose:
Hidden layers.
Loss functions.
Region Based CNNs (R-CNN).
Recurrent Neural Networks (RNN).

For each term you must provide an explanation of the term (using diagrams if appropriate), the context within which it is used and also describe its purpose.
Hidden Layers:
The intermediate layers of neurons in a neural network. Between input and output.
They perform learned transformations that map input features to useful internal representation.
They are used in Deep Learning, CNN's, RNN's, classification tasks.
They introduce non-linear decision boundaries.
They allow the model to approximate complex functions.

Loss Functions:
Measures the difference beween a model's predictions and the ground truth e.g. Cross-entropy.
They are used when *training* neural networks. And evaluating model quality.
They are used in optimisation using gradient descent.
They guide weight updates in the network by providing a signal that tells the optimiser how wrong the model is.
It understands the objective of the model e.g. Maximimise classifications.

R-CNN:
Region-Based Connected Neural Networks detect objects by extracting regions likely to contain the required data.
They pass this data through a CNN and classify each region.
Therefore they are used in object detection, image segmentation, instance-level recognition.
It allows accurate detection of objects in complex scenes.

RNN:
Recurrent Neural Networks are networks with loops, allowing the information to persist across multiple steps where otherwise it would have been filtered out by the neural network.
The hidden state summarises the previous inputs.
Enables tasks like prediction, particularly where sequential, past data is required.

2023
Compare and contrast:
Canny edge detection.
Connected Components Analysis.
Mean Shift Segmentation.

You must provide a list of the differences and similarities between the techniques. Each of the differences and similarities must be clearly explained. NOTE: Marks will only be awarded for the detailed comparison of techniques. No marks will be awarded for separate descriptions of the techniques
All of these techniques are used for region analysis.
Canny edge detection takes in a greyscale image and outputs a binary map of edges in the image.
Connected components analysis takes in binary images and outputs a labelled image of regions where binary points are connected by at least a pixel.
Mean shift segmentation is able to take in a colour image and outputs a similar image where pixels of similar colours are merged into regions.

Canny detection is a series of steps which begins with a Gaussian smoothing/blur of the image to reduce noise. Mean shift segmentation also has the capacity to reduce noise as it merges similar colours together which can mitigate the effects of Gaussian or salt-and-pepper noise. CCA does not have the capacity to reduce noise as it connects regions based on pixels. If noise wants to be reduced in a binary image, a morphological operation such as opening or closing must first be used. Canny detection then does a gradient computation such as a Sobel edge detection. After non-maxima suppression and thresholding, this produces a map of the edges within the image. Mean shift segmentation also preserves strong edges in the image.

CCA is the simplest operation with the least computational intensity, mean shift segmentation works by shifting each pixel towards the densest region in its locality, using a kernel. Canny is the most computationally complex with several complex operations such as another edge detector and non-maxima suppression

---

Compare and contrast:
Support Vector Machines.
Robust Object Matching using a cascade of Haar classifiers.
Scale Invariant Feature Transform.

You must provide a list of the differences and similarities between the techniques. Each of the differences and similarities must be clearly explained. NOTE: Marks will only be awarded for the detailed comparison of techniques. No marks will be awarded for separate descriptions of the techniques

---

Compare and contrast:
Hough for circles.
Chamfer matching for finding circles.
Mean shift segmentation followed by an evaluation of a measure of circularity.

You must provide a list of the differences and similarities between the techniques. Each of the differences and similarities must be clearly explained. NOTE: Marks will only be awarded for the detailed comparison of techniques. No marks will be awarded for separate descriptions of the techniques

Purpose:
Each of these techniques is used for finding circles or circle-like objects in an image.
The Hough transform works by accumulating votes in the circle parameter space.
Mean Shift Segmentation segregates an image based off colour/local kernel density.
This allows a measure of circularity to be applied to the resulting regions.
Chamfer Matching does not require strict matching and allows different degrees of circle recognition.
Input:
Both Chamfer Matching and the Hough Transform take in edge maps. But Chamfer Matching requires an additional reference image to base its matching on.
Mean shift segmentation works on pixel level so colour RGB images can be used as input
Output:
The Hough Transform outputs the most likely circles based on votes.
Chamfer Matching can also produce multiple outputs at a time but instead of definitive circles. It produces a measure of how likely the shape is to be a circle.
A measure of circularity prorudces a value between 0-1, indicating how much the boundary of the segmented region correlates with a circle.
Additional Parameters:
Mean shift segmentation can take in markers from expert/human intervention.
This prevents regions bleeding into each other and creates a more reliable output.
Hough needs to know what parameter space (line,circle,etc.) it is working in.
Chamfer Matching needs information about hte type of boundary before being able to compare.
Robustness:
Measure of circularity and Chamfer Matching can both be qualitatively validated.
Both produce a figure of how well the boundary has matched a circle.
Hough requires an existing measure of what is considered circle-enough.
Chamfer matching may fail if the boundary is unclear or obscured.
Mean shift segmentation also may fail in unclear/obscured situations, making the circularity measure harder to use.
Hough is decently robust against noise.
Method:
Mean shift segmentation works by taking each pixel in the iamge and moving it towards a local kernel density until convergence, moving in the direction of increasing density. Extremely computationally expensive.
Chamfer Matching assigns every point in the image a Chamfer value. Edges = 0, non-edges = ∞.
It iterates top left to bottom right, considering preceding pixels above and left, re-assigning values based on how far from the object boundary the pixel is.
Iterates similarly bottom right to top left, considering the preceding pixels below and right.
Hough works by mapping the edge points into the circle parameter space.
Votes are accumulated indicating what the best fit(s) to the parameter space is.

2022

Compare and contrast:
Non-Maxima Suppression as used in first derivative edge detection.
Non-Maxima Suppression as used in the first stage of SIFT.
Non-Maxima Suppression as used in Moravec corner detection.
Non-Maxima Suppression as used in the Hough transform for circles where the radius is unknown.

You must provide a list of the differences and similarities between the techniques. Each of the differences and similarities must be clearly explained.
In all cases, non-maxima suppression retains only strong responses.
Each form of non-maxima suppression compares the current pixel to it's neighbourhood.
All outcomes output a sparser representation of the source image.

In first derivative edge detection e.g. Canny, non-maxima suppression inputs the gradient and direction image from the edge detection. The Non-maxima suppression only keeps the pixels that are highest along the edge detection. This results in sharp and well-defined edges being retained after the non-maxima suppression.

In SIFT keypoints, the input to non-maxima suppression is images of differing scale. The non-maxima suppression seeks to keep the keypoints that are stable and unique by finding points that can be matched across the images.

In Moravec corner detection, non-maxima suppression takes in the corner strength at each pixel and only keeps the pixels which are stronger than it's neighbours, which results in an output of strongest corners, with the purpose of only keeping the important corners in the image.

In Hough transform circles with an unknown radius, non-maxima suppression takes in the votes for the circle centres and radius and keeps the peak of the votes, which outputs the candidate circles with the purpose of detecting circles supported by edges

---

Compare and contrast:
Connected Components Analysis.
Watershed Segmentation.
Mean Shift Segmentation.

You must provide a list of the differences and similarities between the techniques. Each of the differences and similarities must be clearly explained.

All of these are methods of segmenting an image into regions. All produce labelled regions or clusters as an output.
Inputs: CCA takes in only binary images, watershed segmentation takes in greyscale images, mean shift segmentation has the capacity to take in colour images
Outputs: CCA produces a labelled image of the regions located, watershed segmentation produces segmented regions based on the flooding of catchment basins, mean-shift segmentation outputs an image which appears smoother than its input as regions are denoted based on the modal value of their neighbours
Methods: CCA tacks each pixel, making a note of each neighbouring pixel in the binary image which is a match. CCA conducts a second pass where it merges any regions from the first pass which are also connected and updates the labels as appropriate.
Watershed segmentation uses catchment basins by treating the image as a topology and "flooding" the regions until the maxima is reached. Once the maxima is reached, the segments of the image have been defined.
Mean-shift segmentation iterates over the image, analysing the modal pixel, moving the pixels towards their local maxima. This method depends on the kernel size set.
Purposes: the purpose of CCA is simply to identify regions in a binary image and label them. There is no colour or orientation analysis in CCA.
Watershed segmentation is more beneficial for capturing the boundaries of the regions. Markers can be set in watershed segmentation, allowing the algorithm to use user inputs to better identify regions. It is slower than CCA as it treats the image first as a topology.
Mena-shift segmentation preserves the colour of the image, so no pre-processing is required (e.g. thresholding to binary or converting to greyscale). It is the most computationally intensive and depends on the user's kernel selection which introduces a degree of non-determinism which is not present in CCA. Mean-shift segmentation can also preserve f

---

Compare and contrast:
Robust object detection using a cascade of Haar classifiers.
Scale Invariant Feature Transform.
Principal Components Analysis.

You must provide a list of the differences and similarities between the techniques. Each of the differences and similarities must be clearly explained.

2021
Compare and contrast:
RANSAC for finding straight lines.
Hough transform for lines.
Recursive boundary splitting to find straight lines.

You must provide a list of the differences and similarities between the techniques. Each of the differences and similarities must be clearly explained.

All of these methods are used to find straight lines in images.
All use edge or boundary points as an input.
All are used in shape or object boundary analysis.

RANSAC and the Hough transform take in a set of edge points determined from an edge detector e.g. Canny.
Recursive boundary splitting takes in a binary or edge map.

RANSAC outputs the best-fitting line
The Hough transform peaks according to the line parameters
Recursive boundary splitting outputs segments of boundaries approximated by straight lines.

RANSAC works by randomly sampling points from the edge points, fitting a line according to these edge points, and repeating until it converges, finding the line of best fit. RANSAC is non-deterministic and depends on the randomly sampled points it chooses. RANSAC is very robust at handling noise, it is explicitly designed to reject outliers that are not within the line of best fit.

The *Hough* transform works by mapping each of the edge points
Initialise Hough space accumulator to 0. for every edge point increase accumulator for every line that can go through that edge point. Search for maximums, as there is a good chance that a straight line can be found there.
The Hough transform is deterministic, as it does not use the random sampling as RANSAC does, but it is less robust at handling noise.

Recursive boundary splitting works by splitting a curve where it deviates from a straight line, until a threshold is reached where a straight line is found. It assumes that boundaries are continuous and can be approximated by these straight lines it generated. It is deterministic but less robust at handling noise. It may be sensitive if there is noise along the boundary it is trying to search for

---

Compare and contrast:
Support Vector Machines.
Chamfer Matching.
Scale Invariant Feature Transform for recognition.

You must provide a list of the differences and similarities between the techniques. Each of the differences and similarities must be clearly explained.

Compare and contrast:
Static background model.
Median Background Model.
Gaussian Mixture Model.

You must provide a list of the differences and similarities between the techniques. Each of the differences and similarities must be clearly explained.
All of the above are methods of separating a background and a foreground in a video.
They all produce a binary or greyscale mask to represent the pixels which are moving in the video.
They all use a pixel-level comparison between the current frame and the background estimate.

Each of these techniques takes in a sequence of video frames but processes them differently.

The *static background model* uses a single image as the background and detects motion by comparing all subsequent frames to this fixed background.
This is not adaptable to lighting changes and performs poorly when considering transient motion like things moving in the wind should not be included in the background.
It is deterministic and simple. Its use case is in very controlled environments where the camera does not change.

The *median background model* takes in a sequence of frames but instead of using the first frame alone as its background, it slowly adapts to consider the median of all of the frames it has processed so far. This performs better than the static background model at handling lighting changes, but it uses more memory than the static background model. It also requires several frames in order to initialise.

The *mixture of Gaussians* or *Gaussian Mixture Model* is the most complex of the three models. For each pixel, the intensity is modelled as a mixture of Gaussians with the parameters updated over time. This algorithm is the most adaptive of the three but has a much higher computational cost because it processes per-pixel probabilities. This use case is most suited for real-world systems like surveillance where there is a lost of noise in the background of the scene and changing backgrounds

---

Compare and contrast:
Gaussian Smoothing
Median Smoothing
Opening

You must provide a list of the differences and similarities between the techniques. Each of the differences and similarities must be clearly explained.
These are all methods of "cleaning" images which may have noise introduced to them.

Gaussian Smoothing and Median smoothing can work on colour and greyscale images. Whereas opening is a technique used in binary images by eroding then dilating the binary image. (Remove the edges of the foreground pixels, then pad out the foreground pixels).
Gaussian smoothing works well to combat Gaussian noise, which is a slight blur usually consistent across the image.
Median smoothing works well to combat salt and pepper noise. A type of noise where a pixel is

misrepresented as (0,0,0) or (255,255,255) or some other value distinctly different from what it should be.
Gaussian smoothing works by merging the image with a Gaussian weighted average based on the distance from the centre of a kernel.
Median smoothing works by replacing each pixel with the median value of the pixels in its neighbourhood.
Opening works to fill small holes and break up small connections in the binary image, preserving the original shape of the image more than dilation and erosion on their own.
Gaussian smoothing blurs the image smoothly. Median smoothing preserves stronger edges.

All can be used in the preprocessing of an image i.e. for region segmentation or edge detection

---

Compare and contrast:
Robust object detection using a cascade of Haar classifiers.
Template Matching.
Principal Components Analysis.

You must provide a list of the differences and similarities between the techniques. Each of the differences and similarities must be clearly explained

---

Compare and contrast:
Non-Maxima Suppression as used in first derivative edge detection.
Non-Maxima Suppression as used in the first stage of SIFT.
Non-Maxima Suppression as used in Moravec corner detection.

You must provide a list of the differences and similarities between the techniques. Each of the differences and similarities must be clearly explained.
The first step in SIFT is scale space extrema detection:
This involves considering the image at different resolutions i.e. a difference of Gaussian smoothings.
Stable keypoints locations are determined by the extrema of the difference of Gaussians across the scale space.
An extrema in the first stage of SIFT bust be bigger/smaller than all its neighbours and be bigger than the neighbours in the other spaces/adjacent scales in any octave.

Non-maxima suppression is a general technique used to determine the extrema in different algorithms.
In all cases, non-maxima suppression only retains the strong responses.
Non-maxima suppression compares the current pixel to its local neighbourhood.
All outcomes output a sparser, less-detailed representation of the source image.

Non-maxima suppression when used in first-derivative edge detection inputs the gradient (and direction) image from the edge detector e.g. Sobel/Canny. The non-maxima suppression retains only the pixels which are the highest among the edge map, resulting in sharper, better defined edges being outputted.

Non-maxima suppression when used in the first stage of SIFT (i.e. location of extrema) inputs the images of differing scales after the Gaussian smoothing has been applied at several different degrees of intensity. The non-maxima suppression locates the extrema in each scale by only retaining the keypoints which are consistent across the scale space, only retaining those with the maximum closeness.

Non-maxima suppression when used in Moravec corner detection inputs the corner strength value at each pixel and only keeps the pixels which are stronger than its neighbours, which results in an output image containing only the strongest/most likely corners

---

2024 - Application

Monday, November 03, 2025 9:44 PM

1. (a) **[APPLICATION QUESTION]** Describe how you would automatically locate and read license plates in images such as that shown below. Your solution must consist of a series of computer vision techniques and you must provide details of how the techniques will be applied including expected input and output for each technique. You must NOT use neural networks or deep learning as part of your solution.

[30 marks]

1. Convert the image to greyscale
   Input: RGB image
   Output: Greyscale image

2. Perform a Canny Edge detection. A first and 2nd derivative edge detection, non-maxima suppression and double thresholding with hysteresis to find edges in the image.
   Input: Greyscale image
   Output: Binary edge map

3. Perform the Hough Transform for straight lines on the resulting edge map to remove any lines with curvature by accumulating votes in the hough space
   Input: Binary edge map
   Output: Straight lines in the image located

4. Examine the intersections of Hough Lines to produce regions which are encapsulated by four straight lines i.e. Rectangles
   Input: Hough Lines
   Output: Regions encapsulated by Hough line intersections

5. Perform a measure of rectangularity on the resulting regions. Drop any non-rectangular shapes and drop any rectangles taller than they are wide e.g. The window grates in the background. Perform a measure of elongatedness to further eliminate false positives.
   Input: Binary Edge Map
   Output: Elongated rectangles wider than they are tall
   Note: this will destroy the blue rectangle at the left of the license plate but this is irrelevant for the purposes of this application

6. Limit the searchable space to these located rectangular regions by combining the original RGB image with the outcome of the rectangularity measure
   Input: RGB Image and located rectangles
   Output: Candidates for license plates

7. By using the previous Hough intersections of each of these remaining regions, perform a perspective geometric transform on each of these identified regions to obtain a straight on view of the license plate.
   Input: Candidates for license plates

Output: Straight-on view of candidates

8. With the search space fully limited to straight-on view of license plates, convert the RGB Image to greyscale and Otsu threshold to binary.
   Input: RGB image
   Interim: Greyscale image
   Output: Binary image

10. Invert the binary image such that the black letters are foregrounded instead of the white background.
    Input: Binary image with backgrounded lettering and foregrounded white plate
    Output: binary image with foregrounded lettering for better detection

11. Clean the binary image with opening and closing to break apart close together regions, fill small holes and preserve the overall structure of the characters.
    Input: Noisy binary image
    Output: cleaned binary image

12. Perform Connected Components Analysis on the binary image to produce an identity for each character.
    Input: Binary image
    Output: Binary image with foreground labels

13. For each CCA region (a character), obtain measures of elongatedness, rectangularity, circularity, convex hull, minimum bounding rectangle, area, perimeter, in order to construct a linear classifiers table which will correspond each character on the license plate with the closest match.
    Input: CCA regions (characters)
    Output: A read license plate

Notes before starting:

- It is clear that the license plates in an image are at different angles. Once we know whereabouts the plates are, we can perform a perspective transform. In order to do a perspective geometric transform, we need four feature points. We can find the feature points with Moravec corner detection.
- I don't want to use binarty thresholding because one of the cars is white.
- However, if I convert the image to greyscale and do a binary threshold then I have the black parts of the license plates and the corners. Then I can do CCA on the black parts of the license plate, maybe after an opening.
- License plates also lie at the bottom of the car, so it may be possible to cut off the top half of the image to reduce computational complexity
- Cars are very round, so we may be able to apply a Moravec corner detection to single-out the license plates. We can see where 4 corners are in the image.

Ans:

- Starting with an RGB image containing several cars at different geometric perspectives.
- Assuming that images to be analysed are similar to the example image, displaying cars lined up in a row, I will cut off the top half of the input image, as license plates are located at the bottom of the car.
- Also assuming that all license plates follow the same structure as the input image i.e. Republic of Ireland license plates, I want to perform a Chamfer Matching on these license plates against a target image containing some example of a Republic of Ireland license plate

1. First I will convert the image to **greyscale**.
2. Then I will perform a **Canny Edge Detection**. (Chamfer matching requires an edge map and Canny requires a greyscale image). Non-maxima suppression is completed as part of the Canny

process.

3. Then I will perform a **Chamfer Matching** against the RoI license plate target image. This gives a measure of similarity between the target image and locations in the input image. I have chosen Chamfer matching as it can deal with the different sizing of the license plates (close, far away) and within reason the different perspectives, provided the license plate is not unreasonably skewed out of view of the camera. I am assuming that the license plate is fully in frame and its contents could be fully legible.
4. Chamfer Matching may return some **False Positives** in shapes which are rectangular and contain some darker objects within them e.g. the white railings in the example image. To mitigate this I will apply a **threshold** which states that the objects must have at least a 75% match with the Chamfer target image to be considered valid.
5. I will **overlay** this measure of similarity with the original RGB image such that we have only isolated the license plates by using a bitwise AND. *[LOCATING REQUIREMENT FULFILLED]*
6. Maintaining the four corners of the license plate, I will perform a **Perspective Geometric Transform** to give a fully straight-on view of the located license plates.
7. This greyscale, straight-on view can have an **Otsu binary thresholding** applied to it, to isolate the black letters from the white background and the lighter blue strip on the right.
8. I will apply **Connected Components Analysis** on this binary image, successfully extracing the individual letter and number regions. *[READING REQUIREMENT FULFILLED]*
9. This system may achieve false negatives where the car is very far away, such that the Chamfer Matching is unable to detect it.

2. (a) **[APPLICATION QUESTION]** Describe how you would automatically locate and track the coloured helmets of horse riders in a video feed from a stationary camera which gives views like those below. Your solution must consist of a series of computer vision techniques and you must provide details of how the techniques will be applied including expected input and output for each technique.

[25 marks]

Photo by Mathew Schwartz released under the Unsplash License

Photo by Jeff Griffith released under the Unsplash License

- Mean Shift Segmentation - localised colour
- **K Means is NOT localised, it is across the whole image!** But can be used.
- Top region
- High saturation
- Circular
- Moving region

1. Use a **Gaussian Mixture Model** as a background subtractor. This video is filmed outside and GMM takes in video frames and produces a binary output separating the foreground from the background while eliminating small movements such as trees blowing in the wind. Only moving regions need to be considered for this tracking.

2. Perform **Mean Shift Segmentation** on the regions of interest *only* to extract colours. Mean

shift moves colours towards a local kernel density to give colour and region based segmentation.

3. Perform a threshold on the **Saturation channel** to identify the brightly coloured helmets. Mean shift segmentation has ensured that colours are smooth across their respective regions. This produces a binary image which may produce false negatives e.g. The darker brown helmet may not be picked up by this.

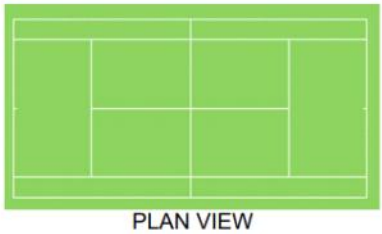
4. Perform **Connected Components Analysis** on the binary result. This will give an identity to each of the individual binary regions identified.

5. For each region identified by CCA, perform a **Measure of Circularity** on the region, helmets should be vaguely circular.

6. Use **Mean Shift Tracking** on each of the identified CCA objects (overlayed with the original RGB video). This tracks the motion of the different helmets in the scene by taking two consecutive frames in the video and represent the speed/direction of the movement.

3. (a) **[APPLICATION QUESTION]** Given an image of a tennis court (such as that shown below) describe how you would automatically locate the court (i.e. the white lines) using edge detection and transform it to a plan view. Your solution must consist of a series of computer vision techniques and you must provide details of how the techniques will be applied including expected input and output for each technique. You must NOT use neural networks or deep learning as part of your solution.

[25 marks]



PLAN VIEW

COMPLETE IMAGE OF THE COURT

1. Convert the image to greyscale.
   Input: RGB Image
   Output: Greyscale image

2. Perform a Canny edge detection on the greyscale image. A 1st and 2nd derivative edge detection with non-maxima suppression and double thrseholding by hysteresis to identify all edges in the image.
   Input: Greyscale image
   Output: binary edge map

3. Perform the Hough transform for straight lines on the resulting edge map to eliminate non straight lines i.e. People, cars. It accumulates votes in the Hough space for each edge point, voting for how straight the edge points are.
   Input: Binary edge map
   Output: Straight lines in the image

4. This will result in a number of false positives for detecting the court, i.e. The parking spaces are straight lines and the net is a straight line. Evaluate the intersections of the Hough lines.

This will intersect/create regions across all the lines in the tennis court, the net, etc.
Limit the computation to regions which are fully encapsulated by Hough intersections i.e. Rectangles/squares
These regions have four straight lines on each side.
Input: Hough Lines
Output: Intersections of Hough lines and the encapsulating region they produce i.e. Squares/rectangles.

5. I am making the assumption that the tennis court is the focal point of the image, and there is no set of straight lines in the image which could possibly produce a larger area. With this assumption, count the number of pixels contained within each Hough intersection. Take the one with the largest number of pixels and thus largest area as the located tennis court.
   Input: All regions encapsulated by Hough lines
   Output: the largest region encapsulated by Hough lines

6. The computation has been reduced down to the maximal area encapsulated by Hough intersections. By taking these intersecting points as feature points, perform a perspective geometric transform on the planar tennis court to produce a plan view of the court.
   Input: 4 feature points / Hough intersections
   Output: Tennis court plan view

Ans:

1. First I will convert the image to **greyscale** as most edge detection algorithms take in a greyscale image as an output.
2. Then I will apply a **Sobel Edge Detection** to the image. I have selected Sobel as I would like to preserve the direction of the edges. This is because tennis has a specific court, therefore counting the number of horzontal vs. vertical edges in the plan view will help confirm if the located court is or ir not a tennis court. Sobel edge detection is also more noise-tolerant than Robert's 1st Derivative Edge Detection. I have made this distinction because there is some noise present in the image e.g. the tennis players standing on the field. (The noise from these players could also be mitigated by a Gaussian blur, however I have chosen to omit this as I do not want to potentially destroy the edges before they are detected. This is a risk in this implementation and could lead to false negatives)
3. With the edge, magnitude and direction image that Sobel produces, I would be able to extract a large rectangle (two horizontal edges, two vertical edges), with many smaller rectangles contained inside it. This gives me an image of the court. By using the orientation that Sobel provides, this will filter out any edges that do not meet this criteria e.g. the lamppost which contains two vertical edges in the image and the posts of the gate. I will filter out any edges that are not of a significantly large size e.g. the storage box by using **contour following**. This is under the assumption that the tennis court is the focal point of the image and there are no set of edges which are significantly larger than the edges of the tennis court.
4. [*OPTIONAL: IF NECESSARY*] This segmentation of the tennis court could further be refined by a **Watershed Segmentation** which also have the capacity to segment out the different rectangles within the court.
   a. Watershed segmentation takes a greyscale image and treats the image as a topological graph
   b. Pixels "flow" towards the minima, revealing distinct regions
   c. In the context of this implementation, watershed lines can also be applied to the segments of the tennis court to better spearate them out, although this requires a level of human intervention.
5. Given the edges/regions identified in step 3/4/both, I would apply **Moravec corner detection** to the region. I have chosen Moravec as the example image appears to be of a low resolution and I think it is ok to assume that images containing an entire tennis court are taken from far away and thus lose some resolution. Moravec corner detection is more suited for lower resolution images than Harris-Plessey which works better on higher resolution images
   a. Moravec corner detection compares the local variation around a point in the image.
   b. The lower the similarity of the pixel is to its surroundings, the more likely it is to be a

corner.

6. By selecting the four outermost cornors (easy because we already have the main edges/region of interest), we can apply a **Perspective Geometric Transformation** on the four main feature points. This gives the desired Plan View of the tennis court.
    a. Perspective geometric transformation works by mapping points to new points using projective relations that can converge lines.

2023 - Application

Monday, November 03, 2025 9:45 PM

1. (a) **[APPLICATION QUESTION]** Describe how you would find blue signs (such as those in the pictures below) and recognise the writing and arrow signs. Your solution must consist of a series of computer vision techniques and you must provide details of how the techniques will be applied including expected input and output for each technique.

[30 marks]


1. Histogram back project blue onto these images. Take a large sample of royal blue under different lighting conditions to isolate blue in the image.
   Input: RGB Image, samples of royal blue
   Output: Greyscale probability map of royal blue in the scene

2. Otsu Thresholding. Get binary view of the image.
   Input: Greyscale image.
   Output: Binary image

3. Binary image may be noisy as this was a probability map before. Clean with opening (erosion-> dilation) and closing (dilation->erosion). This breaks apart close together regions, fills small holes and maintains the structure of the image.
   Input: Noisy binary image
   Output: Cleaned binary image

4. Perform Connected Components Analysis to label the segments of the binary image. Two passes to assign a label to a pixel based on its connectivity to its neighbours, and a 2nd pass to connect connected large regions.
   Input: Cleaned binary image of blue pixels
   Output: Labelled binary image

5. Perform a measure of rectangularity on each of these CCA regions to avoid false positives e.g. A blue car, the blue sky. Drop any poor matches for rectangles.
   Input: CCA regions
   Output: Rectangular CCA regions

6. Perform Contour following on each of the identified CCA regions to obtain a view of the boundaries of the CCA objects. Contour following traces the object based on the similarity of the colour/intensity.
   Input: binary regions
   Output: boundaries of binary regions

7. Using the maximal/minimal x and y coordinates for each of these individual boundaries, perform a perspective geometric transform on the planar signs by using the x,y max,min as four feature points. This allows reading the letters and arrows to become easier

Input: 4 feature points / top&bottom, left&right pixels of the object boundaries
Output: Straight-on view of the signposts

8. Invert the binary image such that the lettering becomes foregrounded and the blue (which is white currently) becomes backgrounded.
   Input: White sign, black lettering
   Output: White lettering, black background

9. Clean each of the letters with opening and closing as described in step 3 and perform CCA on each letter as described in step 4.
   Input: binary characters
   Output: cleaned and labelled binary letters

10. For each labelled binary character, obtain measures of perimeter, area, rectangularity, circularity, minimum bounding rectangle, convex hull, etc. Use these measures to construct a linear classifiers table for each of the letters. Select the letter with the highest probability of match as the found letter.
    In the case of arrows, they will have the same circularity, rectangularity, etc. The convex hull can still be used to linearly classify the arrows.
    Input: Binary characters
    Output: Classified/read characters

Ans:

1. All of these blue signs are of a similar shade of blue. They do not massively differ in terms of light blue, navy etc. the only thing that changes about the colour of these signs is the quantity of white text on the sign and the time of day which changes the lighting. Perform **Histogram Back Projection** against the sample of blue signs.
   a. Histogram back projection works by taking a set of training images and a query image and comparing a colour (or greyscale) histogram of the training image to each pixel in the source image to find where similar colour distributions lay.
   b. This appears to be reliable at finding the blue signs, and can even deal with cases where there may be multiple blue signs in the image. However, it may produce **False Positives** in situations where an object of a similar colour is also visible in the frame e.g. a blue car driving by in the scene.
2. Histogram back projection gives a greyscale image equivalent to the probability of the similarity in colour. This gives a vague notion of where the blue signs are. By applying an **Otsu Threshold** to this with a very high threshold, we can extract the precise locations where the blue signs are.
3. **Connected Components Analysis** can be used on this binary image to fully extract the region of interest
   a. CCA works by making a first pass over a binary image, labelling the white pixels and any pixels connected to that region
   b. It then makes a second pass to converge any labelled sections which may also be connected to each other
4. CCA operates on binary images, once the regions of interest are extracted, they can be applied onto the original source image. To mitigate the false positive scenario described in Step 1, apply a simple edge detector e.g. **Roberts First Derivative Edge Detection** can be applied after the region area is converted to greyscale. I have chosen this as edges are not the main point of this appliation, I do not need the detector to be robust against noise like Canny or preserve the orientations like Sobel. By applying **Recursive Boundary Splitting** and counting the number of times recursive boundary splitting occurs, rounded objects e.g. blue cars can be filtered out. Whereas regions with not a lot of recursive boundary splitting required e.g. straight edge signs will need less recursions
   a. Roberts Edge Detection works by computing gradient magnitudes from differences between diagonally adjacent pixels
   b. It requires a threshold to be applied, defining what is considered a valid edge
   c. It requires non-maxima suppression to be applied, a technique which only keeps the "most valid" edges by iteratively removing the weakest links

    d. Recursive boundary splitting works by taking in the source image and the defined contours for that image. It outlines the detected contours as a series of striahgt lines by recursively taking the line and making it smaller and smaller such that a rounded line is represented by many tiny straight lines.
5. *[ALTERNATIVE TO STEP 4 - PROBABLY A LOT BETTER]* The **Hough Transform for Straight Lines** is another way to remove false positive blue objects. This technique works natively on the binary image that CCA operates on. It takes in the binary image and some parameter space which to map the image to e.g. straight lines.
    a. It initialises a Hough space accumulator to zero and increases the accumulator for every edge point that it comes across.
    b. It often gets multiple responses and therefore applies non-maxima suppression in order to maintain only the "most valid" straight lines.
6. Only identified regions that are made up of within-reason straight lines are valid signs, this removes more rounded blue objects from detection.
7. By using a Bitwise and from these extracted valid regions and the original image, we are now only considering an in-colour version of the valid regions
8. Use **colour thresholding on the red channel** to extract the white text from the blue background. The blue and red channels are non-overlapping and white has a strong red component as white consists of (255, 255, 255) in RGB space. Blue will be null in the red channel and white will be revealed strongly. This leaves a binary image containing only the extracted text from the sign.
9. To better read the sign, an **opening** and **closing** could be applied to the binary image.
    a. Opening is erosion followd by dilation, preserving the benefits of erosion, this can be used to break apart close-together regions e.g. letters which may be hard to read as they are close together
    b. Closing is dilation followed by erosion, preserving the benefits of dilation, this fills small holes and can make the letters more legible in cases where the image resolution in poor and the letter has not fully been picked up by the red channel thresholding

2. (a) **[APPLICATION QUESTION]** Describe how you would track the two players on a tennis court (who will change ends regularly and will also go out of view from time to time), from a video of the court (from an unmoving camera) with views such as those shown below. You may **not** assume that the clothing will be as shown. The players can wear any clothing they wish to wear. Your solution must consist of a series of computer vision techniques and you must provide details of how the techniques will be applied including expected input and output for each technique.

[25 marks]

1. Use a **Gaussian Mixture Model** as a background subtractor.
   This video is outdoors and movement from branches, etc. May be detected in another background subtractor.
   It is ok to assume that people are moving when they are playing tennis.

GMM takes in the video frames and outputs a binary video separating the foreground and background

2. Use an **Opening** and **Closing** operation on the binary images to clean them.
   The result of the GMM should be just the moving people. Opening outputs the image after erosion and dilation. It will remove any noise present around the person. Closing outputs the image after dilation and erosion. It will fill small holes present in the binary person.

3. To find the associated shapes in the image, apply **Connected Components Analysis**. This will give an identification to each moving binary region in the image.

4. To confirm that these shapes are people, do a **Measure of Rectangularity** and a **Measure of Elongatedness** on each of the objects which CCA found. People are vaguely rectangular and elongated objects. This will rule out things like moving birds landing on the court.

5. Once people have been confirmed, **Mean Shift Tracking** can be used which takes two consecutive frames of the video and searches for a region of similar size and weighted probabilities of where the person is moving to. It outputs a bounding box around the moving person.

6. If there is no foreground (white) pixels found in the binary image, it indicates that the players have left the court. At this stage, the CCA, shape analysis and tracking does not need to be run until foreground pixels reappear in the video.

7. To prevent false positives, it may be beneficial to introduce a **threshold** which counts how long moving pixels should be present in the frame before beginning tracking.

3. (a) **[APPLICATION QUESTION]** By finding the *vanishing lines* (as shown in red below) describe how you would obtain a front view of a building. Your solution must consist of a series of computer vision techniques and you must provide details of how the techniques will be applied including expected input and output for each technique.
[25 marks]


Original image


Front view

The *vanishing lines* (which correspond to lines which are parallel in the real world) intersect in a single point (called the vanishing point; shown in green).

Notes before starting:

1. Convert the image to greyscale
   Input: RGB image
   Output: greyscale image

2. Perform Sobel Edge Detection on the greyscale image by convolving the the pixels with a Gaussian distribution, accounting for the fact that edges change slowly and not suddenly by one pixel, and examining the orthogonal rate of change to give a gradient magnitude and gradient detection for the edges. This requires non-maxima suppression and thresholding to produce a binary output. I have chosen Sobel over Canny as this application requires knowledge of both horizontal and vertical edges.
   Input: Greyscale image
   Output: binary edge map

3. Only considering the horizontal points from the Sobel detection (i.e. Suppress the gradient verticals), perform the Hough transform for straight lines to remove any edges with curvature from detection. This takes each edge point and accumulates votes in Hough space for how straight the lines are
   Input: binary edge map
   Output: detection of straight horizontal lines

4. For each Hough Line in the image, find the point in the image which the maximal number of Hough lines intersect it. The vanishing point has been located.
   Input: Horizontal lines
   Output: Maximal intersection / vanishing point

5. Constraint the computation to consider only horizontal edges which intersect with the vanishing point. Examine the angle between the edges and the vanishing point. Choose the two greatest angles to give the maximal y coordinate vanishing line and the minimal y coordinate vanishing line.
   Input: all vanishing lines
   Output: top and bottom vanishing lines

6. Returning to the output of the original edge detection, suppress and threshold conversely to obtain a view of the vertical lines. From this, perform the Hough transform again as described in step 3.
   Input: All vertical lines
   Output: all vertical straight lines.

7. With the obtained vertical straight lines, use the top and bottom vanishing lines to determine the vertical lines where the vanishing lines intersect perpendicularly
   Input: all vertical straight lines
   Output: vertical straight lines which the max/min vanishing points intersect perpendicularly.

8. From these perpendicular intersections, select the vertical lines with the maximal and minimal x coordinate in the image i.e. The building has been encapsulated by the max/min vanishing lines and their corresponding max/min perpendiculars.
   Input: All verticals intersecting with the max/min vanishing points perpendicularly
   Output: the max/min x coordinate vertical lines

9. There are now just 4 Hough intersections remaining. Use these 4 points to perform a perspective geometric transform and obtain a front view of the building
   Input: 4 intersections / 4 feature points
   Output: front view of the building

2022 - Application

Friday, October 31, 2025 4:17 PM

1. (a) **[APPLICATION QUESTION]** Describe how you would reliably locate emergency exit signs such as those shown the top row of images below. Your solution must consist of a series of computer vision techniques and you must provide details of how the techniques will be applied including expected input and output for each technique. [25 marks]

First row of photos shows some emergency exit signs.

Second row of photos shows some other signs/objects.

Notes before starting:

- All of the emergency exit signs are the same size
- So if I apply some transform of the viewpoint then they will all be forward facing
- There may be some lighting differences in the image so histogram back projection is not really reliable here
- But after all the images have the same viewpoint then the green area should have a roughly similar rectangularity, elongatedness, etc. Value
- The white icons inside the emergency exit signs are also the same. If I obtain a binary image of this and the % of pixels which are white should be roughly the same across the board because I've already done a perspective transform, so just count the pixels

Process:

1. These appear to be images of a moderate resolution. I will perform a **Moravec Corner Detection** on this image to identify the corners in the image.
    a. Moravec Corner Detection works by comparing a pixels to its local surroundings
    b. The less similarity there is between the current pixel and its neighbours, the more likely it is that it is a corner.
2. This detection will also produce corners from within the sign e.g. the white door symbol and any corners in the background. However, the emergency exit signs are all the same size. I can resolve this by only considering the four corners which are an appropriate distance apart e.g. further apart in length than they are in width. The pixels between the corners should be of a certain **threshold**.
3. A further confirmation which could be carried out is to select a pixel which is slightly "inward" from the detected corner e.g. if you select the top right corner detected, select some pixel which is more-left and more-downwards than that corner. That pixel should be green, again because all emergency exit signs are the same shape, provided that the pixel selection is not overlapping with the person icon. See if this pixel has a value which is approximately **near (0,255,0), within a certain threshold**.
    a. The green channel is overlapped by both red and blue so I could not simply threshold "yes"/"no" on the green channel.
    b. I can cheat around this though by comparing the colour of a pixel within the sign to the value (0,255,0). The value of the green pixel in the sign should be approximately similar

to this value, within reason. This assumes that the image has not been taken in complete darkness. Some shadowing on the green pixel of interest is ok as it should not be wildly dissimilar from (0,255,0).

4. Once Step 2 and Step 3 have confirmed that you are most likely dealing with a green sign of an approximately valid shape, perform a **Perspective Geometric Transform** using the valid 4 corners obtained in Step 2. This will allow the sign to be straight-on. This assumes that there is only one emergency exit sign of interest in the frame.
    a. Perspective Geometric Transformation works by taking in four feature points as an input.
    b. It works by mapping points using projective relations which can converge lines
5. Convert this straight-on region to greyscale, then to binary using **Otsu Thresholding**. The white icons may be blurry due to the perspective change to perform an **opening** to mitigate this.
    a. Otsu Thresholding works by taking a greyscale image and obtaining its histogram.
    b. It computes the threshold value T and replaces pixels with saturation below that threshold as background (black) and replaces pixels with saturation above that threshold as foreground (white).
    c. Opening works by performing an erosion on the binary image then a dilation. It preserves the general shape of the binary image features but maintains some of the benefits of erosion.
    d. Erosion takes the foreground objects of the binary image and removes the outermost pixels, it is used to break apart close together areas
    e. Dilation takes the foreground objects of the binary image and pads the outpost pixels with more foreground, it is used to fill small holes.
6. Again, all emergency exit signs are the same, therefore they should contain an approximately similar amount of pixels. **Count the number of foreground pixels** in the cleaned binary image and compare this to an appropriate threshold i.e. by using a sample of an emergency exit sign, not obscured by perspective shift or time of day lighting changes.
7. If the percentage of white pixels in our image is **within a certain threshold** of the sample image, we have identified a potential emergency exit sign.

- I have denoted Step 3 as "optional" because comparing the distances between corners as per Step 2 should give a reliable amount of true positives for emergency exit signs, if one were to omit the colour analysis conducted in Step 3, a hypothetical emergency exit sign of a different colour.
- This could also produce false positives on a hypothetical green, rectangular object which happens to contain a similar % of white pixels as an emergency exit sign

1. Histogram Back Projection on Green
2. Convert to Binary
3. CCA
4. Rectangularity measure, drop non-rectangles, drop rectangles whose height > width
5. Search space now limited to green rectangles which are wider than they are tall. Invert binary image
6. CCA on the inversion. There should be three distinct shapes if it is the emergency exit sign. Discard if more than 3 CCA regions or less than 3 CCA regions
7. Shape analysis, convex hull, MBR

2. (a) **[APPLICATION QUESTION]** Using edge detection describe how you would automatically locate a set of downwards steps (such as those shown below). You may assume that the edge of the steps are highlighted by stripes of solid colour which are distinct from the colour of the steps and that the stripes are all the same physical size. You cannot assume any prior knowledge about the colour of the stripes or of the steps or the surrounding area. Your solution must consist of a series of computer vision techniques and you must provide details of how the techniques will be applied including expected input and output for each technique.

[25 marks]

The photos show a series of images of a staircase taken by a camera (frames 10, 43 and 95 from a video sequence) approaching the tops of the stairs. The stripes on each step (yellow & white in this case) will always be a different colour to the colour of the steps, but the colour can vary.

Notes before starting:

- This question states that edge detection must be used
- I would use Sobel edge detection for this question because it also provides the direction of the edge as well as the magnitude. There are clearly a lot of edges in this image so having the direction would allow me to single out just the horizontal lines, perhaps by starting at the bottom of the image and working my way up.
- Downward steps clearly have a decreasing distance between them. The distance between the top step and the 2nd topmost step is way thicker than that of the bottom step and 2nd bottommost step. So knowing this some form of distance calculation can be done.
- The only blocker is that the coloured bar will introduce more edges so the distance between edges won't be this gradual decrease like I want.
- The coloured bar is designed for safety so its obviously going to be way brighter in the luminance channel than the carpet.
- We are looking for just the top edge of the coloured bar, not the bottom edge of the coloured bar. So for each edge, check the luminance of the pixels directly above it
- If the edges (after discounting the edges introduced by the bottom of the coloured bar) are constantly decreasing in distance, we have stairs

Process:

3. (a) **[APPLICATION QUESTION]** Describe how you would locate and recognise bicycle symbols printed on the road such as those shown below. Discuss when your approach might give false positives and what you might do to deal with that. Your solution must consist of a series of computer vision techniques and you must provide details of how the techniques will be applied including expected input and output for each technique.

[25 marks]

First row of photos shows some cycle lane markings on the road

Second row of photos shows some other road markings.

1. Convert the image to greyscale.
Input: RGB image
Output: Greyscale image

2. The road markings we want to consider are white. To avoid detecting any other coloured shapes, perform an Adaptive Threshold on the greyscale image. Adaptive is necessary due to lighting changes from streetlights or car lights, as seen in the bottom left sample image. This takes an n*n portion of the image and thresholds this segment individually, to produce an overall uniform threshold
Input: greyscale image
Output: binary image

3. The road markings appear to be faded, as seen in the bottom right image. To prevent false negatives due to this fading, clean the binary image with closing to fill small holes with dilation and preserve the overall structure of the image with erosion.
Input: binary image with breaks
Output: cleaned binary image

4. Perform a Roberts edge detection, as this is the edge detector most suitable to operating on binary images. This compares one column to the next to detect changes in lighting on one-pixel wide level only. This gives a very thin edge map.
Input: binary image
Output: binary edge map.

5. Perform the Hough transform for circles against this edge map. Do not fix the radius parameter as circles of different radii must be detected for this application i.e. The wheels of the bicycle icon and the circle where the pedals are. This will result in detected circles of multiple radii.
Input: binary edge map
Output: detected circles

6. The bicycle circles are hollow. To prevent false positive e.g. Splotches of white paint on the

road, examine the centre points of the circles. The circles which we wish to consider should consist of two circles with the exact same centre point and different radii. Drop any circles which are standalone (non-hollow). And drop any hollow circles which do not have a corresponding matching centre point with another circle.
Input: detected circles
Output: twin circles with the same centre point

7. This still leaves some false positives e.g. "STOP" markings on the road meet the twin circle criterion.
   By taking the detected twin circles, return to the original cleaned binary image and perform connected components analysis on these areas. Bicycles should consist of three of these twin circles being connected by the same CCA label. The letter "O" in the word "STOP" will be a standalone circle in its own CCA region.
   Input: twin circles with the same centre point & the cleaned binary image
   Output: labelled binary image, dropping any regions which do not contain three twin circles

Notes before starting:

- As the question states, this will inevitably have false positives
- First I would convert the image to greyscale then use Otsu thresholding to convert the image to binary.
- There appears to be some areas where the white road marking are faded so I would dilate the image to fill in the small holes
- There is also some areas in the bicycle that I would like to preserve, so I would then erode the image. This is equivalent to a Closing operation
- The edges of the bicycle are important. I would use a Canny edge detection to get the edge points.
- The bicycle road markings have clear circles. I would use the Hough transform to find the circle points in the image.
- This is where false positives may occur. For example, there may be "lane closed" or "buses only" as a road marking, and the o's in these words would return positive for Hough circles.
- However, the bicycle road marking contains two large circles where the wheels are and a smaller circle underneath the pedals.
- This requires applying the Hough circles transform a second time with the approximate radius of the smaller circle.
- If all three circles can be identified, I am confident that a bicycle road marking has been identified

Process:

1. Starting with an RGB image, I will convert the image to greyscale then apply **Otsu thresholding**. I am dealing with white text on a black background, so I am converting to binary first to eliminate anything else e.g. the kerb/footpath which may be in frame.
   a. Otsu Thresholding works by taking a greyscale image and computing its histogram.
   b. It computes the threshold value T.
   c. Any pixel in the greyscale image where saturation is below T is replaced by background (black) and any pixel in the greyscale image where saturation is above T is replaced by foreground (white).
2. With this binary mask, I will perform a **dilation** to fill small holes in the binary image e.g. where the paint of the road marking has come away
   a. Dilation works by taking the foreground objects in a binary image and removing the outermost pixels. It is used to break apart close together regions
3. I will perform an **erosion** (this is equivalent to a **closing** operation) too, as there are important edges in the image I would like to perserve.
   a. Erosion works by taking the foreground objects in a binary image and padding the outermost pixels. It is used to fill small holes.
4. Using a Bitwise AND with this binary mask and the original image gives an RGB image with the areas of interest. I would convert this RGB image to greyscale, such that I can apply **Canny**

Edge Detection.

a. Canny Edge Detection works by performing a Gaussian Smoothing operation
b. It then uses some other edge detection e.g. Sobel to get an edge map
c. It double thresholds the results and performs non-maxima suppression to ensure that only the most valid edges are kept
d. It uses edge tracking by hysteresis to compute the edge lines

5. With the binary edge map from the edge detection, I would apply the **Hough transform for circles** to the binary image. This may produce false positives as the questions mentions e.g. if the letter "o" was written as a road marking. However, the bicycle icon contains three circles. Two larger circles where the wheels are and a smaller circle where the pedals are. If the two large wheels can be found in the initial Hough Transform pass, I would perform another Hough Transform with a smaller radius, ensuring that if a smaller circle is found, it is an appropriate distance away from the two larger circles.
    a. The Hough transform works by taking in a binary edge image and some parameter space which it should be mapped to e.g. circles.
    b. It accumulates votes for how well each point can be mapped into the parameter space and the most votes is the most likely occurrence of that shape.
6. If these three circles can be found, a bicycle road marking can be found.

2021 - Application

Friday, October 31, 2025 4:38 PM

1. (a) **[APPLICATION QUESTION]** Given an image of a packing label taken from any angle (such as that shown below left) describe how you would locate and extract any visible bars codes within the label (again as shown below right). Your solution must consist of a series of computer vision techniques and you must provide details of how the techniques will be applied including expected input and output for each technique. [25 marks]

Image of Packing Label

Extracted Bar Codes

1. Convert to greyscale
   Input: RGB image
   Output: Greyscale image

2. Adaptive threshold to binary. Due to poor lighting. Take n*n portion of the image and threshold separately across image to obtain a flush binary view.
   Input: Greyscale image, size of segments
   Output: Binary Image

3. Clean the binary image with opening (erosion->dilation) and closing (dilation->erosion) to mitigate noise from the threshold.
   Input: noisy binary image
   Output: cleaned binary image

4. Do CCA on the binary image to label pixels which are connected together and to iterate the regions in post-processing. Pass and give pixels the same label if they are conencted. Pass again to connect any regions which are connected together.
   Input: binary image
   Output: labelled binary regions

5. Measure the rectangularity of each CCA region to get rid of false positives e.g. The white paper in the background of the sample image. Drop any poor rectangle detections from the computation
   Input: labelled binary regions
   Output: binary rectangles

6. Use contour following on the located binary rectangles. This application makes the assumption that the package is the focal point of the image and there are no other significant white rectangles in the image. If this assumption is invalid, it can be mititgated by counting the number of pixels contained within each CCA region and maintaining the largest. Contour following follows the pixel intensity/colour to give the boundary of the region.
   Input: binary rectangle
   Output: outline of the binary rectangle

7. Use recursive boundary splitting on the countour points. Drop any detections which have a

significant number of lines (and hence curvature). This represents the contour points as a series of straight lines.
Input: Contour points
Output: series of straight lines representing the edge of the label

8. Take the maximal and minimal x and y coordinates of the resulting straight lines and use these points as 4 feature points to perform a perspective geometric transform and get a front-on view of the label.
   Input: max/min x&y of the label / 4 feature points
   Output: straight on view of the label.

9. Invert the binary image such that the text on the label becomes foregrounded.
   Input: white label, black text
   Output: black label, white text.

10. Clean and CCA the text as described in steps 3 and 4
    Input: White text on the label
    Output: Cleaned and labelled text

11. For each individual connected component, use the pixel locations to input into a RANSAC algorithm for straight lines. RANSAC takes two random points from the data, evaluates how well they fit the model then iteratively adds more points until convergence. Use the result of RANSAC to identify all the lines which are co-linear. These lines should all have different x values and a very similar y value. Drop any CCA regions which do not meet this criteria.
    This will drop letters (not lines)
    This will drop the large horizontal lines across the package (co-linear but not a similar y value).
    This leaves us with just vertical straight lines with similar y values to its nearest neighbours.
    This is the barcodes.
    Input: CCA regions
    Output: Barcodes

Ans:

1. This is clearly a binary problem. First I will convert the image to greyscale then convert the image to binary using **Adaptive Thresholding**. Opting for adaptive thresholding over Otsu thresholding as this package seems to be under some lighting changes with in the small space it occupies, also evident by the glare on the package.
   a. Adaptive Thresholding works by splitting an image into sub-images
   b. It selects a block around a pixel and computes the mean of the pixel values in that block.
   c. It obtains a local threshold from this local mean and uses the local threshold to distinguish if pixels should be foreground or background pixels.
   d. There does not appear to be much black on the label aside from the barcode which is the area of interest.
2. ANDing this binary image with the original RGB image will give an RGB image which only contains the black areas of the label, which is mostly the area of interest.
3. I will convert this image to greyscale and apply a **Harris-Plessey Corner Detection** to these areas of interest. As I would like to compare the difference in the eigenvalues in this image for the purpose of distinguishing the barcodes from the longer black strips on the package. There should be a large amount of successes in the regions where the barcodes are (both eigenvalues high) and a smaller number of successes in the areas where there is simply a long black strip on the package (one eigenvalue high).
   a. Harris-Plessey corner detection takes in a greyscale image and compares each pixel to its local surroundings by using a sum of squared differences.
   b. It represents this as a square matix with computed eigenvalues. If both eigenvalues are high, a corner is detected
   c. If one eigenvalue is high, an edge is detected
   d. If no eigenvalues are high, a constant region is detected.

4. I would evaluate this outputted greyscale image from the bottom-up, as I go eliminating areas which have a weaker greyscale intensity (less likely to be corners). Looking for a lot of corners/features in short bursts close to each other will reveal where the barcodes are.
5. In these areas of intrest which have a significant threshold of corners, I would extract the top left, top right and bottom left corner from each of these areas of interest and perform an **Affine Geometric Transformation** on them. This is to orientate the barcode in a way that appears more striaght e.g. the barcode will appear fully vertical. It works by rotating the three chosen feature points along the axis of where I want them to end up by backwards-mapping
    a. Affine Geometric Transformation works by taking 3 points and applies a linear mapping of the coordinates including rotation, scaling, translation
    b. It corrects geometric distortions while preserving straight lines
6. I think a small **Perspective Geometric Transformation** is also required to ensure that the barcode is fully straight-on to the user of the application. This can be used by taking the fourth corner of the barcode area e.g. bottom right which was not used by the affine transformation.
    a. It works by mapping points using projective relations which converge lines.
7. Step 5 and Step 6 of this process must be iteratively applied for every suspected barcode in the image. But Steps 1-4 will work if there are multiple bar codes in the image. It may be useful to keep a counter to track how many times the geometric transformations have been applied. As different barcodes can lie at different angles, the original image must be the starting point at each iteration of the geometric transformations.

2. (a) **[APPLICATION QUESTION]** Describe how you would automatically locate and track the coloured helmets of horse riders in a video feed from a stationary camera which gives views like those below. Your solution must consist of a series of computer vision techniques and you must provide details of how the techniques will be applied including expected input and output for each technique.

[25 marks]

Photo by Mathew Schwartz relesaed under the Unsplash License

Photo by Jeff Griffith relesaed under the Unsplash License

1. Gaussian Mixture Model. This is an outdoor scene with trees in the background and sand being kicked up by the horses.
   Input: collection of grames represented as a Gaussian distribution
   Output: binary frames where background (within the Gaussian) is black and moving foreground (outside the Gaussian) is white.
   Only moving pixels are highlighted in the scene now.

2. Considering just the moving pixels as a mask, perform a mean shift segmentation which gives a smooth colour to each colour/region. It moves the pixels towards the local kernel density.
   Input: Regions of interest/moving pixels

Output: Region and colour segmentation of the regions of interest.

3. To isolate the helment from the jockey & horse, threshold on the Saturation channel using a strong thresholds. The helmets are designed to be more saturated in colour than the horse for visibility. A threshold of brightness could also be conducted to rule out other false positives.
   Input: Region and colour segmentation of the regions of interest
   Output: Binary view of how saturated colour is

4. With the binary output from the threshold, perform Connected Components Analysis to give a label to each of the highly saturated regions.
   Input: binary frames
   Output: labelled regions within the binary frames

5. For each CCA region in the frame, perform a measure of circularity to further rule out any false positives e.g. The jersey of the jockey is a similar highly saturated colour as the helmet. Drop any CCA regions which have a poor return for the circularity metric.
   Input: Labelled binary regions
   Output: Circular binary regions (helmets)

6. Use mean shift tracking on the located helmets to track them across the scene. This is most suitable as MST treats the feature as a histogram and tracks the direction in which the histogram distribution is moving throughout the scene.
   Input: feature, approximate location
   Output: Tracking by a measure of the direction in which the histogram of the feature is moving throughout the scene

3. (a) **[APPLICATION QUESTION]** Using edge detection describe how you would find the road markings (i.e. the white and yellow lines) in a video from a moving camera giving images such as those in the scenes below. Discuss when your approach might give false positives and what you might do to deal with that. Your solution must consist of a series of computer vision techniques and you must provide details of how the techniques will be applied including expected input and output for each technique.
[25 marks]

2020 - Application

Dé Máirt 2 Nollaig 2025 17:03

1. (a) **[APPLICATION QUESTION]** Given a video of a tennis match taken from a camera which is not moving (with a view such as that shown in the Complete Image of the court below) describe how you would locate the place on the court where each ball bounce occurs and show the locations on the plan view (as shown below). You are provided with the pixel coordinates of the four corners of the court. You may assume that the tennis ball will always be at least 10 pixels in diameter and will contrast with the court. Your solution must consist of a series of computer vision techniques and you must provide details of how the techniques will be applied including expected input and output for each technique.

[30 marks]

COMPLETE IMAGE OF THE COURT

PLAN VIEW

TYPICAL IMAGE OF THE BALL IN MOTION

2. (a) **[APPLICATION QUESTION]** Describe how you would automatically locate any traffic cones in images such as those shown below. Your solution must consist of a series of computer vision techniques and you must provide details of how the techniques will be applied including expected input and output for each technique.

[20 marks]

1. Threshold the image on the **Luminance Channel** to separate out the very bright white of the centre of the traffic cone from its darker surroundings (road). This may produce false positives in the white clouds or glare from windows. This produces a binary image with objects of high luminance in the foreground and objects of low luminance in the background

2. To eliminate the false positives, perform **Connected Components Analysis** on the binary image. This gives an identification to each of the binary regions, noting where regions are connected. **Count Pixels** in each CCA area and discard any areas which contian egregiously more or less pixels than the median amount.

3. To confirm that a traffic cone has been found, take a sample of the pixels directly above and directly below the valid CCA area's which are the white portion of the cone. If the intensities of these above-and-below areas have a high amount of Red, a medium-high amount of Green and a low amount of Blue when **thresholded across the three RGB channels** then a traffic cone has been located

Alt. It is also possible to take a large sample of images of traffic cones under different circumstances (daytime, nighttime, rain, snow) to build up a dataset of traffic cones. **Histogram back projection** can be performed on target images, producing a greyscale image, displaying where the most likely locations of traffic cones are. This gresycale image can be further **thresholded** to extract a more rigid interpretation. This approach may fail if there is a significantly orange coloured object in the scene as it does not consider size like Steps 1-3 do. If there were a firetruck or a brick wall in the scene, this may return a higher than expected probability on the traffic cone histograms.

3. (a) **[APPLICATION QUESTION]** Describe how you would automatically determine the speed of rotation of any wind turbines visible in a video from a camera which is not moving. You may assume that the wind turbines face the camera directly such as in the frames shown below. in images such as those shown below. Your solution must consist of a series of computer vision techniques and you must provide details of how the techniques will be applied including expected input and output for each technique.

[30 marks]

1. The large vertical pole should not be considered when determining how fast the turbine is spinning. A **Median Background Model** can be used to remove this static area from consideration. This produces a binary image where moving pixels are in the foreground and static pixels are in the background by aggregating the results of previously seen frames.

2. Threshold on the **Saturation Channel** to separate the white turbines (absence of colour) from the blue sky and green trees. This gives a binary image of the foreground and background. The sky should have already been removed by Step 1, but things like birds flying by/dark clouds can be eliminated by this threshold.

3. Perform some cleaning on this binary image. **Open** the image with erosion and dilation to remove noise and break apart close together segments. **Close** the image with dilation and erosion to ensure the blades of the turbine are not destroyed.

4. Perform **Contour Following** with **4-connectivity** on the cleaned binary image to extract the boundaries/edges of the wind turbine blades.

5. Take a reference image of the turbine blades and keep a note of where the contour following says the blade are in space. **Counting pixels** can be done until the turbine blades reach the approximate position again. (Multiply this bt 3 as there are 3 turbine blades).

6. To match the turbine blades position to subsequent video frames, **Chamfer Matching** can be used with the exisiting edge map. This provides a measure of similarity, and therefore the frame with the highest Chamfer value can be used for the calculations described in Step 5.

2020 - C&C

Dé Máirt 2 Nollaig 2025 17:43

1. (b) **[COMPARE & CONTRAST QUESTION]** Compare and contrast:

- Gaussian Smoothing
- Median Smoothing
- Opening

You must provide a list of the differences and similarities between the techniques. Each of the differences and similarities must be clearly explained. **NOTE:** Marks will only be awarded for the detailed comparison of techniques. No marks will be awarded for separate descriptions of the techniques

[20 marks]

Purpose: Each of these techniques is used to clean noisy images. Gaussian and median smoothing are used to clean RGB or greyscale images where opening is only used for binary images. Each of these techniques will destroy some integrity and detail of the image. Gaussian smoothing is used to alleviate Gaussian noise/blur. Median smoothing is used to alleviate Salt & Pepper noise, crackles of (0,0,0) or (255,255,255) RGB in an otherwise non-binary image. Opening is used to break apart close together regions in the binary image.
Inputs: Gaussian Smoothing and Median Smoothing take in an RGB/greyscale image. Opening takes in a binary image
Parameters: Opening does not take in any additional parameters. Gaussian Smoothing and Median smoothing both operate on local neighbours within the image so defining this neighbourhood impacts the output. For Gaussian smoothing, the shape of the Gaussian can also be defined
Complexity & Robustness: Each of these are very simple operations. Opening is the simplest as it considers the least amount of pixels. It only considers the white foregrounded pixels of the binary image. Gaussian and Median smoothing both consider every pixel in the image and thus are more complicated, as they must make a calculation (Gaussian/Median calculation) for every pixel in the image
Outputs: Gaussian smoothing outputs a cleaned version of the image which may destroy the cleanliness of straight lines in the image by introducing a degree of blur to these edges. Median smoothing similarly outputs a cleaned version of the image but edges are sharper and more preserved than in Gaussian smoothing. Median smoothing however may destroy some colour integrity of the image i.e. Shadows and lighting changes. Opening outputs the key structure of the binary image but with most elements of erosion e.g. Breaking apart close together regions.
Methods: Gaussian and median smoothing both work by considering the local neighbourhood of each pixel. Gaussian smoothing considers the colour intensities of the pixels as a Gaussian distribution and re-allocates the pixel colour based on this distribution. Median smoothing considers the colour intensities of the neighbourhodo and re-allocates the pixel colour based on the median value of the neighbourhood. Opening performs an erosion operation followed by a dilation operation. Erosion removes the outermost pixels of the binary foreground which breaks apart close together regions and removes noise e.g. Standalone pixels. The dilation part pads out the new foreground to preserve the original structure of the shape.

2. (b) **[COMPARE & CONTRAST QUESTION]** Compare and contrast:

- Robust object detection using a cascade of Haar classifiers.
- Template Matching.
- Principal Components Analysis.

You must provide a list of the differences and similarities between the techniques. Each of the differences and similarities must be clearly explained. **NOTE:** Marks will only be awarded for the detailed comparison of techniques. No marks will be awarded for separate descriptions of the techniques

[30 marks]

Purpose: Each of these techniques are used for recognition in images. More specifically, PCA and

Robust Object Detection are used to detect/recognise faces in images. Template matching is used for comparing things against some golden standard e.g. In manufacturing electronics. PCA can also be used for facial reconstruction from limited data.
Inputs: Each of these techniques takes very different kinds of outputs but all rely on having some unseen image as input to conduct the recognition on.
Robust Object Detection also inputs a large number of positive and negative classifiers which it will use to create strong and weak classifiers for the face.
Template matching takes in the source image which the recongition will be done on and the target image which is the golden standard.
PCA takes in several face images, treating them as an n*n vector. Such that it constructs a data matrix D of many faces, with each row in the matrix is a face image vector.
Parameters: Template matching must include some degrees of freedom in practice, to account for slight changes in size, orientation, particularly when considering video frame inputs.
Template matching only uses one sample to compare against.
PCA and Robust Object Detection both use many faces/classifiers to make a decision.
Complexity & Robustness: Template matching is one of the strictest forms of matching and therefore is not very robust to drastic changes, unless signficant degrees of freedom are introduced, which impacts performance.
Robust object detection is robust at detecting faces, and is not as complex as PCA, as non-matches can be pruned early in Robust object detecton if they fail one of the early strong classifers. Robust object detection cman fail sometimes, when a face is not fully forward facing or there is a lot of hair in front of the face, etc.
PCA is extremely complex and robust, it considers an enormous amount of data to begin with and must translate this into the reduced dimensionality PCA space, but it is extremely good at idenifying unseen faces.
Outputs: Robust object detection outputs a bounding box around the idenified forward facing faces. Like PCA, it has the capacity for multiple output.
Template matching does not output multiple, it selects the maxima from a 2d array of candidates which meet the match criteria above some threshold.
PCA selects the eigenvectors from the reduced dimensionality dataset which have the largest eigenvalues. Similarly choosing a maxima as Robust object detection does through non-maxima suppression and how Template matching does through the 2d array.
Methods:
PCA treats each face image as a vector and gathers many samples of faces into a data matrix D. The mean centred data of D is calculated which reduces the dimensionality which is not needed for recongition. The covariance matrix is computed and the eigenvalues and eigenvalues are calculated. The eigenvectors are normalised. The eigenvectors with the largest eigenvalues are chosen as the most reliable generalised data which is used to conduct the recognition task.

Template Matching looks over every pixel in the source image and generates a score for how well the match criteria is met. The match criteria being based off the features of the target image. The maxima of this 2d array of values is considered as the strongest match. Unlike PCA, all detail of the image is maintained. And like Robust object detection, it passes over every pixel

Robust Object Detection uses the large number of positive and negative classifers to create strong and weak classifiers which are cascaded together. Only images which pass through one classifers are evaluated against the next, making the algoriuthm very efficient. Classifiers are simple (Haar-like) and compre the general brightness of one area of the image to the surrounding area. This passes over every location in the image, which may cause some different locations in the image to fire a detection based on the cascade of classifers. This is what allows the detection of multiple faces but hte same face can have multiple detections. Non-maxima suppression is used to accumulate votes for how true the recognition is and suppress and classifications that overlap, such that only one strongest classification remains

3. (b) **[COMPARE & CONTRAST QUESTION]** Compare and contrast:

- Non-Maxima Suppression as used in first derivative edge detection.
- Non-Maxima Suppression as used in the first stage of SIFT.
- Non-Maxima Suppression as used in Moravec corner detection.

You must provide a list of the differences and similarities between the techniques. Each of the differences and similarities must be clearly explained. **NOTE:** Marks will only be awarded for the detailed comparison of techniques. No marks will be awarded for separate descriptions of the techniques

[20 marks]

Purpose - Compare:
All of these techniques use non-maxima suppression to generate sharper, more specific outputs. The purpose of this is that the outputs of first derivative edge detection, SIFT and Moravec corner detection are more robust and less noisy.
Inputs:
NMS as used in first derivative edge detection inputs the gradient image which is produced from the edge detection technqiue.
First derivative edge detection e.g. Roberts edge detection can be noisy if used on a greyscale image, producing an edge likelihood for each pixel in the image.
NMS as used in the first stage of SIFT inputs the scales of the image which SIFT is being applied to. These are the same image at different levels of Gaussian blur.
The goal of SIFT is to identify keypoints which are obvious across all scale spaces, which requires suppressing the non-persisting features.
NMS as used in Moravec corner detection inputs the cornerness map which is produced from the corner detection technique, similar to the preprocessing required in the case for first derivative edge detection.
Similarly, Moravec corner detection assigns a cornerness value to each pixel in the image, indicating how likely that pixel is to be a corner.
Outputs:

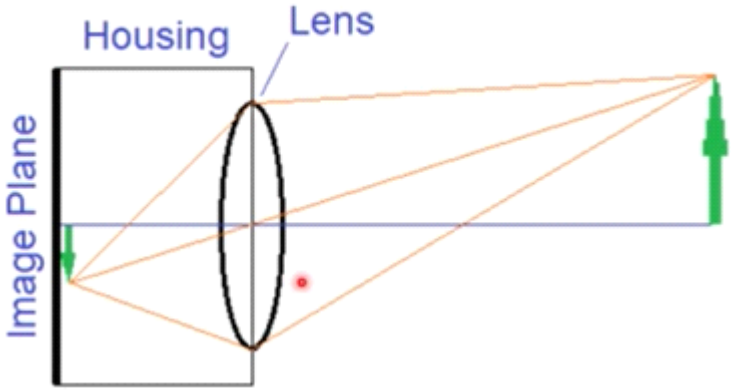
Images – Camera Models, Digital Images

Thursday, September 18, 2025 8:44 AM

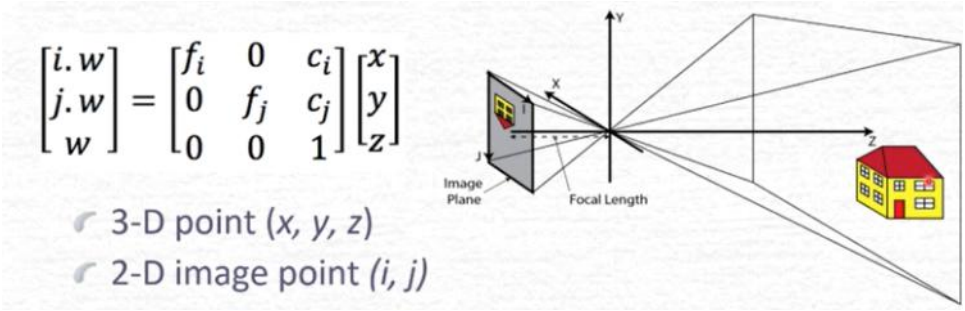
Camera Models
Digital Images
Colour Images
Noise
Smoothing

Camera Models:

Components: a photosensitive lens, a housing, a lens

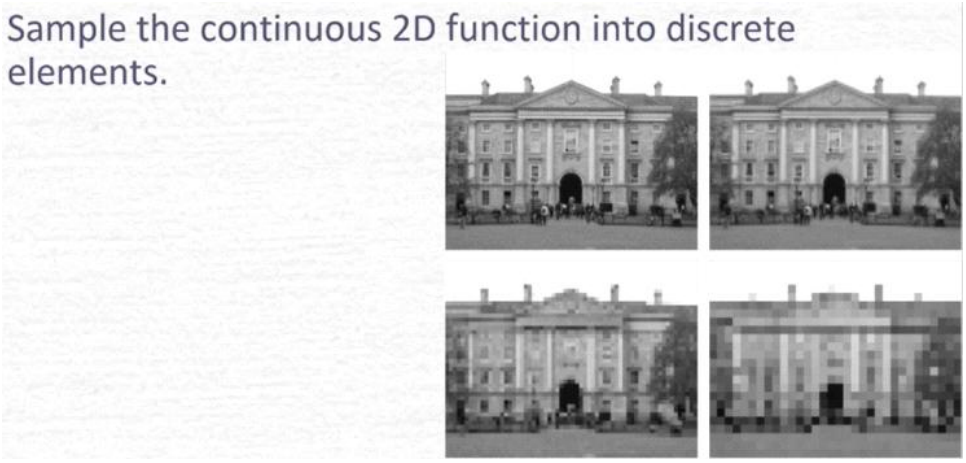


Mathematical model needed
The simple pinhole camera model – distortions



X,y,z needs to be related to i,j
Need to deal with the optical centre in the translation

Digital Images
Images are continuous 2D functions of reflected scene brightness (i,j)
Image has to be sampled at discrete locations
Has to be quantized into values – no continuous numbers



Sensor is a 2D array of photosensitive elements, in between there are non-photosensitive gaps – each photosensitive element is not infinitesimally small.
Issues: elements have a fixed area, gaps

Sampling:
How many samples do we require?
Too many – wasted space and computation time – use an appropriate resolution
Needs to be enough for the objects/application of interest
Open CV to change an image into a different size image:

```
Mat image, smaller_image;
resize( image, smaller_image,
                        Size( image1.cols/2, image.rows/2 ));
```

Quantisation:
Represent the individual image points as digital values – typically 8 bits
Any time something is changing slowly (sky colour) step will occur with lower bits – artificial contours
How many bits do you need? Don't want wasted space, less bits = less ability to distinguish
Open CV to change quantisation

```
void ChangeQuantisationGrey( Mat &image, int num_bits )
{
  CV_Assert( (image.type() == CV_8UC1) && (num_bits >= 1) &&
                                                (num_bits <= 8) );
  uchar mask = 0xFF << (8-num_bits);
  for (int row=0; row < image.rows; row++)
    for (int col=0; col < image.cols; col++)
      image.at<uchar>(row,col) = image.at<uchar>(row,col) & mask;
}
```

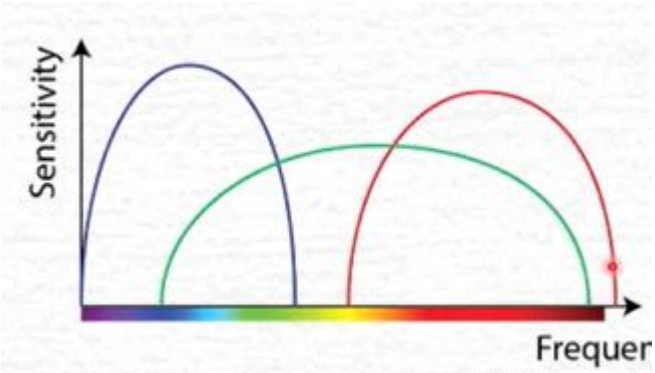
Images – Colour Images

Thursday, September 18, 2025 8:58 AM

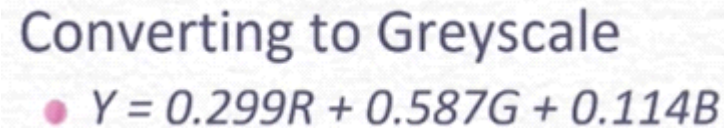
Luminance only – simple representation – humans can understand

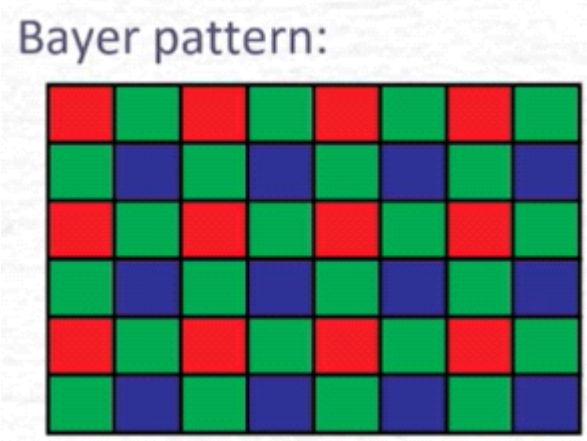
Colour images provide luminance + chrominance
Multiple channels (usually 3, RGB)
16.8Million colours (each channel 256 values)
More complex to process
Facilitate certain operations

RGB
Most common



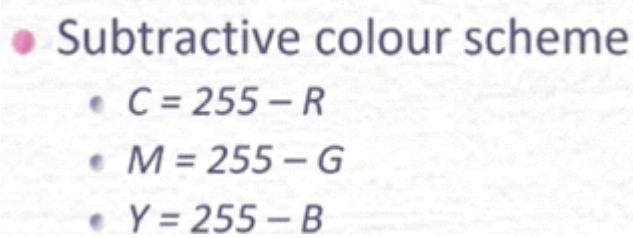
Converting to Greyscale: weighted sum of RGB for each pixel





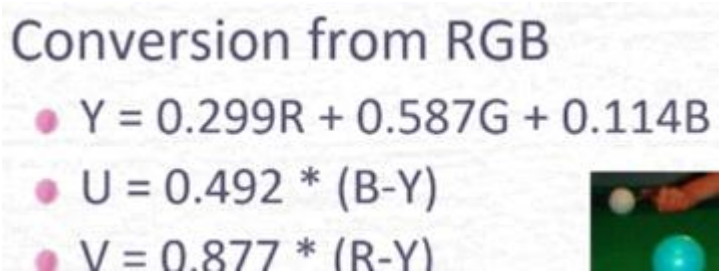
Sometimes colour is sensitive to all visible wavelength

CMY
Secondary colours – inverted RGB

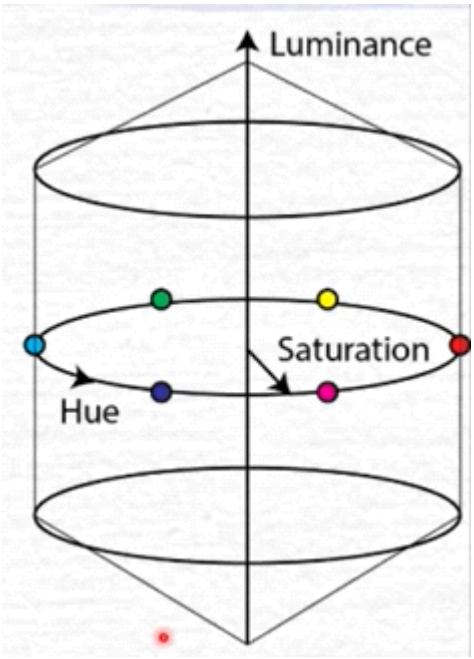


Often used in printers

YUV
Used for analogue television signals
Luminance and chrominance components



HLS – Hue Luminance Saturation
Values humans can relate to



Hue 0-360 degrees
Luminance 0-1(255)
Saturation 0-1(255)
Watch out for circular hue

HLS problems: very low or very high luminance – close to black or white – hue and saturation values are almost meaningless. Must check luminance first.
Very low saturation – hue is meaningless – noise

Images – Noise & Smoothing

Thursday, September 18, 2025 8:58 AM

Noise degrades the image – need to ensure it doesn't impact processing in an extreme way.
Don't want to take two images of the same scene and get two different results

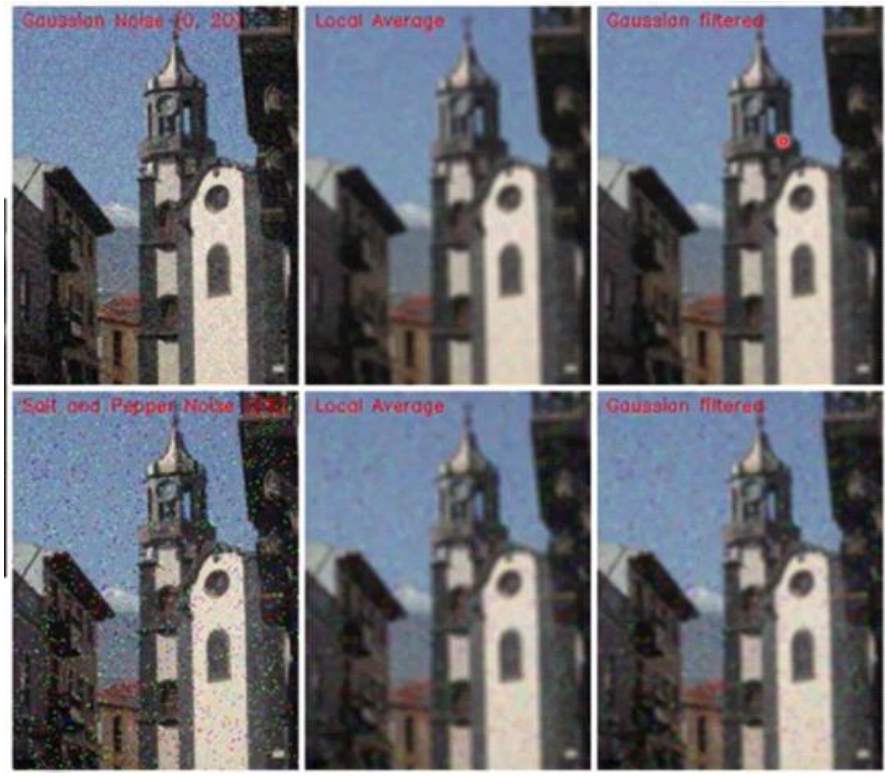
Noise can be caused by the envrionment, device, electrical interference, digitisation, transmission.

Measuring Noise: signal received/noise received
Gaussian Noise & Salt+Pepper Noise

Gaussian Noise: a value is being added or subtracted to a pixel in a Gaussian distribution
Salt + Pepper noise: black or white points in the image

Smoothing: look at the surrounding pixels to determine.
Linear and non-linear smoothing to attenuate the noise.
Can't be sure what is and isn't noise!
Local Averaging, Gaussian Smoothing, Median Filtering

Local Average - take a weighted average of the neighboring pixels
Gaussian Smoothing - change the weights to mimic a Gaussian distribution (center). May use a larger filter. Want to suppress noise but don't want to damage edges



Median Filtering - use the median value. Use if there is salt+pepper noise. Does not blur *edges* that much. Can be applied iteratively. Damages very thin lines and sharp *corners*. Used to be computationally expensive. More efficient now. People don't tend to use.

Use gaussian filter for gaussian noise

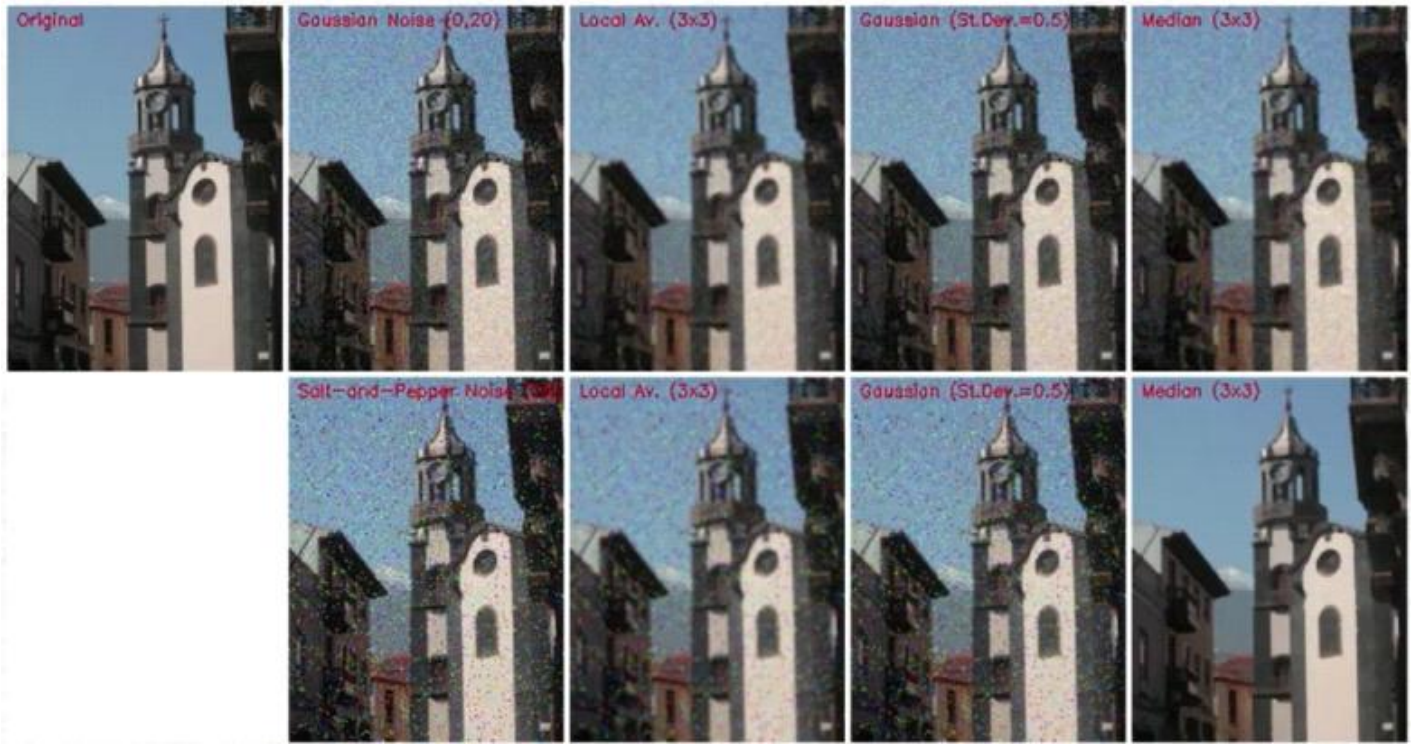


Image Pyramids

Processing at different scales. We don't understand the size of objects in advance.

To process images at multiple scales efficiently.

Smooth image (often Gaussian) and subsample (usually by a factor of 2).

Images Minitest

Thursday, September 18, 2025 4:24 PM

TECHNIQUES - IMAGES

Déardaoin 4 Nollaig 2025 18:28

Technique	Input	Output	Method	Purpose	Other
Threshold Red					
Threshold Green					
Threshold Blue					
Threshold Hue					
Threshold Luminance					
Threshold Saturation					
Local Averaging					
Median Filtering					
Gaussian Smoothing					

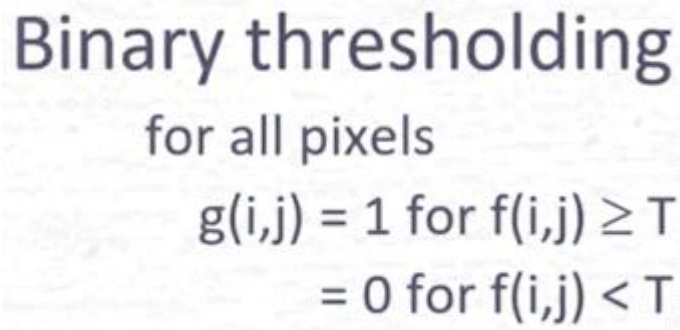
Binary Image Processing

Thursday, September 18, 2025 8:59 AM

Pixels are either of interest or not of interest e.g. license plate – black areas more important than white areas.

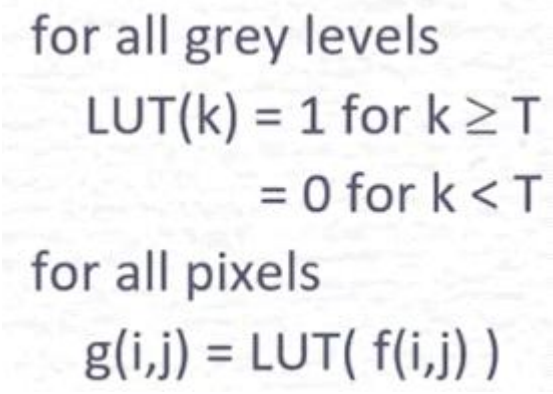
Binary image processing is very very fast – could be done on old computers

Thresholding – greyscale to b&w



T is some predefined threshold.
Simple scenes only

Look Up Table: 1D array for every possible greyscale – efficiency

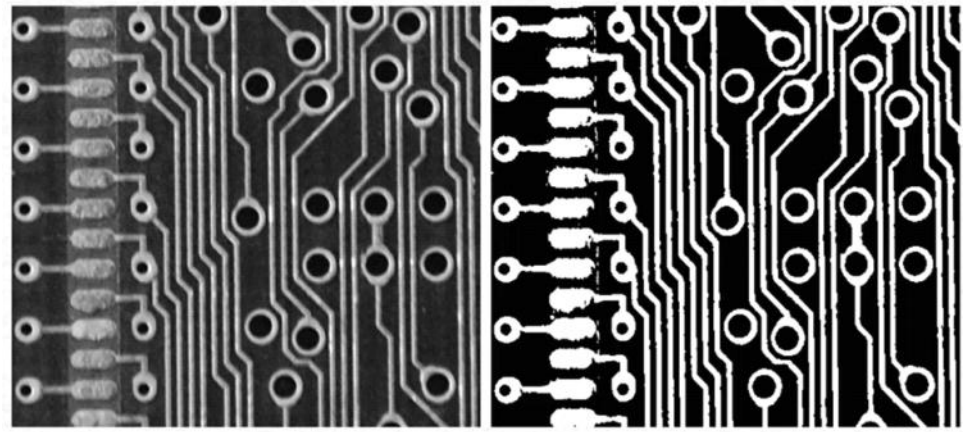


Separate objects of interest from background

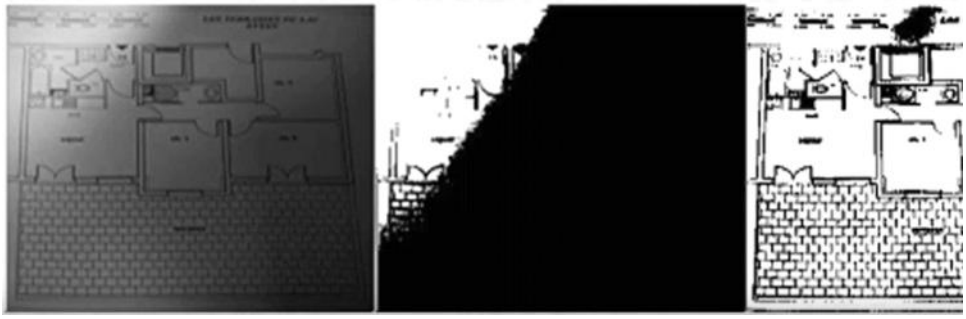
How do we determine the best threshold?
Manual setting – use slider
Issue of changing lighting – threshold will fail quickly
Bulbs degrade over time
Not feasible.

Threshold Detection – Ostu Thresholding:
Minimise the spread of pixels – smallest variance within the object of interest pixel values & background values
T can only take 1 of 256 values.
Sum weights within normalised distribution
Smallest within class variance = largest between class image

Threshold Detection – Otsu Thresholding



Variations – Adaptive Thresholding
Divide into sub-images to deal with lighting changes

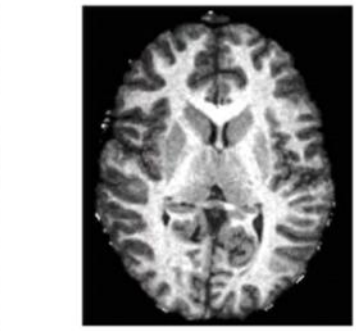


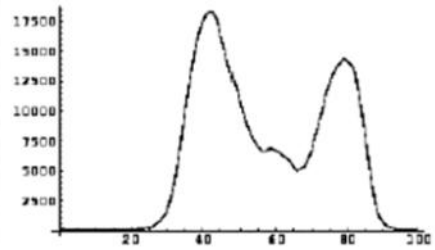
Compute thresholds for all subimages
Interpolate between thresholds of subimages

Whole Image:

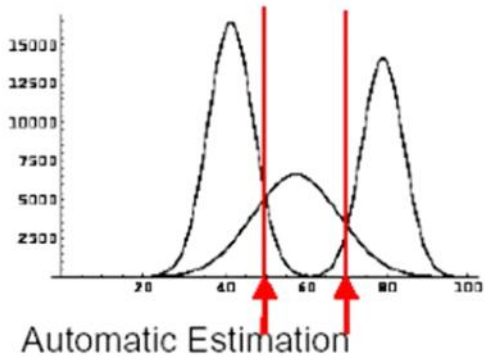
Open CV on a greyscale version: compares each pixel to the average of all the pixels around it.
Compare with an offest to avoid noise/strange effects.
Have to do tuning on the site of the region is appropriate.

Multi-Spectral Thresholding
Thresholding should only be applied where we can very distinctly separate the objects we're interested in





MRI scan (3D GRE) (top) and histogram (bottom)



Automatic Estimation of parameters for wm, gm and subcortical structures (Levenberg Marquardt).

Cleaning Binary Images

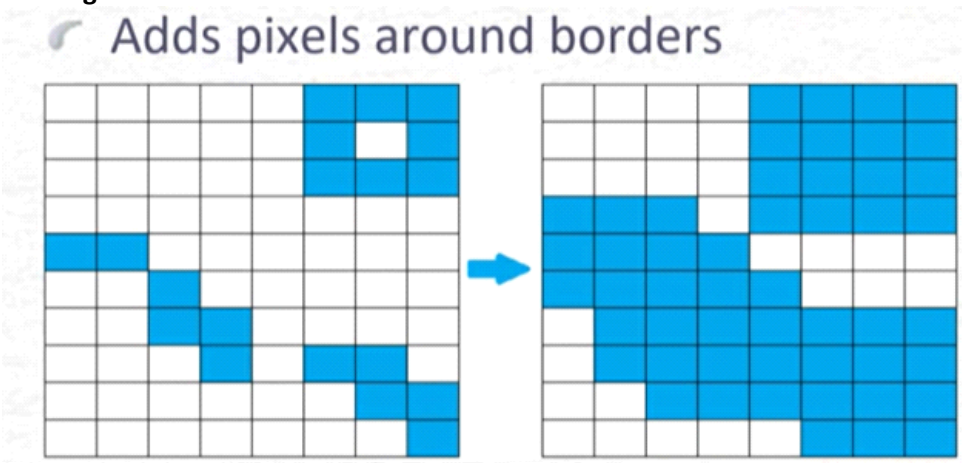
Thursday, September 18, 2025 8:59 AM

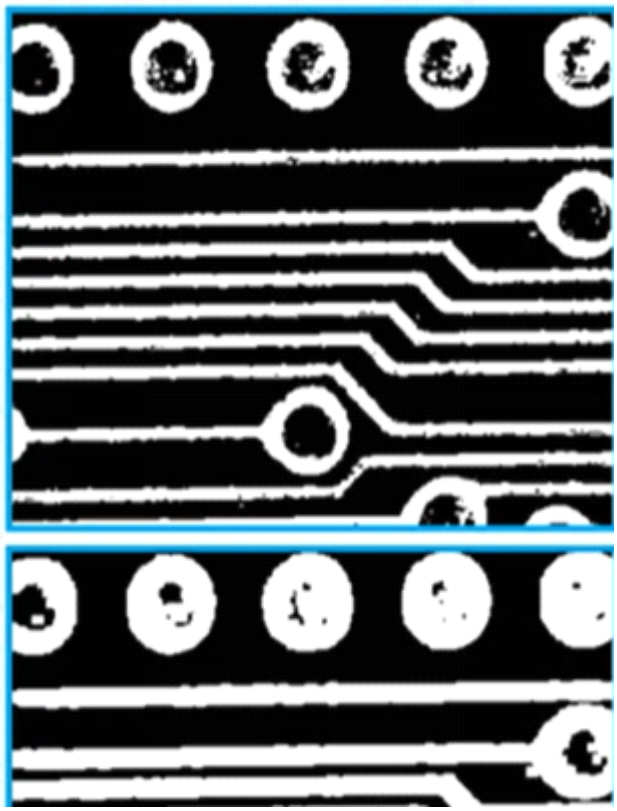
Cannot use normal smoothing operations

Need to remove noisy points – smooth binary region boundaries

Treat images as sets – erosion/dilating/opening/closing

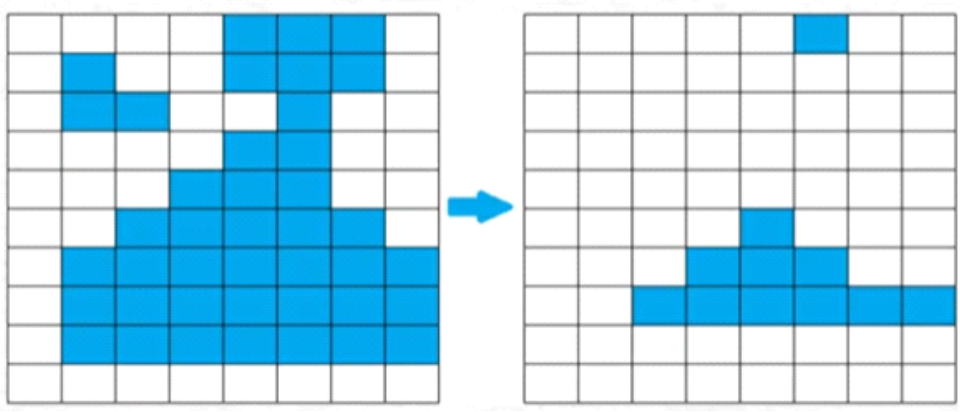
Dilating:

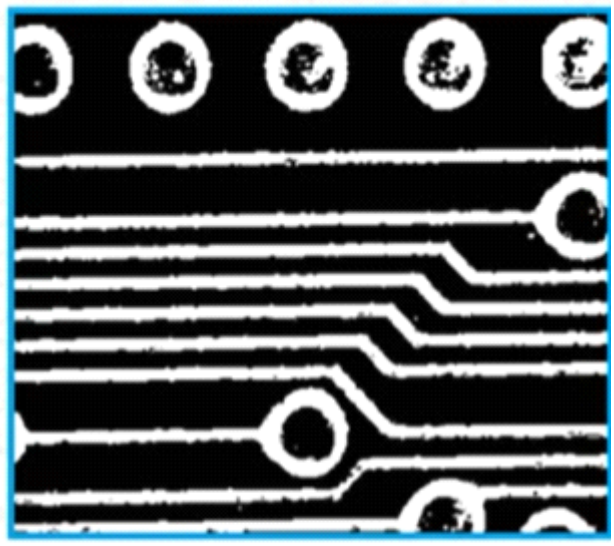




Makes regions bigger – fills small holes – joins close regions

Erosion:
Removes border pixels – NOT INVERSE OPERATION AS DILATION

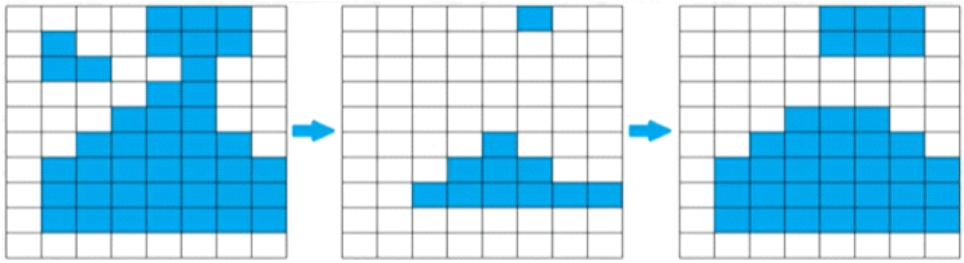




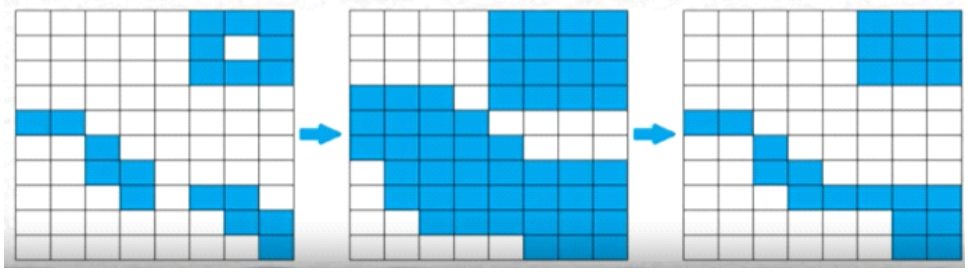
Makes regions smaller –removes noise points – removes narrow bridges

Combine Erosion & Dilation – Opening – Many benefits of erosion

Removes noise – removes narrow bridges – roughly maintains region size – smooths shape

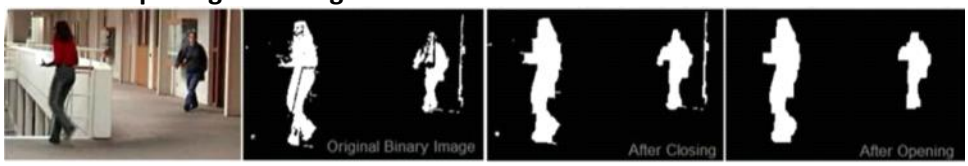


Combing Dilation and Erosion – Closing – Many benefits of dilation



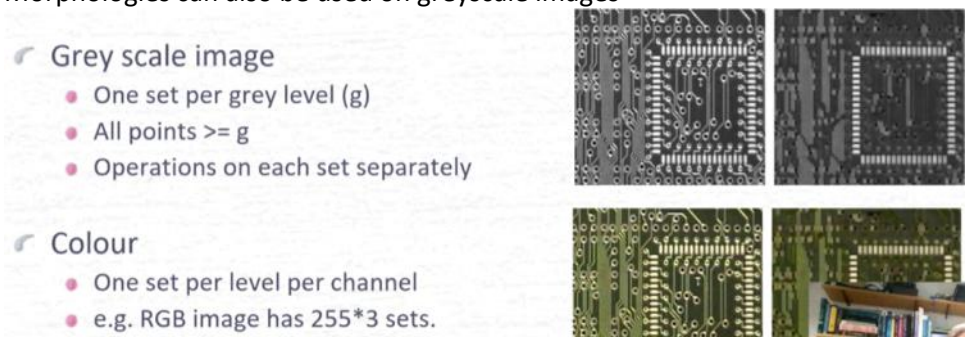
Fills small holes – joins close regions – roughly maintains region size

Combine Opening & Closing



Isotropic Structuring Element – removes small image details
Opening and opening and opening changes nothing
Closing and closing and closing changes nothing
This is NOT true for erosion and dilation

Morphologies can also be used on greyscale images



Per set or per set per channel

Local Maxima
Dilate and compare to the original image (to find local maxima)
Threshold the original image
Logical AND the results together – until found exactly the same value in the dilated image as in the original image

TECHNIQUES - BINARY

Déardaoin 4 Nollaig 2025 18:38

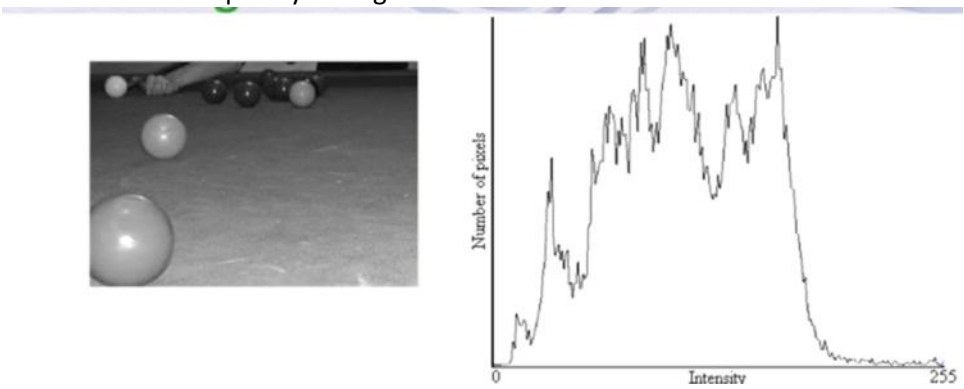
Technique	Input	Output	Method	Purpose	Other
Otsu Thresholding					
Adaptive Thresholding					
Erosion					
Dilation					
Opening					
Closing					

Histograms

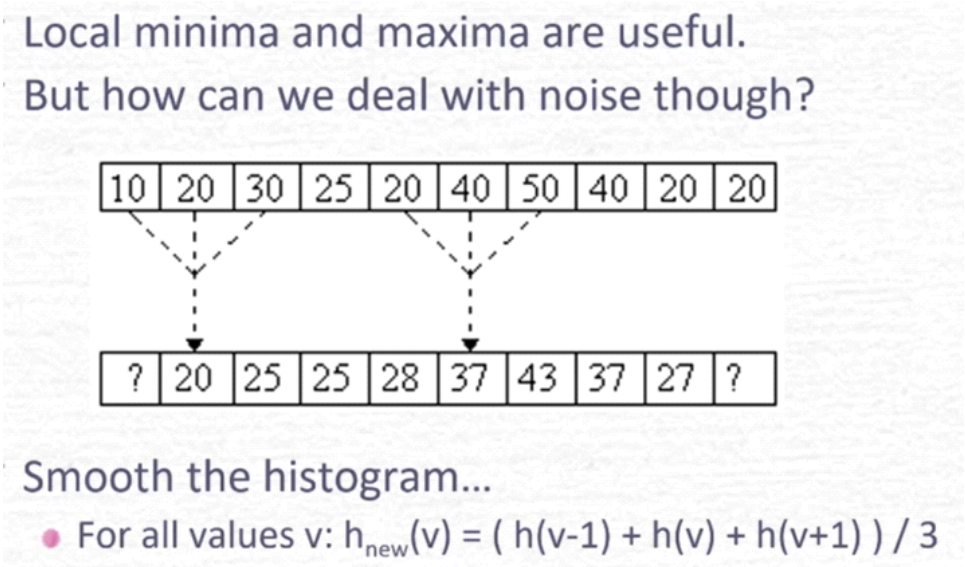
Friday, September 26, 2025 11:14 AM

Brief summary of image content- not where pixels are in the image
Analysing colours & greyscales

1D Histograms
Determine the frequency of brightness values

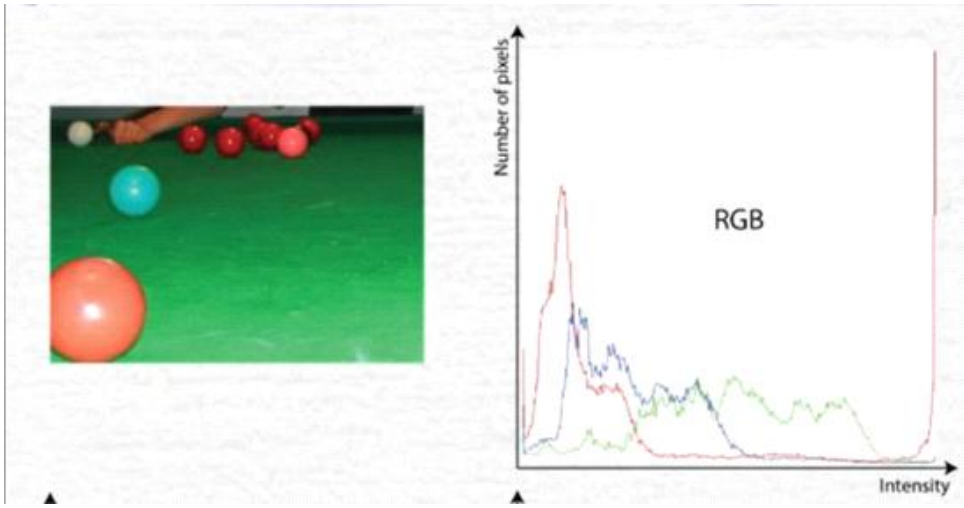


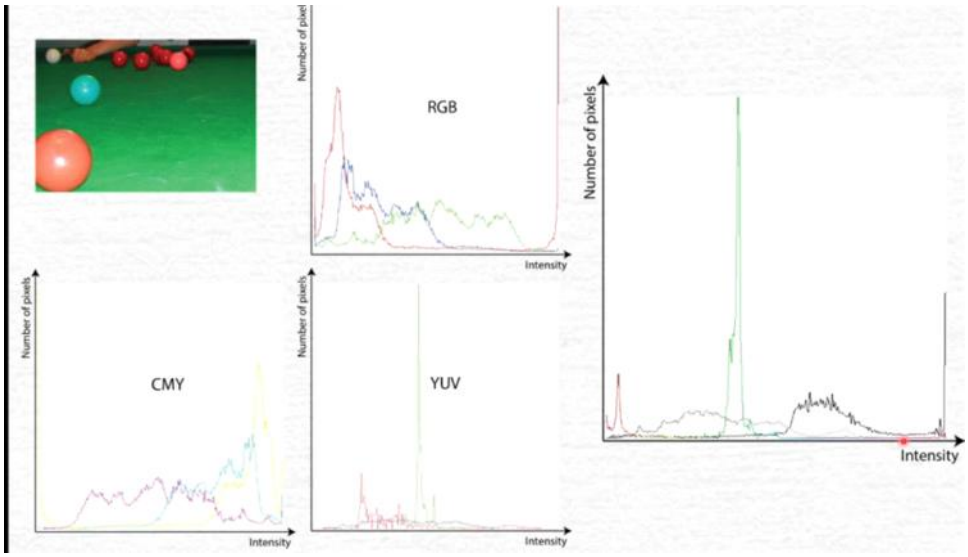
How many pixels have a particular greyscale
Initialise h(z) to 0 for all 256 values of z
Add 1 to all pixels (i.j): h(f(i.j))++
Global information from the image- can be used on the segments of the image
Histograms are not unique – works for jumbled up images
Local minima and maxima are useful – but histograms are very noisy
Smooth histogram with local averaging



Smoothed values takes the local averages – but ends are ?

Colour histograms – computing independently for each channel





Create an array of histograms in opencv
Processing every channel separately
Channels are NOT independent

3D histograms

Better discrimination comes from considering all channels together
Reduce quantisation – reduce number of cells that appear in 3D histogram

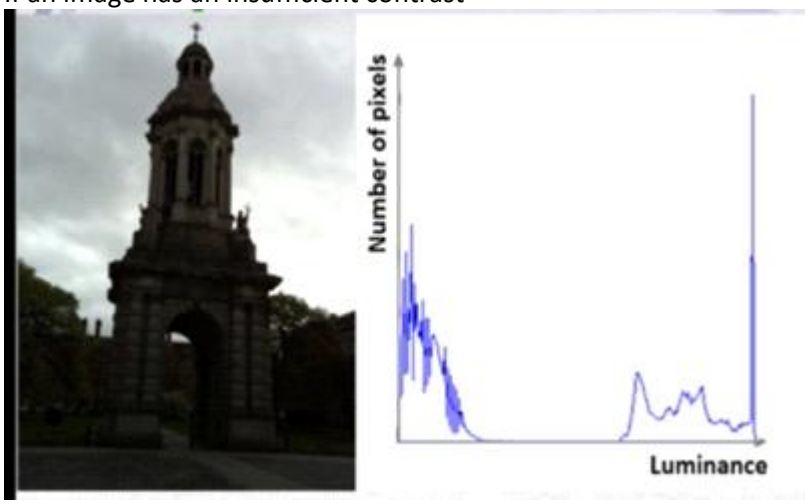
Blacker centre = weak/no response
Whiter centre = strong response
Typcially use same number of bins per channel

Histogram Equalisation & Comparison & Back Projection

Friday, September 26, 2025 11:27 AM

Equalisation

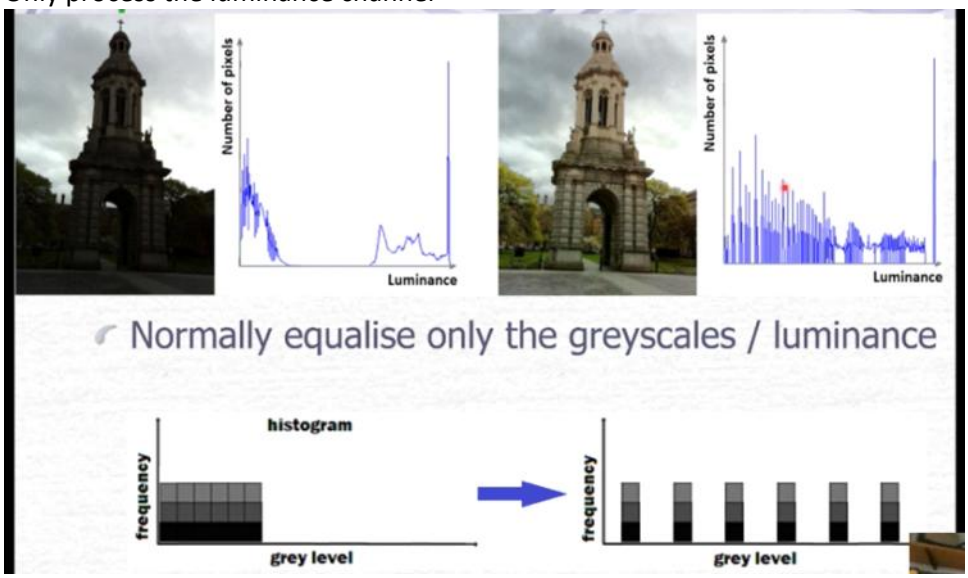
If an image has an insufficient contrast



Only really helps humans – doesn't really help the computer

How can we improve contrast – evenly distribute among channels – leads to colour distorion

Only process the luminance channel



Create a histogram is flat.

Aim for flat or normal distribution

Note we will get gaps in the resultant histogram

Spread the cells – don't take individual pixels and put them into different greyscales.

```
// Create a lookup table to map the luminances
// h[x] = histogram of luminance values image f(i,j).
pixels_so_far = 0
num_pixels = image.rows * image.cols
```

```
for  input = 0 to 255
  pixels_so_far = pixels_so_far + h[ input ]
  LUT[ input ] = (pixels_so_far*256) / (num_pixels+1)
// Apply the lookup table LUT(x) to the image:
for  every pixel f(i,j)
   g(i,j) = LUT[ f(i,j) ]
```

/

Histogram Comparison

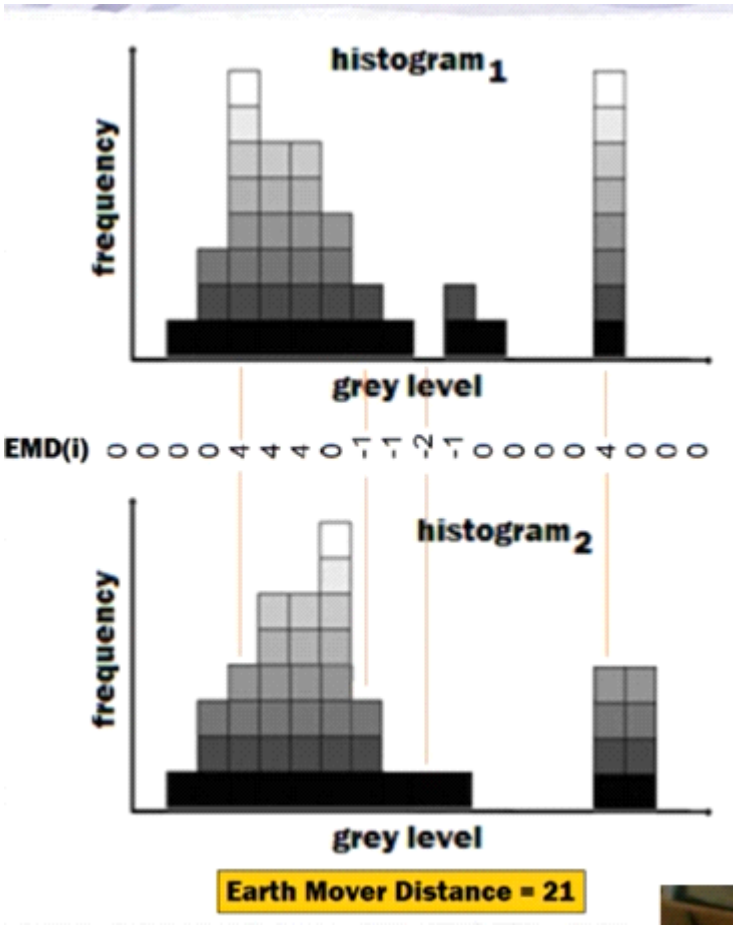
To find similar images – use metadata

One simple possibility to compare is to compare images – but not reliable i.e. blue could be the sea or the sky, not the same

Need a metric for comparisons

Not the most useful for comparing images – useful for comparing sections

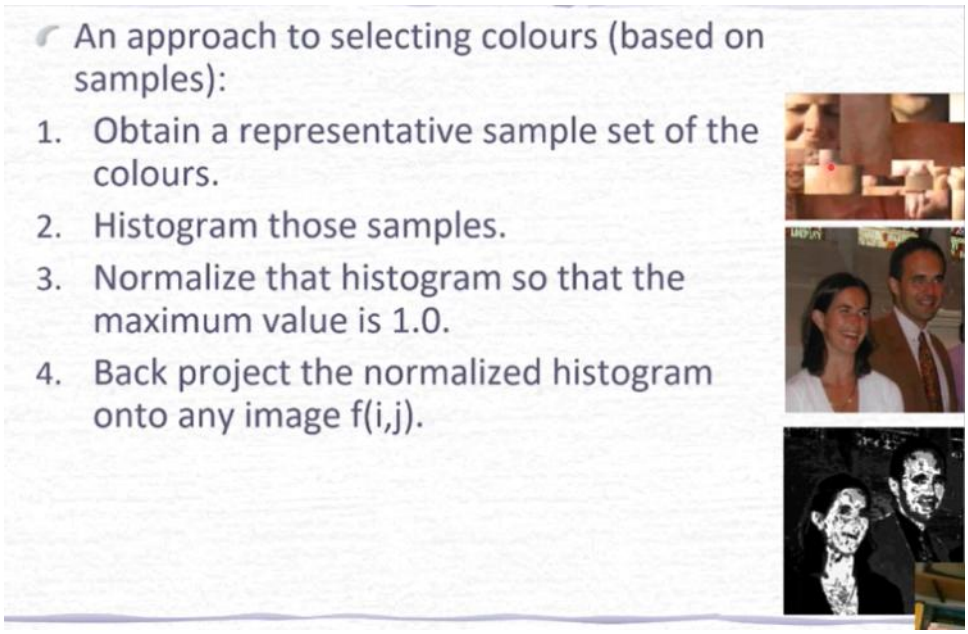
Earth Mover's Distance – minimum cost for turning a distribution into another distribution



Colour EMD is harder to compute

Back projection

An approach to selecting colours - some knowledge of what you want e.g. skin pixels
Obtain a representative sample set of the desired colours
Histogram the samples (3d) and normalise so the max val is 1
Back project the normalised sample histogram onto the image.
Gives the similarity between any pixel and the sample set.
NOT GETTING A BINARY OUTPUT – NOT A COMPLETE TECHNIQUE ON IT'S OWN



TECHNIQUES - HISTOGRAMS

Déardaoin 4 Nollaig 2025 18:39

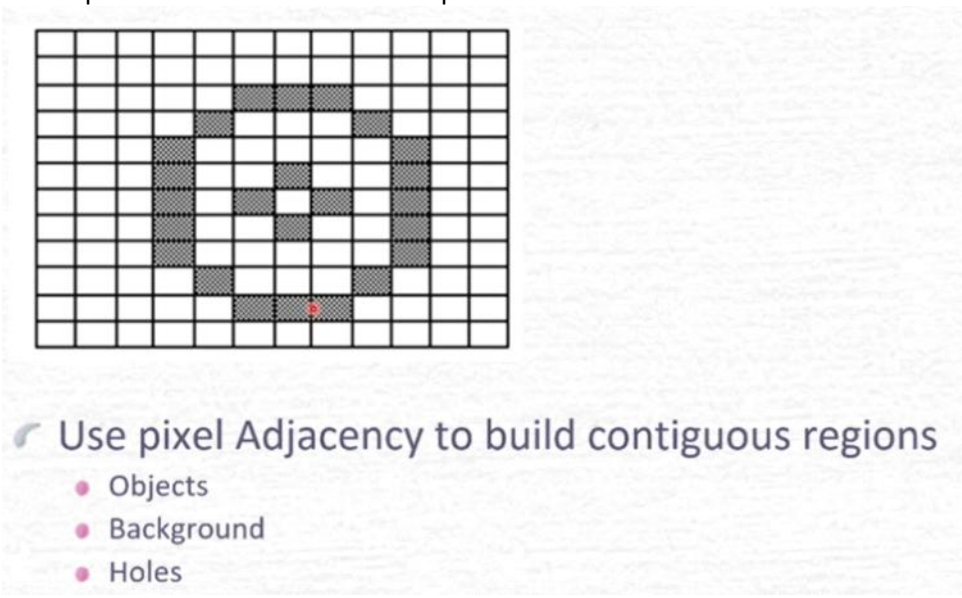
Technique	Input	Output	Method	Purpose	Other
1D Histogram					
Colour/3D Histogram					
Histogram Comparison					
Earth Mover's Distance					
Histogram Back Projection					

Region Segmentation

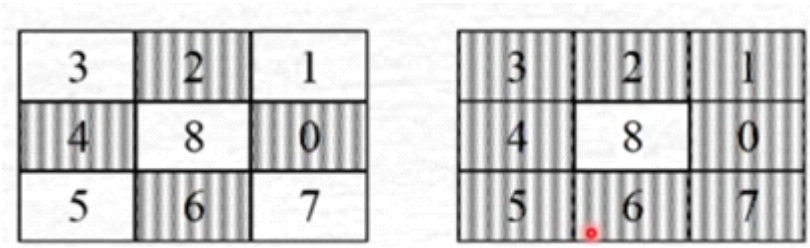
Friday, September 26, 2025 11:49 AM

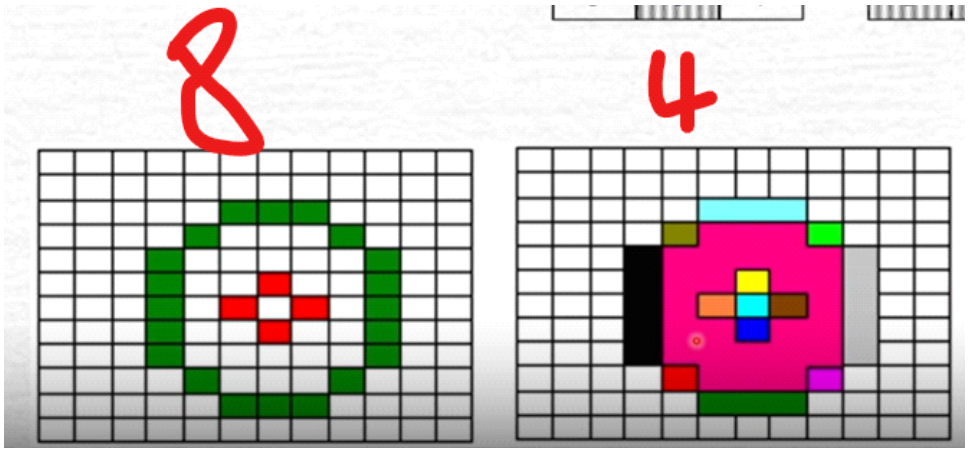
Segmentation
Split image into smaller parts
Preferably corresponding to parts of objects.
Region based or edge based
Video segmentation – break into clips – segmenting objects over time

Connectivity
Which pixels are connected to other pixels

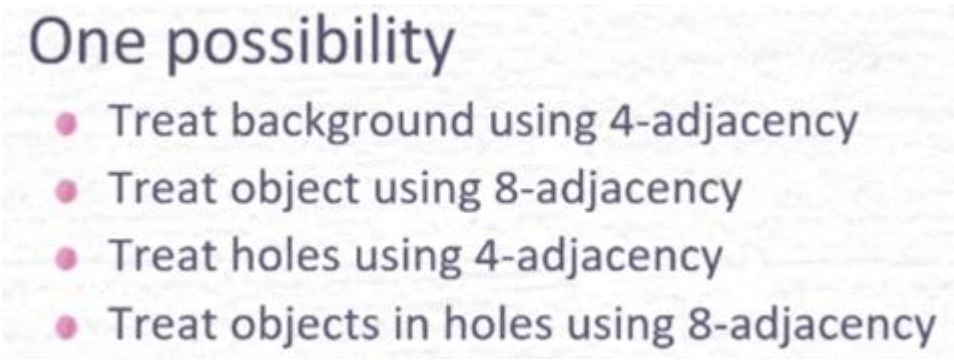


Need a single interpretation of a scene
4-adjacency or 8-adjacency

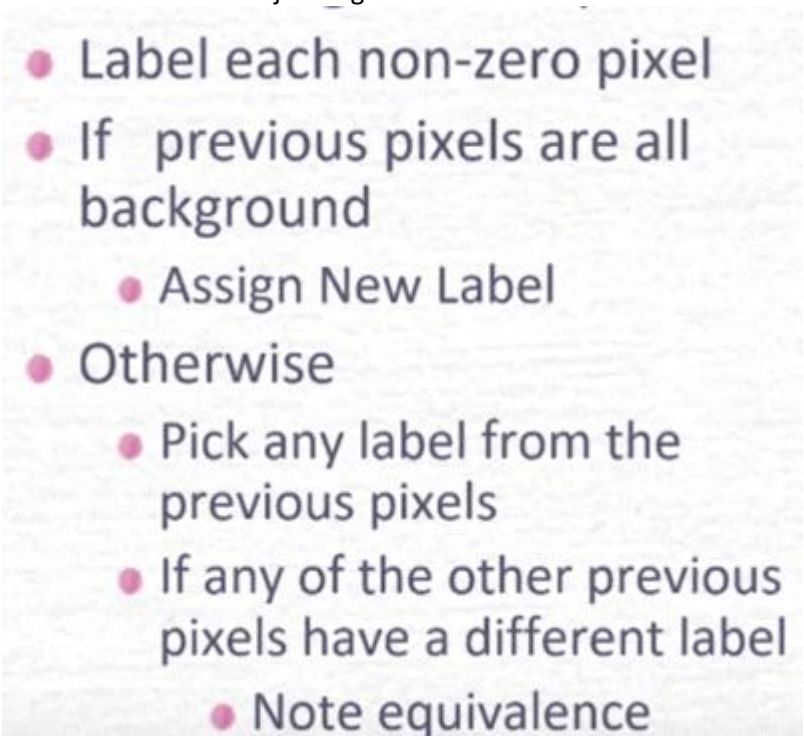


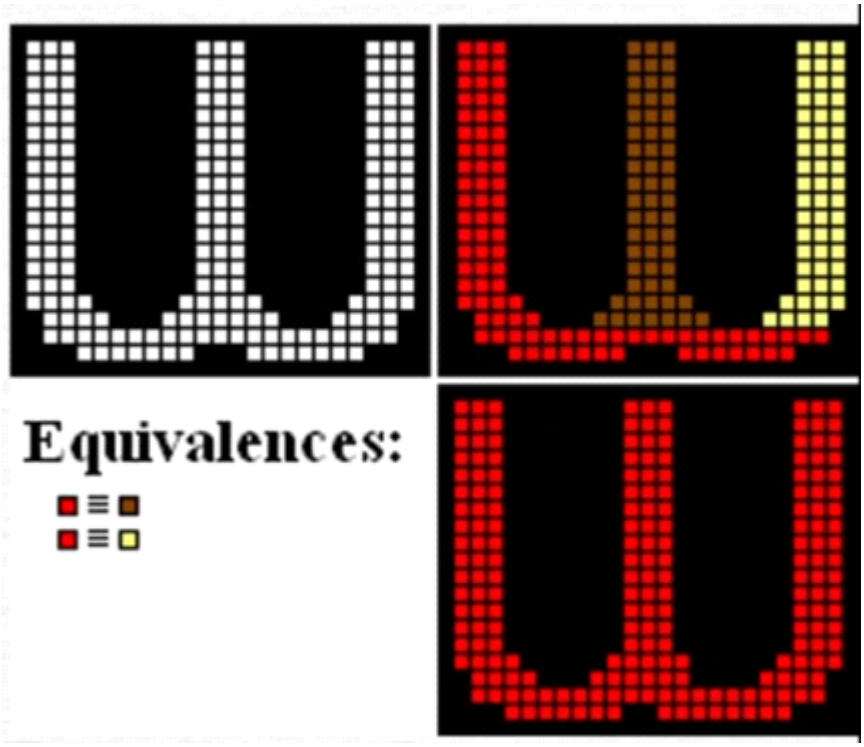


Use the connectivity techniques alternately – binary – object pixels and background pixels



Binary images – greyscale into b&w
We need to be able to join together





Relabel equivalences afterwards

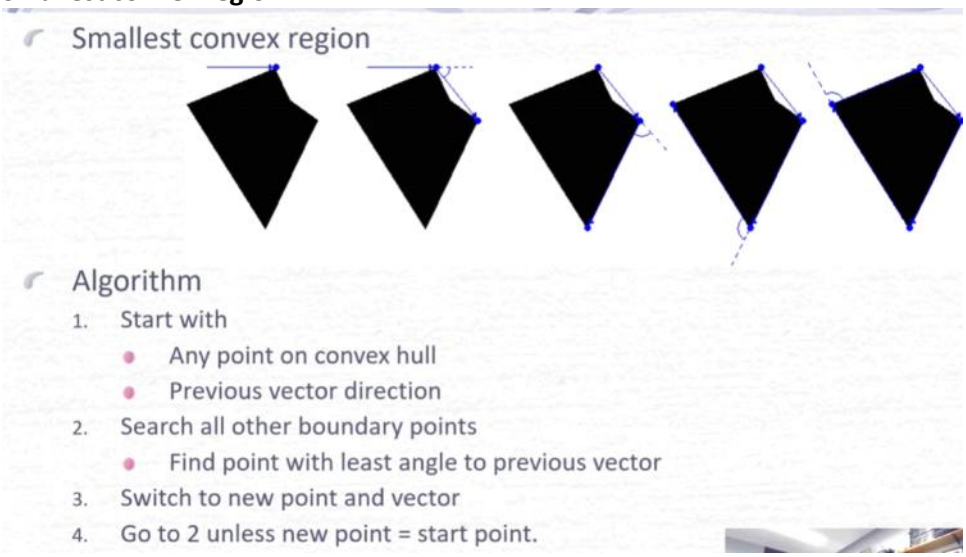
Region Featuring

Friday, September 26, 2025 12:02 PM

Given a segmented object/shape we can count the pixels in that shape and that will give us the area

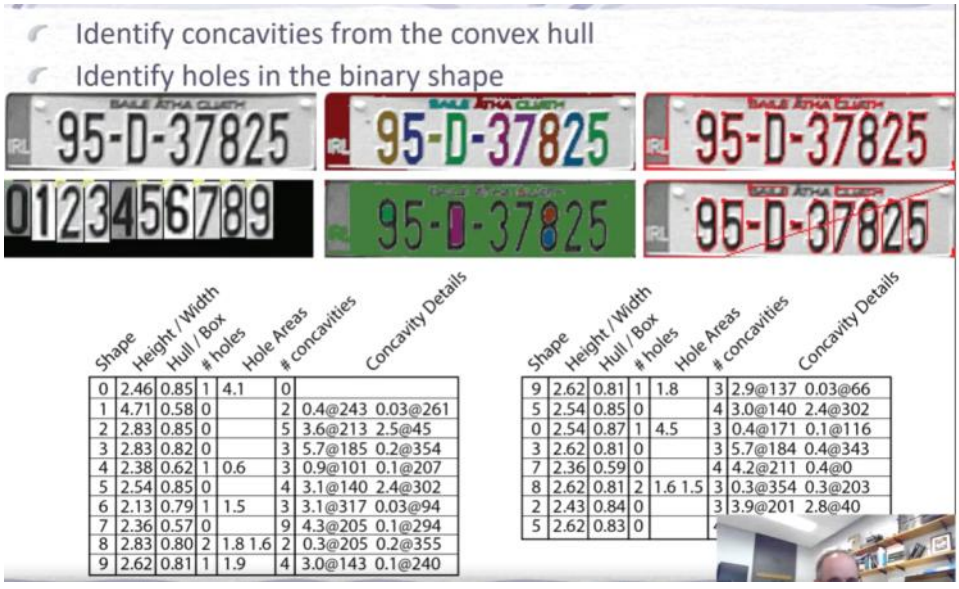
Minimum boundary rectangle
Length/width ratio
Rectangularity area/(length*width)

Elongatedness
Cannot be length and width
Smallest convex region



Moments: measure the distribution of shape

Concavities & Holes



Shape	Height / Width	Hull / Box	# holes	Hole Areas	# concavities	Concavity Details
0	2.46	0.85	1	4.1	0	
1	4.71	0.58	0		2	0.4@243 0.03@261
2	2.83	0.85	0		5	3.6@213 2.5@45
3	2.83	0.82	0		3	5.7@185 0.2@354
4	2.38	0.62	1	0.6	3	0.9@101 0.1@207
5	2.54	0.85	0		4	3.1@140 2.4@302
6	2.13	0.79	1	1.5	3	3.1@317 0.03@94
7	2.36	0.57	0		9	4.3@205 0.1@294
8	2.83	0.80	2	1.8 1.6	2	0.3@205 0.2@355
9	2.62	0.81	1	1.9	4	3.0@143 0.1@240

Shape	Height / Width	Hull / Box	# holes	Hole Areas	# concavities	Concavity Details
9	2.62	0.81	1	1.8	3	2.9@137 0.03@66
5	2.54	0.85	0		4	3.0@140 2.4@302
0	2.54	0.87	1	4.5	3	0.4@171 0.1@116
3	2.62	0.81	0		3	5.7@184 0.4@343
7	2.36	0.59	0		4	4.2@211 0.4@0
8	2.62	0.81	2	1.6 1.5	3	0.3@354 0.3@203
2	2.43	0.84	0		3	3.9@201 2.8@40
5	2.62	0.83	0			

Perimeter length:
Length of the contour
Difference between a 45 degree angle and the direction

K Means Clustering

Friday, September 26, 2025 12:02 PM

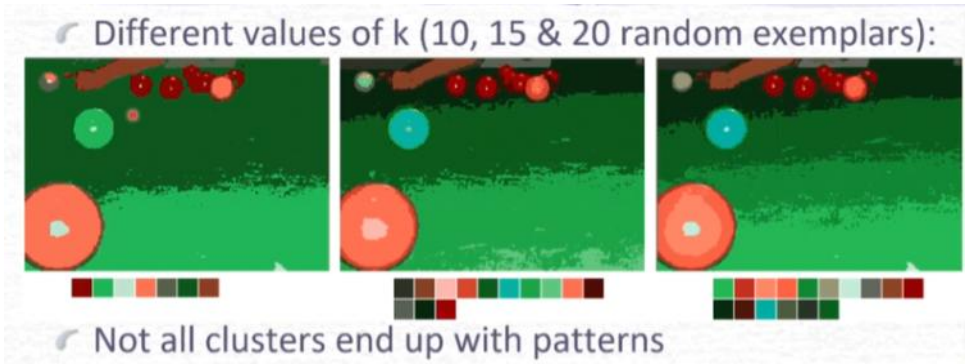
Identify significant colours in images – with concise descriptions and used for object tracking

Reduce the number of colorus in the image i.e. compression

Create k clusters of pixels - k colours – all pixels associated with a k
Unsupervised learning

Old algorithm – k must be known in advance and determine k with max confidence
Initialise k cluster exemplars either randomly OR extract exemplars from image

Allocate pixels to the closest existing exemplar – recompute exemplar



CCA – YouTube

Thursday, October 02, 2025 12:01 PM

[Intro2Robotics: Connected Components in a Binary Image]

TECHNIQUES - REGIONS I & II

Déardaoin 4 Nollaig 2025 18:41

Technique	Input	Output	Method	Purpose	Other
Connectivity/Adjacency					
Connected Components Analysis					
Minimum Bounding Rectangle					
Measure of Rectangularity					
Measure of Circularity					
Measure of Elongatedness					
Convex Hull				Identify concavities and holes	
K Means Clustering		Histogram segmentation only			
Watershed Segmentation					
Mean Shift Segmentation					

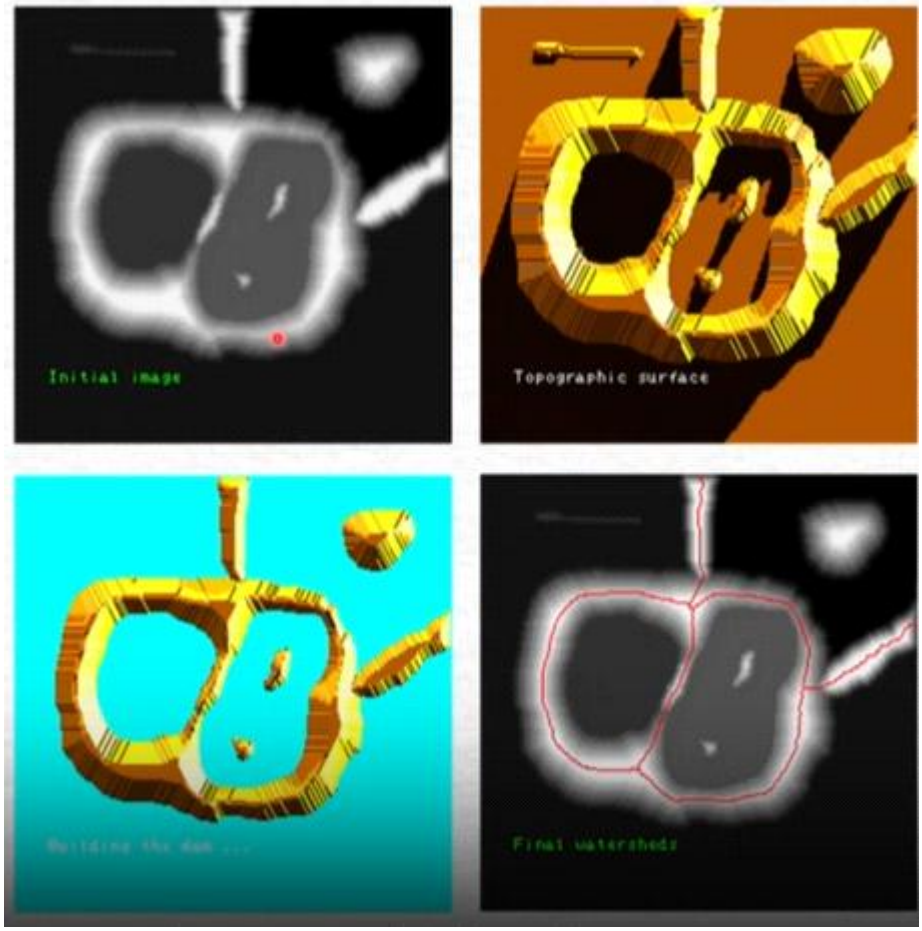
Watershed Segmentation

Thursday, October 02, 2025 11:07 AM

- **CCA** – binary image – connecting the dots into a region/shape
- **K-Means** – doesn't separate into regions but labels pixels by colour – those pixels don't have to be connected. Need to use CCA if you want to connect

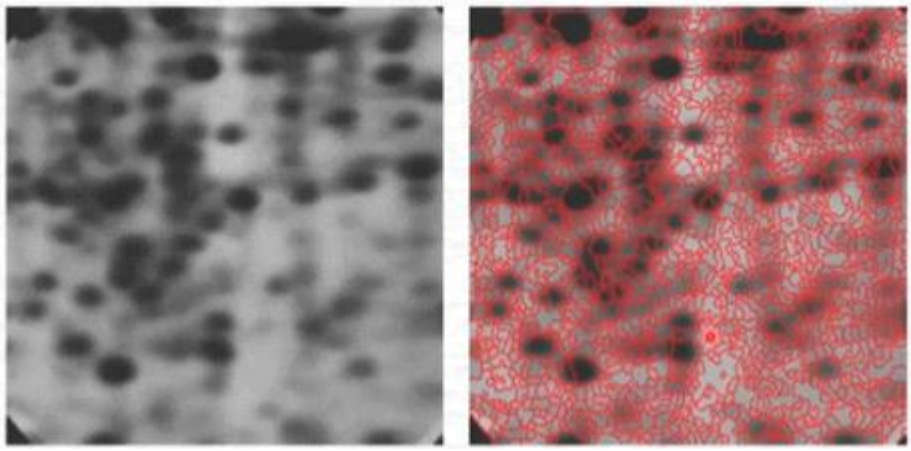
- **Watershed –** separate image into a group of regions where each region is described by points that are neighbouring
  - Identify all minima – looking for local minima withing image
  - Label as different regions – give different labels
  - Flood from the minima, extending the regions – take every grey level in turn, filling the basins
  - Where regions meet, we have watershed lines – a separation between two regions/catchment basins

Minimum what? Gradient – first derivative of image – at all of the edges/rate of change we get a response. Where there is no response – we are in the centre of a region – so start with gradients and work upwards



Watersheds were filled by the darkest region first, then next darkest region, then next.
The final watershed shows the image separated into regions

Watershed can do massive oversegmentation



You can put markers on the image and force it to segment in a more specific way. They are the starting region labels. Final output will have X regions, 1 corresponding to each label.

Small alterations to labels can give radically different answer.
Use a-priori labels to identify objects

Mean Shift Segmentation

Thursday, October 02, 2025 11:08 AM

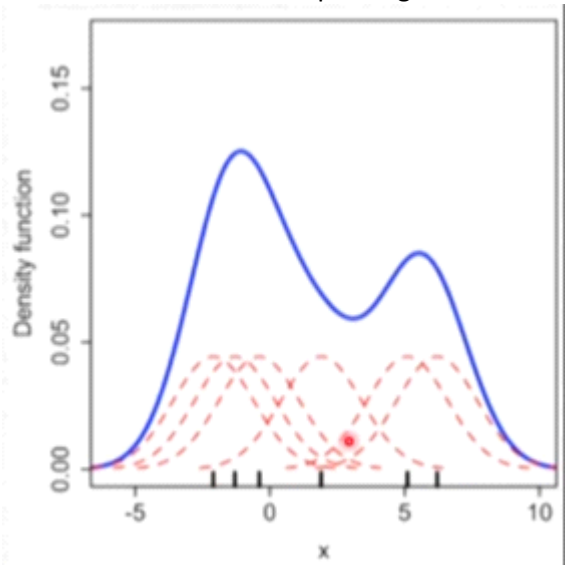
K-means clustering: requires the number of clusters to be known – takes no account of spatial location

Mean Shift segmentation:

- Do not need to know the number of clusters
- Can provide spatial AND Colour segmentation
- Goal to take each point in the scene and associate with a high-density cluster – expensive technique
- Move pixels in the direction of local increasing density
- Can do mean shift on colour and greyscale

Kernel Density Estimation:

- Given a sparse dataset determine an estimate of density at each point
- Call the samples the centre of a Gaussian - applying some normalisation gives a continuous function – gives the average density at that particular point.
- Smooth all data samples together – add them all together



Sum of kernel is 1 for any kernel

Mean shift – basic algorithm

- For each pixel – estimate the local kernel density
- Estimate the direction in which the kernel is increasing (mean shift vector) - depends on the eight directions for the pixels one pixel away from the current pixel – move there if it improves the kernel density for the current pixel - looking for similarity to a particular colour – a region similar to the colour we are at
- Shift the pixel to the new mean
- Re-compute until the location stabilises for every pixel
- Finally identofy pixels which ended up in the same location – same cluster
- Determine the local mean of those similar pixels

Local kernel density

- Limit the points included in the kernel density estimate based on distance from the current pixel and colour of the current pixel – weighting relative to the current point
- Must use a spatial kernel AND a colour kernel
- Both kernels can be Gaussian

We are interested in the rate of change of the mean shift vector to find the optimal direction to

move in

Pros & Cons:

- + do not need to know the number of clusters – compared to kmeans
- + provides spatial & colour segmentation
- - selection of kernel/parameters can be very hard
- - slow if there are a lot of clusters

Introduction to Video

Thursday, October 02, 2025 11:07 AM

Objects of interest - motion detection, moving object detection and location, derivation of 3d object properties

When is an object of interest - reasonable size, max/min velocity and acceleration
Assumption that the parts of objects will correspond from frame to frame – not going to magically change colour
Assumption that the parts of an individual object move together

Problems - lighting changes gradual/sudden, shadows, weather. **Illumination and appearance changes**
Problems – background changes if a parked car comes in and stays, does it become part of the background. Background objects that oscillate – trees/water
Problems – setup: camera motion, frame rate, field of view, distance

Static Background – background subtraction – see moving objects:
Pick empty scene and subtract it from every other frame of the video – gives the difference image
Thresholding – do we have a moing pixel object or not:
Comparing the difference to some threshold t
Too high – false negatives = things that are not flagged as moving which were moving
Too low – false positives – things that are flagged as moving but are not
High contrast is required but cannot be guaranteed
How to deal with colour? - average difference on all channels, threshold separately

Background Models

Friday, October 03, 2025 2:42 PM

Median background model:

Consider each pixel and compare current value of pixel to median value over its history (time)

TECHNIQUES - VIDEO & VIDEO II

Déardaoin 4 Nollaig 2025 18:44

Technique	Input	Output	Method	Purpose	Other
Static Background Model					
Median Background Model					
Gaussian Mixture Model					
Exhaustive Search					
Mean Shift Tracking					
Optical Flow					

Tracking

Wednesday, November 05, 2025 9:28 PM

Objects may
be undergoing complex motion,
change shape,
be occluded,
change appearance due to lighting/weather,
May physically change appearance (removing/putting on clothing)

Exhaustive search
Template matching/Chamfer matching.
Extract the object to be tracked from a frame
Use a similarity metric. Look in all possible positions in future frames.
Normalised cross correlation = template matching
Multiple degrees of freedom in search if there is no constraints on the object
Consider sizes, orientations and positions can change.
Tracking may fail if the object motion is too complex. Evaluating is difficult.
Most techniques used for tracking can be applied exhaustively but this is expensive

Mean Shift Tracking
Representing object as a histogram to track as it moves frame to frame.
Find the direction in which the distribution is moving through the scene.

- Get the template object e.g. a person. We require to model the object
- Using a histogram to represent the object of interest.
- Typically an elliptical shape is used.
- There is weighting in the ellipse, weight the values relative to their location. Closer to the centre of the ellipse. Further to the edge of the ellipse might not actually be the object of interest so they are less important in the weighting.
- Modelling the candidate regions that the object could move to.
- Go backwards to validate the tracking, make sure it makes sense forwards as well as backwards.
- Using an iterative gradient ascent to locate the best match. Looking around current location to find out what direction to move in to get the best match. This is NOT exhaustive
- To find the best location, compare the distribution directly (not exhaustive, use the gradient ascent). Look for a shift in the direction.
- Normalised cross correlation gives a large peak which is not ideal to slowly achieve.
- Consider the gradient of the similarity function

Can be histogram of:
Colours,
Oriented gradients,
Texture, etc.

Mean Shift Analysis

Wednesday, November 05, 2025 2:55 PM

Mean shift tracking under a range of conditions and compare with normalised cross correlation
Number of bins per channel
Convergence parameter
Kernel type

Optical Flow

Wednesday, November 05, 2025 2:55 PM

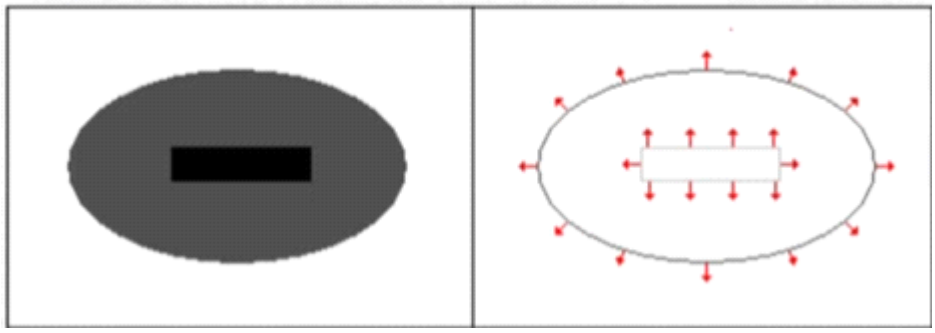
Dense optical flow
Computing a motion field for every point in the image. Where every pixel is going to from frame to frame.
Direction and magnitude points for every pixel in the image
Based on the brightness constancy constraint – object points will have the same brightness over a very short period of time.
Need to find displacement which will minimise residual error
To compute the optical flow assuming that the displacement is small

First Derivative Edge Detection

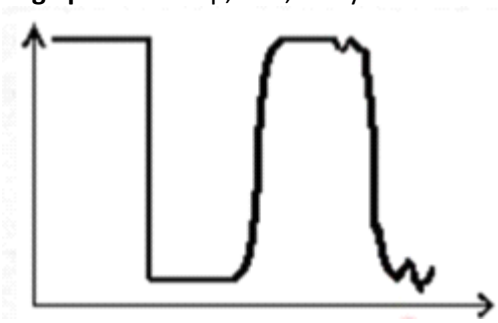
Thursday, October 09, 2025 9:26 AM

Edges are an approach to segmentation – discontinuities in the image
An alternative to region based processing

What is an edge:
Finding the place in an image where brightness changes abruptly
Edges have magnitude (gradient) direction (orientation)



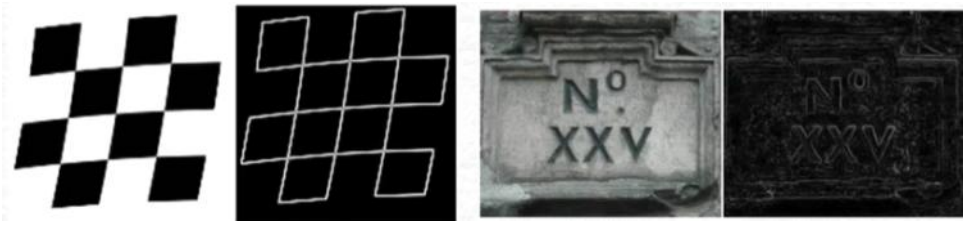
Edge profiles: step, real, noisy



1st derivative edge detection
Rate of change in two directions – in an edge there is an increase in the rate of change then a drop in the rate of change across the pixels that make up the edge
Two direction – two partial derivatives

Digital images
Derivatives work on continuous functions – we have digital/discrete images
Discrete domain: differences, orthogonality

Roberts edge detector
Looks at 2 partial derivatives – consider any image points (i,j). Looks at the difference between that point and the point in the next row and next column. Looks at the orthogonal rate of change.

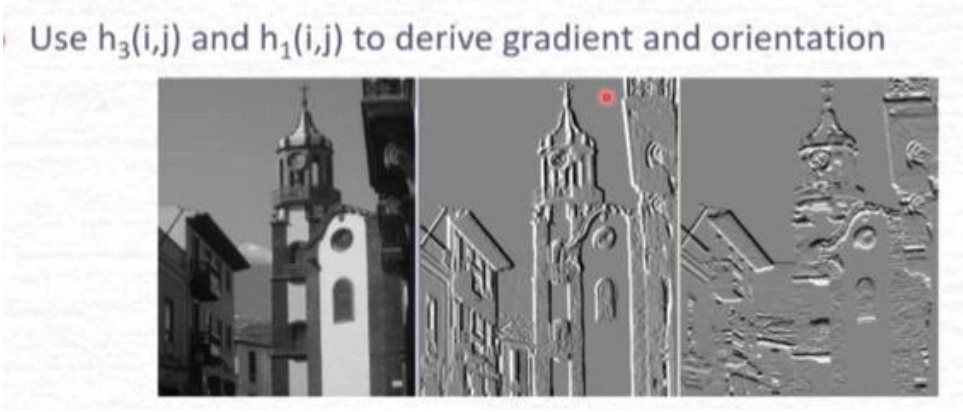


Incredibly sensitive to noise – any noise with any pixel that noise will be retained in the computation
In binary images we get a really strong response – not so much in other images – only one appropriate for binary images – other produce too wide edges

Need to define partial derivatives such that they cross at a single middle point
Cross at the centre of a pixel – Roberts is crossing between pixels – a shift of half a pixel – like to stay in the same spatial domain

Evaluate points that aren't too close together
Deal with some degree of image noise

Prewitt - Compass edge detector
Partial derivatives defined for a number of directions – mask centred to every possible image
Difference of the point above it and point below it – local average of 3 points above and subtracting 3 points below – deals with separation and noise – noise attenuated by averaging
Same mask is rotated by 45 degrees – 8 versions of same operator

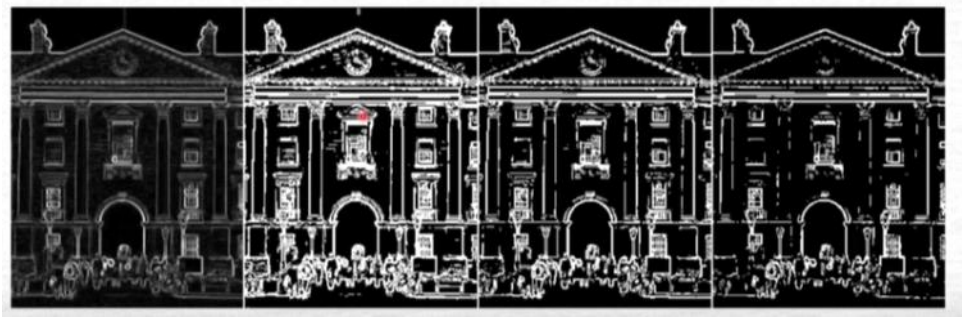


Vertical edges and horizontal edges can be combined – white value very positive change

Sobel – compass operator
More gaussian shape
Orientation can be computed at every single point – not just significant edges – any rate of change gives an orientation value

Sum of the absolute values of the partial derivitaves can be used – not as accurate as root mean square

We have gradient image and orientation image
Apply threshold to gradient image - get too many or too few edge points



Non maxima-suppression
Quantize edge orientations
For all points look at the two points orthogonal to the edge
Thresholding can then be used after non maxima suppression

Canny & Colour Edges

Thursday, October 09, 2025 9:29 AM

Canny edge detection
Combination of first and second derivitave.
2nd deriv was to improve localisation
2nd deriv is the rate of change of the rate of change.
Not clear that there is any improvement in edge localisation
Thresholding incorporated so the result of canny is a binary image

Optional analysis at multiple scales because opbjects appear at multiple scales – may give an idea of significant boundaries in a scene

1. smooth image with Gaussian
2. Compute first derivative gradients and orientations – for location of edge points
3. Use 2nd derivative zero-crossing to suppress non-maxima
4. Threshold edges using hysteresis – double threshold
   Below high threshold & above low threshold = edge point

Smooth image using different Gaussians
Look for correspondence across multiple scales to identify significant discontinuities

Colour edge detection
Grey levels are ordered – colours are not
Multispectral edge detection is addressed with: output fusion (combine edges from different RGB channels),
multidemensional gradient methods (combine gradients) very difficult to develop and no agreed solution,
colours are 3 dimensions so consider them as a vector – distance between colours in 3d

Contour Segmentation

Thursday, October 09, 2025 9:29 AM

Still have pixel-based result i.e. a pixel is an edge or not

Contour segmentation = join the dots

Represent contours as a series of line segments instead of points

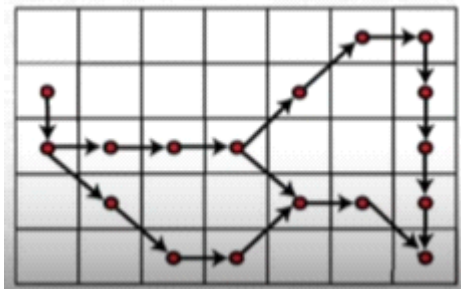
Boundary Chain Codes
Each chain contains the start point and a list of orientations

Number of edge points dictates the edge contour – splits in edge contour can happen

Orientation and scale dependent – slightly position dependent
Can use smoothing with this representation
Hard to get a consistent representation

Lots of noise outside of the sign in the image. Contours don't correspond to individual objects in the scene

Represent an edge image as a graph
For each edge point we can create a node in the graph corresponding to some edge point – some cost and some orientation
The only waypoints can be connected if they are 8-connected neighbours



Consider the strength of the edges (cost) and the difference in curvature

Segment all edge chains in a scene

Start with strongest node in the graph and expand all forwards and all backwards of the specified edge.
Repeat
Modify edge chains to fill small breaks and remove duplicate contours – repeating until stable

Extract straight line segments
Take contours and convert to segments
Straight lines require a tolerance t
Decide where endpoints of line are

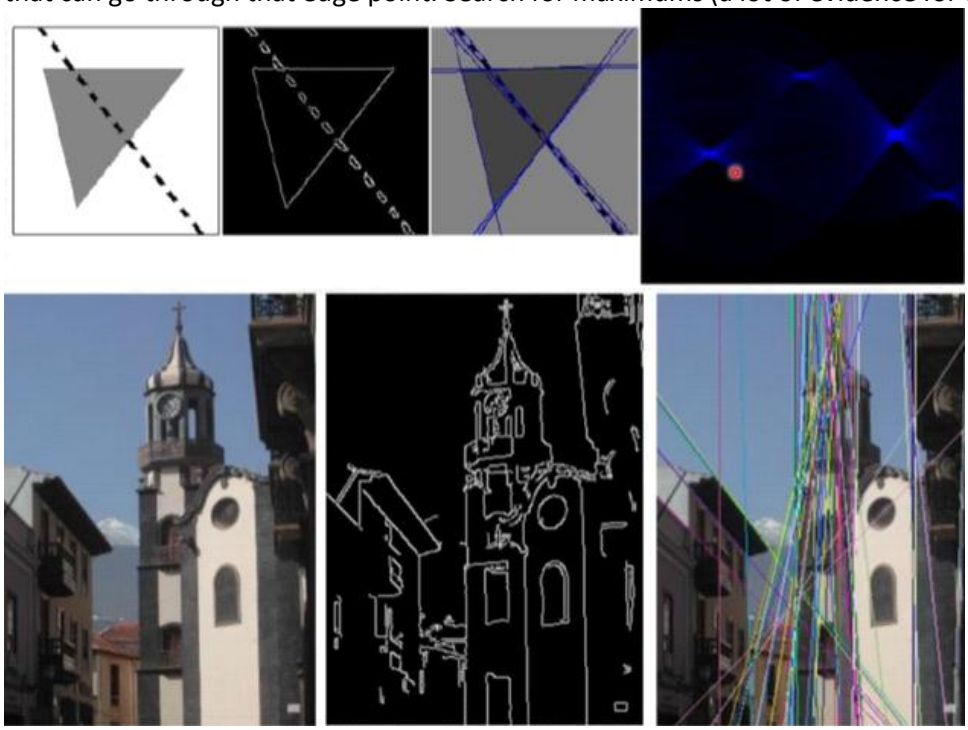
Recursive boundary splitting
Split at furthest point and keep going until the distance is less than the specified tolerance

Hough Transform

Thursday, October 09, 2025 9:29 AM

Hough transform

Transforming from image space to the likelihood of the existence of some shape in the scene
Lines, circles, generalised shapes
Can take image space and project into a Hough space – one unit of evidence increased – accumulates evidence for a particular line
Initialise Hough space accumulator to 0. for every edge point increase accumulator for every line that can go through that edge point. Search for maximums (a lot of evidence for there being a line)



Often get multiple responses
Takes no account on if edge points are connected or not

Hough circle detection

Assume constant radius, transform from image space to hough space again

Least Squared & RANSAC

Thursday, October 09, 2025 9:29 AM

Common to have a number of data points that may form some feature OR something else
These techniques can be used for more than just straight lines and simple shapes
Determine best possible position or pose of this feature/object

Least Squared Error
Linear fit best matching data
Minimise residual error associated with points
Assumes line is not vertical and assumes error distribution is normal – points that should be included in the regression are known – dropping points that are far from estimates

RANSAC
INVOLVED RANDOMNESS- NON DETERMINISTIC
Uses minimum number of data points to determine the model
Take n data points and pick 2 randomly to define line equation
Look at line equation and see what is in tolerance of this
If the consensus set is not big enough – adjust it
If consensus set is big enough – recompute the model using ALL points in the consensus set

TECHNIQUES - EDGES

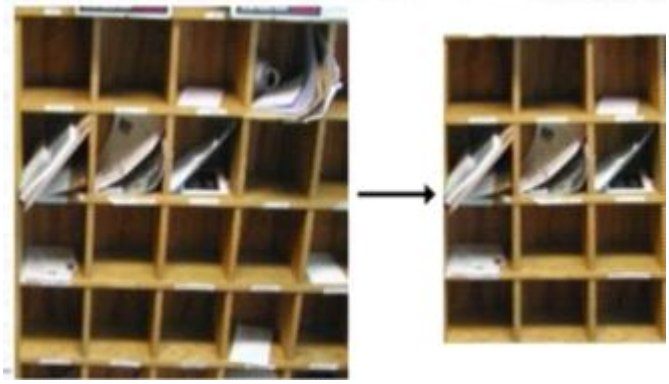
Déardaoin 4 Nollaig 2025 18:50

Technique	Input	Output	Method	Purpose	Other
Roberts Edge Detection					Not very robust to noise
Prewitt Edge Detection					
Sobel Edge Detection					
Canny Edge Detection					
Contour Following					
Recursive Boundary Splitting					
Hough Transform					
Hough Circle Detection					
Least Squared Error					
RANSAC					

Geometric Operations

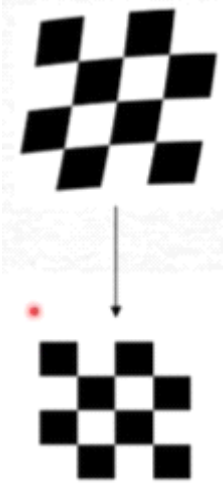
Wednesday, October 15, 2025 2:02 PM

To remove distortion from an image/make easier to process/account for difficult condition



brought bookshelf exactly into frame

Problem: distorted image
Mapping: to a corrected image



Define a transformation such that every pixel in the corrected image can come from somewhere in the distorted image

- Define the relationship/transformation - possible to know in advance
- More common to determine through correspondences – find a well-distributed number of points and their matches
- Apply the transformation – look at the corrected image and work out where that came from in the distorted image and interpolate a value
- Each input points go to a real value

Affine transformation
Taking corrected point and working out where it came from in terms of i,j
Unknown transforms required at least 3 observations between corrected space and distorted space

TECHNIQUES - GEOMETRIC

Déardaoin 4 Nollaig 2025 18:52

Technique	Input	Output	Method	Purpose	Other
Affine Transformation					
Perspective Transformation					

Moravec Corner Detection

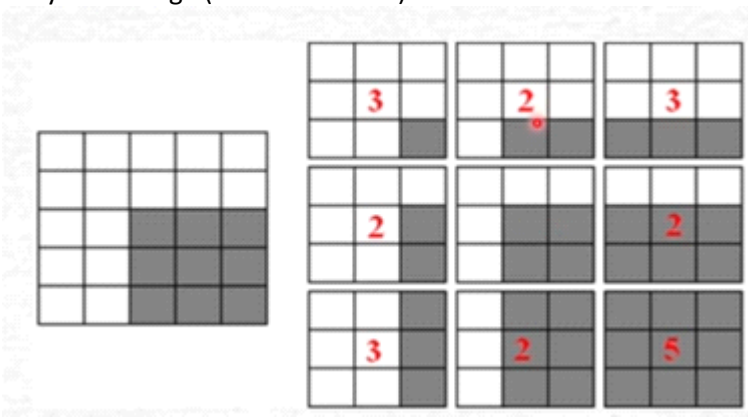
Saturday, October 25, 2025 9:37 PM

Steps for corner detection:

1. Determine cornerness value for each pixel, main difference, produces a cornerness map
2. Non-maxima suppression, multiple responses, compare to local neighbours
3. Threshold the cornerness map, significant corners

Moravec

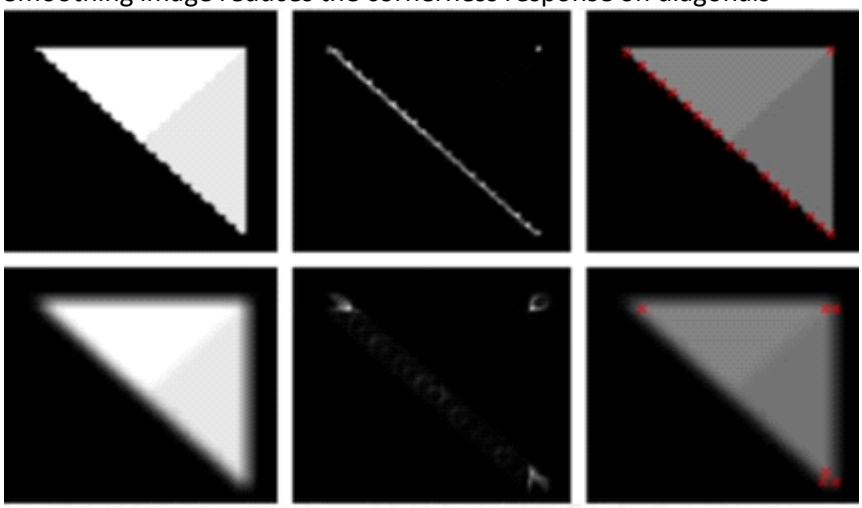
Compares the local variation around a point. Select minimum value
Greyscale image (could be colour)



Minimum difference: two pixels are different
The lower the minimum difference = the lower the cornerness value

Anisotropic response (different depending on the orientation of the feature) – diagonal lines

Smoothing image reduces the cornerness response on diagonals



Possibly a noisy response – use a significantly large area or smooth
This does quite well in low resolution images
Harris does better in high resolution images

Harris-Plessey Corner Detection

Saturday, October 25, 2025 9:37 PM

Main difference is the determining of the cornerness map
Harris uses a lot more complexity than Moravec

Uses: partial derivatives, gaussian weighting, matrix eigeanvalues

Native in opencv
Greyscale image ONLY

Sum of squared difference between two regions (similar to moravec)
We end up with a square matrix to compute eigeanvalues
The matrix is the rate of change in the i&j directions
If both eigeanvalues is high = corner
One eigeanvalue is high = edge
No eigeanvalue is high = constant region

Cornerness measure = $C(i,j) = \text{determinant}(M) - k*(\text{trace}(M)^2)$

More computationally expensive, sensitive to noise, somewhat anisotropic
Very repeatable response (features moving through space), better detection rate (particularly on high resolution images)

SIFT – Scale Invariant Feature Transform

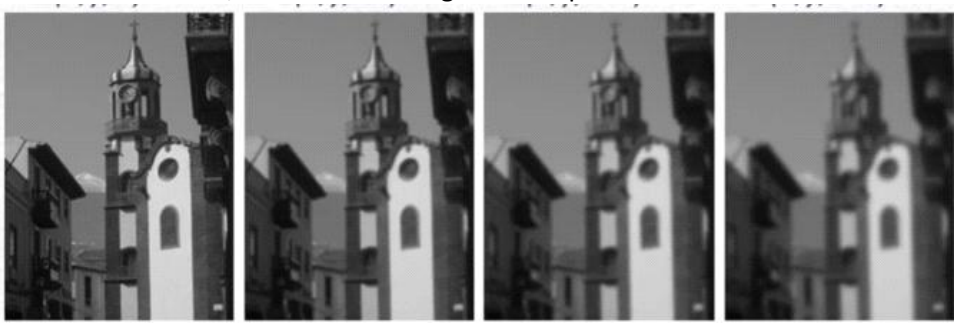
Saturday, October 25, 2025 9:37 PM

Repeatable, robust features for tracking, recognition, panorama stitching

Invariant to scaling and rotation, partially invariant to illumination and viewpoint changes

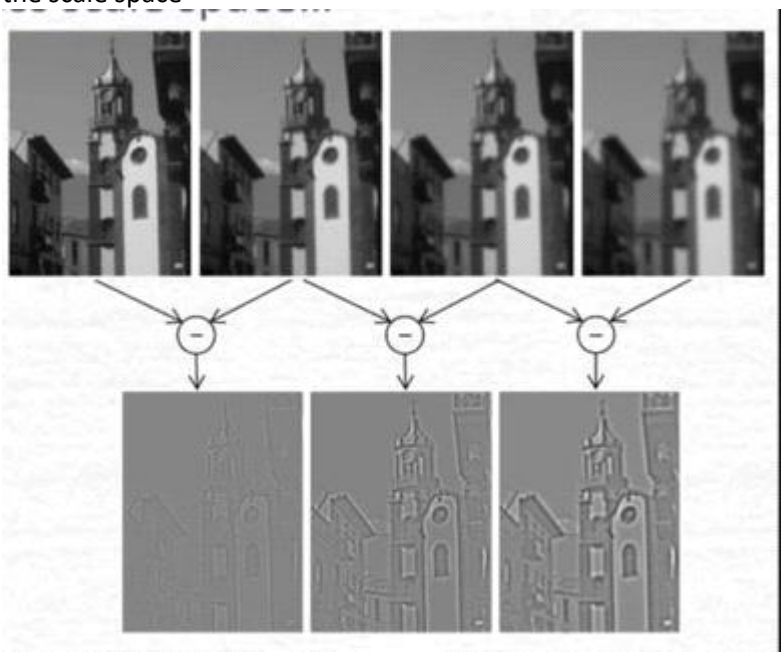
1. Scale space extrema detection
Scale space, consider different resolution
Difference of gaussian, smoothed by gaussian smoothing
Locate extrema.

For scale invariance, consider the image at multiple scales



Consider at each octave of scale space, a doubling of sigma

Stable keypoint locations are defined as the extrema of the difference of gaussian functions across the scale space



Extrema: a point that is bigger/smaller than all its neighbours
Must be bigger than the neighbours in the other spaces/adjacent scales in any octave

2. Accurate keypoint location
Sub pixel locate
Filter response – remove low contrast and feature along an edge

Originally, location and scale taken from central point
Locate points more precisely
Model data as a 3d quadratic

Locate the interpolated maxima/minima
Discard low contrast keypoints – evaluated from the curvature of the 3d quadratic

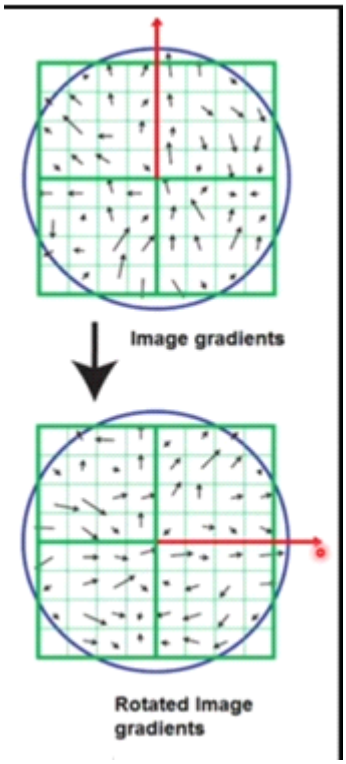
Discard poorly localised keypoints - e.g. along an edge, similar eigeanvalue computation as harris corner detection

3.Keypoint orientation assignment
For the scale invariance, the keypoint scale is used to select the smoothed image with the closest scale
For orientation invariance we describe the keypoint – the principal orientation
Create an orientation histogram for the keypoint
Weight by gradient magnitude
Sample points around the keypoint
Highest peak = the orientation, but there are multiple possible orientations
Stable results under noisy conditions
Multiple orientations = multiple key points in the same place

4.Keypoint descriptors
Could sample image intensity at relevant scale
Match using normalised cross correlation
But this is sensitive to: affine transformations, 3d viewpoint changes, non-rigid deformations

Use blurred image at closest scale
Sample points around the keypoint

Compute gradients and orientations around that point
Rorate by keypoint orientation
Divide region into subregions



For each subregion, create histogram weighted by gradient, weighted by location (Gaussian), distribute into bins

32 values that describe a keypoint/feature
An orientation associated with that feature
32 values are based on the orientation being normalised – facilitates matching between keypoints

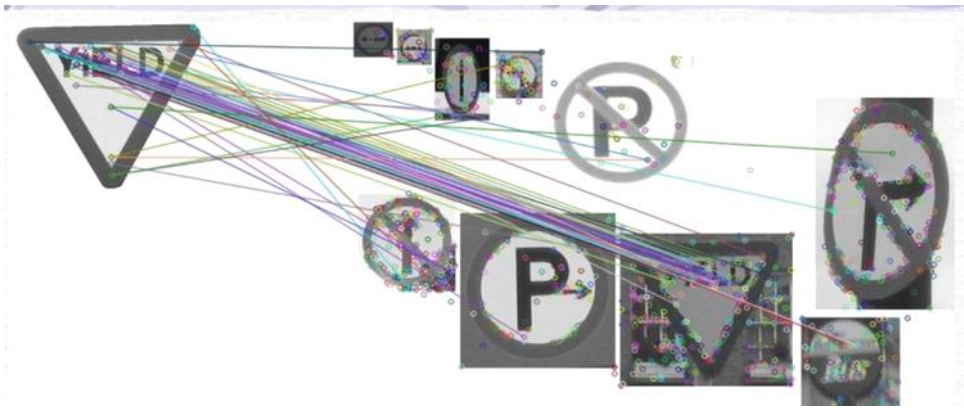
5. matching descriptors – dropping poor ones

6.Recognition of an object – matched a number of features to locations that make sense

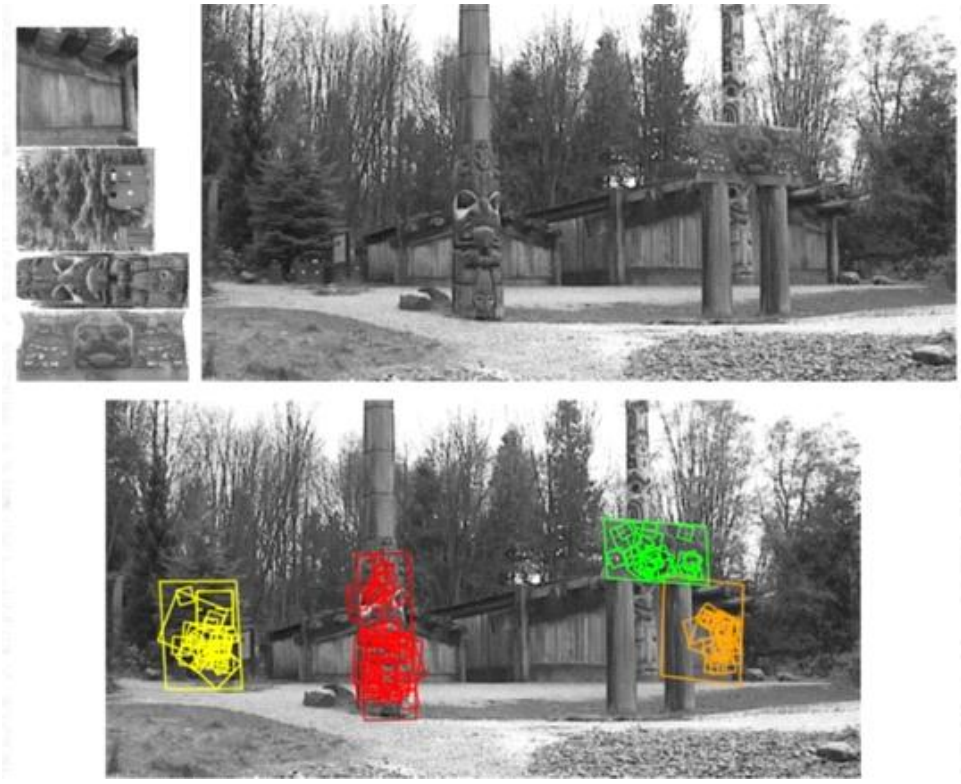
SIFT Recognition

Saturday, October 25, 2025 9:37 PM

5. matching descriptors – dropping poor ones
Nearest neighbour matching – euclidean distance between keypoints
Compare the distance of the closest neighbour to the 2nd closest (from a different object) and make sure they're not too similar – gets rid of a load of inappropriate matches and a little number of real matches



6.Recognition of an object – matched a number of features to locations that make sense
Match at least 3 features for recognition because this can define an affine transformation
Cluster the matches
Use 4d hough transform – two for location, one for scale, one for orientation
Use really broad bins
Consider all bins with at least 3 entries



SIFT (ChatGPT)

Tuesday 18 November 2025 17:07

Where are important points in this image?
&
How can we describe each point in a way that lets us find it again in another image?

- SIFT detects corners, blobs, textured areas - NOT flat sky, plain walls
- Even if some image is zoomed, rotated, darker, etc. SIFT will still be able to do recognition on the keypoints it already knows
- SIFT ignores scale changes (scale invariant), ignores rotation, ignore lighting changes
- SIFT takes in a greyscale or colour image but usually greyscale

Find special spots --> Describe them in a way that is robust --> match them between images.

1. **Finding Keypoints At Different Scales**

- SIFT creates a stack of blurred images at different stages, ranging from not that blurred to really blurred. Each of these is called a **different scale**.
- SIFT compares these **scales** to each other using a Difference of Gaussians. This is because features which are more important in the image will continue to prevail even as the blur worsens.
- The **candidate keypoints** which are identified in this step are identified based on how they stand out from their neighbours in space (x,y) and scale (blur)

2. **Remove Unstable Keypoints**

- There are currently many **candidate keypoints**, some are noisy or not distinct enough
- Some are along edges --> not good because they aren't stable to small changes
- Some are too weak --> don't really stand out from the background
- SIFT throws away keypoints that are too faint (low contrast) or lie on edges
- We only want keypoints that are likely to be found again in other images, low contrast and edges are not likely

3. **Assigning an Orientation to Each Keypoint**

- Around each keypoint, SIFT looks at the brightness changes in different directions around the keypoint (gradient)
- SIFT uses the dominant direction as the orientation for that keypoint
- Now each keypoint has a **position** (x,y), **scale** (how big it is), **orientation** (which way it's pointing)

4. **Creating Keypoint Descriptors**

- The keypoints are turned into a **feature vector** a list of numbers that describe the local pattern.
- Summarise how the image gradients for sub-images are distributed by direction.
- All of the summarised vectors into one long list is the **keypoint descriptor**.
- Fairly robust to small changes in lighting
- Robust to small shifts and distortions
- Rotation invariant and scale invariant.

5. **Matching Descriptors and Dropping Poor Ones**

- For each keypoint in image A, compare its descriptor to all descriptors in image B
- Find the nearest neighbour, the descriptor in image B which is closest to the descriptor in image A we are querying.
- Drop any matches that are poor, the good matches tell you which points in image A map over to image B.

6. **SIFT Recognition**
   - Match at least 3 features for valid recognition because this can define an affine transformation
   - Use a **4D Hough Transform** - Two for location (x,y), One for scale, One for orientation
   - Use really broad bins and consider all bins with at least 3 entries

TECHNIQUES - FEATURES

Déardaoin 4 Nollaig 2025 18:53

Technique	Input	Output	Method	Purpose	Other
Moravec Corner Detection					
Harris-Plessey Corner Detection					
SIFT (features)					
SIFT (recognition)					

Otsu Thresholding

Adaptive Thresholding

Wednesday 12 November 2025 17:05

Average Filter Smoothing

Wednesday 12 November 2025 17:05

Median Filter Smoothing

Wednesday 12 November 2025 17:05

Erosion

Wednesday 12 November 2025 17:06

Dilation

Wednesday 12 November 2025 17:06

Opening

Wednesday 12 November 2025 17:06

Closing

Wednesday 12 November 2025 17:06

Histogram Back Projection

Wednesday 12 November 2025 17:06

Connected Components Analysis

Wednesday 12 November 2025 17:06

K Means Clustering

Wednesday 12 November 2025 17:06

Watershed Segmentation

Wednesday 12 November 2025 17:06

Mean Shift Segmentation

Wednesday 12 November 2025 17:07

Non-Maxima Suppression

Wednesday 12 November 2025 17:07

Roberts First Derivative Edge Detection

Wednesday 12 November 2025 17:07

Sobel Edge Detection

Wednesday 12 November 2025 17:07

Canny Edge Detection

Wednesday 12 November 2025 17:07

Contour Following

Wednesday 12 November 2025 17:11

Recursive Boundary Splitting

Wednesday 12 November 2025 17:09

Hough Transform

Wednesday 12 November 2025 17:08

RANSAC

Wednesday 12 November 2025 17:08

Purpose:

Input - NOT AN IMAGE:

- a set of data points/vectors from the source image

Output - NOT AN IMAGE:

Method:

Affine Geometric Transformation

Wednesday 12 November 2025 17:08

Perspective Geometric Transformation

Wednesday 12 November 2025 17:08

Moravec Corner Detection

Wednesday 12 November 2025 17:08

Harris-Plessey Corner Detction

Wednesday 12 November 2025 17:08

SIFT

Wednesday 12 November 2025 17:09

Mean Shift Tracking

Wednesday 12 November 2025 17:09

Optical Flow Tracking

Wednesday 12 November 2025 17:09

Static Background Model

Wednesday 12 November 2025 17:10

Median Background Model

Wednesday 12 November 2025 17:10

Gaussian Mixture Model

Wednesday 12 November 2025 17:10

Template Matching

Wednesday 12 November 2025 17:10

Chamfer Matching

Wednesday 12 November 2025 17:10

Robust Object Detection / Cascade of Haar Classifiers

Wednesday 12 November 2025 17:10

Principal Components Analysis

Wednesday 12 November 2025 17:11

Template Matching

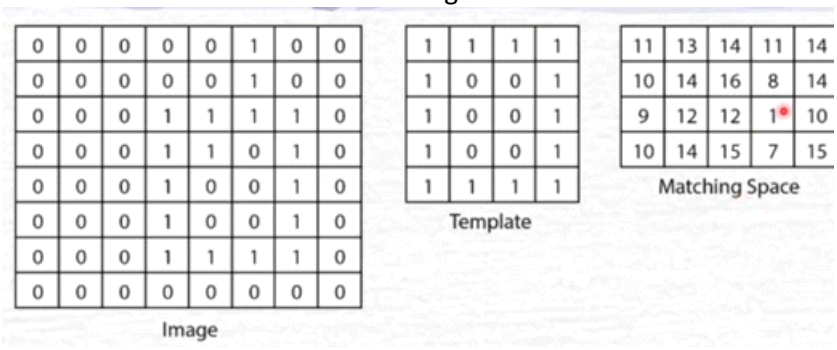
Wednesday 12 November 2025 15:07

Comparing a picture with a picture
Template matching is very expensive - can take a long time if not optimised correctly

Can be used to find things
Purpose: Visual inspection / golden template matching. Making other products like the gold standard

Template matching algorithm
Inputs: image and object (target)
Method: For every possible position in the image pixel by pixel, evaluate the match criteria
Search for local maxima (or minima) of the match criteria above some threshold
Output: maxima of 2d array of matching values

Every possible position is expensive
Local maxima above a threshold is vague



Insists that the entire target is contained within the image space

Finding local maxima/minima take a binary image and dilate and look for unchanged values. AND together with a thresholded version.
Same for local minima, just erode instead

Goal: localise close copies

- Assumes that the size and orientation of the object remains the same.
- If this is not true then a template is needed for *every* size and orientation i.e. many different targets.
- Constrain the geometric distortion by using a low resolution image first then starting to use higher and higher resolution images. Searching in low resolution images first prevents things like glare on windows from causing false negatives. Once the target is found in the low resolution version, you can take those low resolution wins and process them in the higher resolution version.
- Searching higher probability areas first. Can use learning, can learn where particular objects are appearing in an image e.g. middle. Only good for searching for one instance of an object, not for finding all instances because you're limiting yourself to a space

Chamfer Matching

Wednesday 12 November 2025 15:07

Template matching requires very close matches to be considered as a valid match.
Difference between target object and image has to be very small
Chamfer matches relaxes this by considering the boundaries of the target object and found object

Objects often appear very slightly different due to noise, orientation, sampling

Input: A target object and an input image/video as an edge map
Output: A measure of similarity between the template image and the query image based on the boundary information
Purpose: an extension of template matching which is able to deal with increases in size, changes in orientation
Method: for every point in the image, compute a Chamfer value. If an edge point/object point set the Chamfer value to 0 and if it's not a boundary point, set the value to 1.
Go through this image form **top left** to **bottom right** and consider the preceding pixels above and to the left of the current pixel.

∞	∞	∞	∞	∞	0	∞	∞
∞	∞	∞	∞	∞	0	∞	∞
∞	∞	∞	0	0	0	0	∞
∞	∞	∞	0	0	∞	0	∞
∞	∞	∞	0	∞	∞	0	∞
∞	∞	∞	0	∞	∞	0	∞
∞	∞	∞	0	0	0	0	∞
∞	∞	∞	∞	∞	∞	∞	∞

Object pixels (zeros)

∞	∞	∞	∞	∞	0	1	2
∞	∞	∞	∞	1.4	0	1	1.4
∞	∞	∞	0	0	0	0	1
∞	∞	∞	0	0	1	0	1
∞	∞	∞	0	1	1.4	0	1
∞	∞	∞	0	1	1.4	0	1
∞	∞	∞	0	0	0	0	1
∞	∞	∞	1	1	1	1	1.4

After the first stage of processing

This gives you the distance of how far away from an object pixel you are

A second pass starts from the **bottom right** to the **top left**, using points below and to the right, to propagate in the opposite direction - this is the final Chamfer image:

∞	∞	∞	∞	∞	0	∞	∞
∞	∞	∞	∞	∞	0	∞	∞
∞	∞	∞	0	0	0	0	∞
∞	∞	∞	0	0	∞	0	∞
∞	∞	∞	0	∞	∞	0	∞
∞	∞	∞	0	∞	∞	0	∞
∞	∞	∞	0	0	0	0	∞
∞	∞	∞	∞	∞	∞	∞	∞

Object pixels (zeros)

∞	∞	∞	∞	∞	0	1	2
∞	∞	∞	∞	1.4	0	1	1.4
∞	∞	∞	0	0	0	0	1
∞	∞	∞	0	0	1	0	1
∞	∞	∞	0	1	1.4	0	1
∞	∞	∞	0	1	1.4	0	1

∞	∞	∞	0	1	1.4	0	1
∞	∞	∞	0	1	1.4	0	1
∞	∞	∞	0	0	0	0	1
∞	∞	∞	1	1	1	1	1.4

After the first stage of processing

3.8	2.8	2.4	2	1	0	1	2
3.4	2.4	1.4	1	1	0	1	1.4
3	2	1	0	0	0	0	1
3	2	1	0	0	1	0	1
3	2	1	0	1	1	0	1
3	2	1	0	1	1	0	1
3	2	1	0	0	0	0	1
3.4	2.4	1.4	1	1	1	1	1.4

Chamfer Image

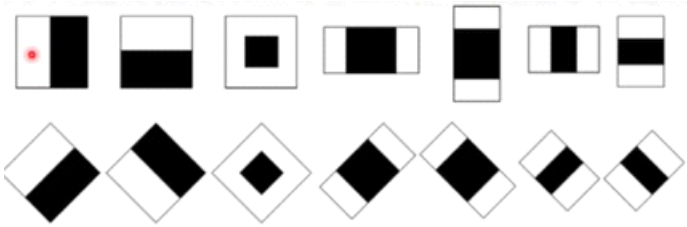
Robust Object Detection Using Cascade of Haar Classifiers

Wednesday 12 November 2025 15:07

Input: Greyscale images
Output: A detected face i.e. bounding box around a face
Method:
Purpose: Find full-frontal faces at different scales

- Train the recognition model using a number of positive and negative samples.
- Use very simple Haar-like features - efficient calculation
- Selects a large number of these features during training to create strong classifiers
- Each feature when combined with a threshold to create strong/weak classifiers
- Pass the image along the strong classifiers (cascading)
- Can work at different scales by looking at different size objects to run the cascading classifiers on. Each window will be scaled to a fixed size for the classifiers

Features:



Place the masks in a specific location in the window of interest
Subtract the sum of white pixels from the black pixels and compare with a threshold

Training:
Variations in size and position is huge, lots of possible features.
Identify what is appropriate to use.
Use negative and positive samples to train the sample.
Take each individual feature and tune it, tune the threshold to maximise correct classification
Combine weak classifiers into strong classifiers

Boosting Algorithm:
ADABoost
With n example images with associated classifications 0 or 1.
A weight is associated with each image.

PCA Principal Components Analysis

Wednesday 12 November 2025 15:08

Input: treat each face image as a vector
Output:
Purpose: data analysis, data compression, recognition. Images that have a significant number of dimensions. Can be used for facial recognition
Method:

1. Create a data matrix where each face image is a row
2. Create mean centred data. Subtract the average face from all the rows.
3. Compute the covariance matrix
4. Solve for eigenvalues and eigenvectors
5. Normalise the eigenvectors
6. Select some number of eigenvectors and their eigenvalues to describe the faces, usually the ones with the largest eigenvalues
7. To find the identity of an unknown image in the PCA space, you can find the closest match/ nearest neighbour

- start off with a number of samples
- Create a samples matrix
- Call PCA to compute eigenvalues, eigenvectors and mean
- Transform the data

Satellites can have 100> dimensions, much of the data doesn't tell us a lot. Need a technique to identify the significant parts.

Analyses data covariance, how it changes, finds principal directions/dimensions in which it changes.

TECHNIQUES - RECOGNITION

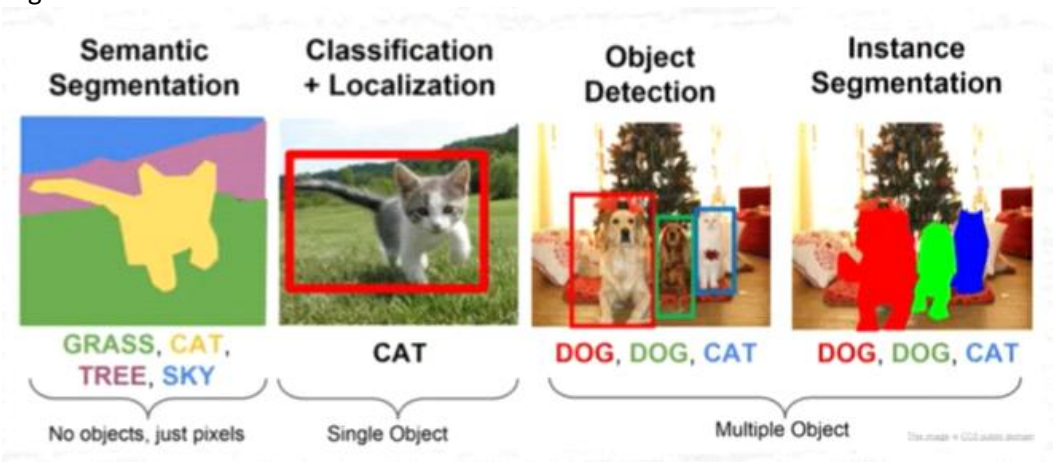
Déardaoin 4 Nollaig 2025 18:54

Technique	Input	Output	Method	Purpose	Other
Template Matching					
Chamfer Matching					
Robust Object Detection Using a Cascade of (Haar) Classifiers					
Principal Components Analysis					

Video 1

Wednesday 19 November 2025 17:05

Recognition Tasks
Take a single image and classify what is in it simply, the main object in the scene
Also: semantic segmentation, localisation classification, (multiple) object detection, instance segmentation



L1 distance between two images to do classification
Nearest neighbour, select the closest - not great

But we can extract some features from images

Linear Classifiers (Nearest Neighbours)
Segment what we're interest in from the rest of the image, compute features for the region and compare to what we know already as a weighted combination and select the closest

Compute: minimum bounding rectangle, convex hull.
Features we can compute from this: area, rectangularity, elongatedness, height to width ratio

Create a feature vector and sum all features together as a weighted sum and get a score.

<u>Problems with linear classifiers</u>:

- How to select the best features? Needs an expert
- How to set the best weights?
- How to deal with unknown objects?

Loss function gives a class that is compared with the correct class. 0 means no loss, exact match

The boundaries may not be linear - warp feature space and turn boundaries into linear boundaries

Likely that classes will overlap - mistakes or overfitting - add a regularisation to the loss function

To find the best weight, compute loss for initial weight and move the weights around to see how the loss function reduces the most.

Video 2

Wednesday 19 November 2025 17:26

Neural Networks
Allows classification where separation is not linear
Solves the problem of selecting appropriate features

- Hidden layers
- Fully connected layers
- Activation functions (ReLu, sigmoid)

Input nodes (features) passed to hidden layer. All input nodes are feeding this.
Activation function within hidden layer allows to deal with nonlinearities
Outputs is a weighted sum of its layer/ the likelihood of a class

What inputs to use? Pixel values / features
How to determine weights?

Activation functions are required to deal with nonlinear spaces

ReLu rectified linear unit
Computes maximum of zero or the value, negatives become zero, positives stay the same
Allows system to train knowing this is the outcome

Size of hidden layers impact capacity

To cope with overfitting in the neural network, change the way which we evaluate regularisation

Back Propagation
Setting weights within a network, uses computational graphs

Back propagate the loss back to the weights and see how the weights change

Video 3

Wednesday 19 November 2025 17:26

Datasets
MNIST dataset, handwritten digits. 60k training images, 10k test images

CIFAR 100 - classes of objects, sample superclasses and classes

ImageNet - 20k classes, 14M images, hand annotated

COCO - 91 common classes recognised by a 4 year old, 5 captions provided per image

CNN - Convolutional Networks (Image Classification)
Fully connected layers

Video 4

Wednesday 19 November 2025 17:26

ResNet - Residual networks

Transfer Learning

RNN - Recurrent Neural Networks (Image Captioning)

R-CNN - Region Based CNN (Multiple Object Detection)

Determine if there is a spoon in the baby food can

Dé hAoine 5 Nollaig 2025 14:21

1. Convert the image to **greyscale**

2. Convert the image to binary using **Adaptive Thresholding** as there appears to be some glare/shadows on the can of baby food

3. **Count foreground pixels**, such that the whiter area of the can is the background and the darker area of the spoon is the foreground. If the foreground pixels are above a certain ratio, there is a spoon in the can. Another threshold can be defined to work out if there are two spoons in the can.

Determine if a label is on a bottle of adhesive.

Dé hAoine 5 Nollaig 2025 14:22

1. Convert the image to greyscale
2. **Otsu Threshold** the image to binary as there is no obvious glare on the bottle.
3. Count the ratio of foreground pixels to background pixels. If there are little to no foreground pixels, then there is no label on the bottle of glue. If there are a lot of foreground pixels, there is a label

Identify the red, white & black pixels of these road signs.

Dé hAoine 5 Nollaig 2025 14:29

1. To find the **black** region, convert the image to greyscale and set a very high value for Otsu thresholding. With a very high value set, the black pixels will be the only pixels included in the foreground, not the red pixels.

2. To find the **red** and **white** regions, convert the image to greyscale and threshold on the red channel. White is also present in the red channel, so the white pixels will be picked up by this. Do another threshold on the blue channel because red and blue do not overlap. Compare the results of the red and blue thresholds to separate the red from the white

Determine the percentage of fat present.

Dé hAoine 5 Nollaig 2025 14:34

Avoiding the marbling of the meat, just focusing on the edge pieces of fat

Convert the image to greyscale

Convert the image to binary with Otsu thresholding

Perform an opening ooperation on the binary image. This keeps the shape structure of the fat but will break apart close together regions in the marbling.

Count the foreground pixels after the opening operation and determine a % against the rest of the greyscale

Locate any shipping containers on the quayside

Dé hAoine 5 Nollaig 2025 14:36

Perform a **Mean Shift Segmentation** on the raw RGB image.

Mean shift segmentation gives a colour and region separation, therefore the ships and water will be affected.

After this, iterate around the regions identified by the mean shift segmentation and perform a measure of rectangularity operation.

Shipping containers are roughly the same size, therefore have some rectangularity value that the shipping containers should be in and around.

Discard any regions which do not fit this measure of rectangularity

Locate and extract the square information signs.

Dé hAoine 5 Nollaig 2025 14:39

Convert the image to greyscale

Convert the image to binary using Otsu Thresholding.

Perform an opening operation, as there are some white areas we want to get rid of e.g wall in the centre of image 4

Perform a closing operatoin, as there are some small white areas we want to try preserve e.g. The square information sign in image 3

Perform connected components analysis on the binary image to obtain the connected binary regions

Iterate over the connected components and perform a measure of rectangularity and a measure of elongatedness on them. Squares are rectangular but are not elongated. This gives the binary regions where the sqaures are (the white border only, many of the information within will be discarded by the rectangularity/elongatedness measure)/

Taking the original binary image, refine the space to only be within the areas defined by the identified white squares.

This separates out the square information signs from the rest of the image.

By taking the uppermost-leftmost, uppermost-rightmost, bottommost-leftmost and bottommost-rightmost pixels of each of these found regions (no need for an independent corner detector when the extracted shapes are already confirmed as squares and have been extracted from the rest of the image) and perform a perspective geometric transformation using these four feature points to fully extract a straight-on view of the sign.

Locate any visible traffic cones

Dé hAoine 5 Nollaig 2025 14:48

Traffic cones are designed to stand out, convert the image into HLS space and perform a threshold on the luminance channel.
This may produce false positives in the daytime e.g. White cars and clouds have high luminance
This may produce false positives in the nighttime e.g. Headlights have high luminance

Road markings also have high luminance in day or night time conditions.

Perform connected components analysis on the binary image containing the high luminance spaces.

Perform a measure of elongatedness on the regions identified by CCA. Traffic cones are taller than they are wide, which should rule out white cars, clouds, and road markings.

Determine if the label is straight on a bottle of adhesive

Dé hAoine 5 Nollaig 2025 15:02

Convert the image to greyscale

Perform a Sobel Edge Detection on the greyscale image, giving a gradient of the magnitude of the edges and a gradient of the direction.

We only want to consider vertical lines, threshold the gradient direction and discard any strong suggestions of horizontal.

With the vertical gradient magnitudes, perform non-maxima suppression on the verticals to get a thinned version of the edge map, threshold this with how strong the edges must be to be considered valid.

Consider the first two vertical lines in the image (working from left to right).

Use contour following on Line 1's edge map to get its connected contour points
Use contour following on Line 2's edge map to get its connected contour points.

Fit two line equations for these contour points using RANSAC. And evaluate the slopes of these lines to determine if they are colinear i.e. The label is straight

Find all (text) notices in an image

Dé hAoine 5 Nollaig 2025 15:23

Text notices in these images are all encased by signposts (edges)

There are also many other edges in the background of these images e.g. Car roof, grass, branches

This is an RGB image, convert it to greyscale.
The direction of edges is not important for this application, we can use a Canny Edge detection which natively implements the non-maxima suppression and thresholding steps required after other edge detectors.
Canny Edge detection outputs a robust edge map.

Apply contour following to the outputted edge map to get the boundary of these straight lines, this should result in a rectangular shape i.e. Where the text notices are located.

By reducing the original RGB image down to just these areas of interest, we have extracted the text notices in the image.

This may produce false postives e.g. The window in the background of image 2 contains four straight lines which may be mistaken for a sign as per this application.

With samples of the different letters in different fonts, we can constrain the Canny edge map down to the area identified by the contour following. By this manner, we can apply Chamfer Matching to the edges within the signs to further identify the letters of the signs.

Locate any door in the view of a static video camera

Dé hAoine 5 Nollaig 2025 15:43

Edges move across the scene as the door moves. They change in orientation because the camera is static and length because a person is walking in front of it partially

Use a static background model. The door is not going to move.

Convert this image to greyscale.

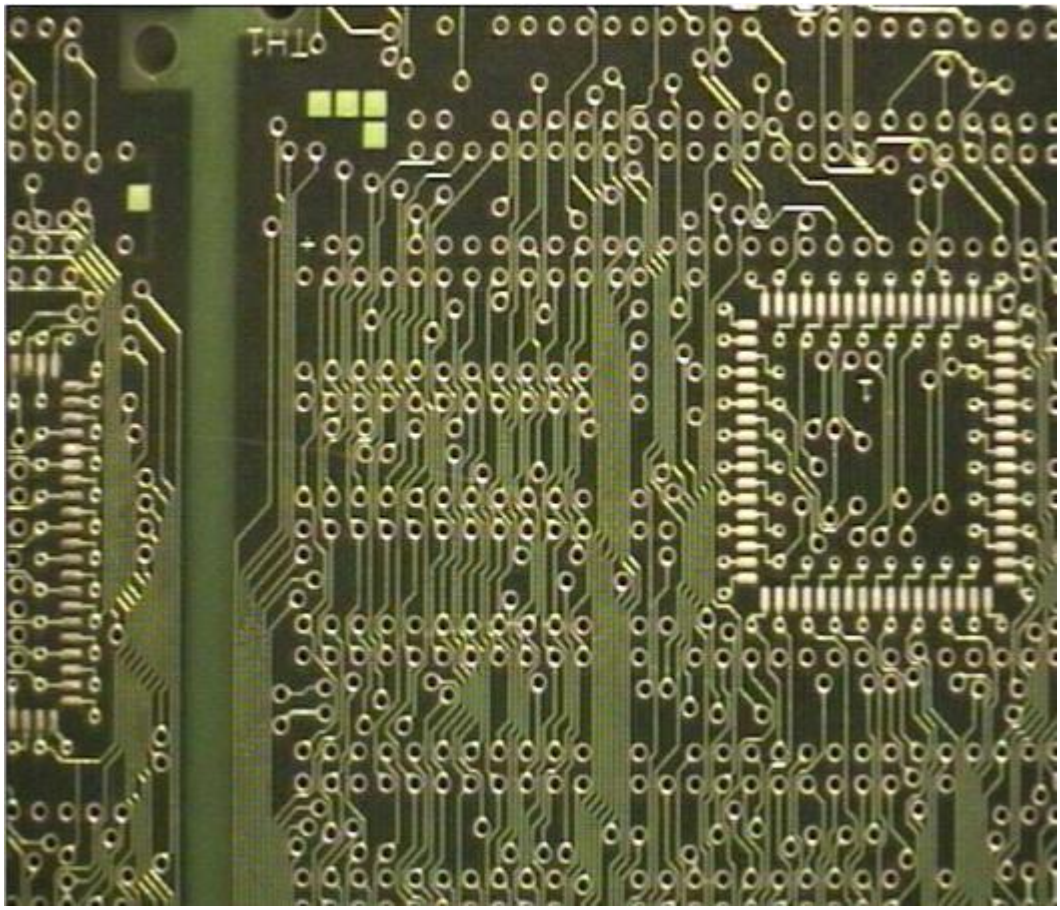
Do a Canny edge detection on this image. Both vertical and horizontal lines are moving in this video when the door opens so direction of the lines is irrelevant.

Perform the Hough transform on the edge map that Canny gives, resulting in the detected srtaight lines. This may also be detected on a persons trousers etc. But ideally a person will not be in the first frame of a security camera video.

After a number of frames, compare the state of the edges against the original image to see if the edges have moved. If the edges have moved, there is a door there

Identify any possible breaks or joins in the PCB tracks.

Dé hAoine 5 Nollaig 2025 16:20



Convert image to greyscale

Convert image to binary using otsu thresholding

Clone this binary image

Perform erosion on one binary image

Perform dilation on the other binary image

Count the pixels and compare to the original binary image

Determine the time from on an analogue clock.

Dé hAoine 5 Nollaig 2025 16:21

Clocks are circles with straight lines in them.

There must also be contrast between the clock numbers/hands and the clock face

Convert the image to greyscale
Perform a canny edge detection
With this edge map apply the Hough transform for circles to detect circles in the image (likely clock)
Reduce the search space to just be this area.
Select the topmost, bottommost, leftmost and rightmost pixel associated with this Hough circle.
Use these four feature points to perform a perspective geometric transformation, such that the clock is now straight-on.
With this edge map that is now straight on and contained within a circle, perform the Hough transform for straight lines.
Use non-maxima suppression on these Hough votes and ensure that the lines located are of a significant size i.e. Don't include the 1 and the 11 on the clock.
There are two angles between clock hands where they meet.
Work out the time by analysing the angles between the clock hands

Determine if people are wearing a backpack in a video stream from a static camera

Dé hAoine 5 Nollaig 2025 19:59

Use a Gaussian Mixture Model for the video stream, as this is likely a crowded scene

Moving pixels are in the foreground

Apply opening on the binary image pixels to break apart people who are standing close together but maintain the overall structure of the people

Perform Connected Components Analysis on the foreground pixels.

For each region identified by the CCA, perform a measure of circularity at the top of the region (a person's head)

Discard any regions which do not have a circle at the top of the region. (not a person)

Discard any circularity matches which are much larger or much smaller than the average (outliers).

Constraint the search to the CCA regions with a circle at the top.

Perform a measure of rectangularity and a measure of elongatedness on the rest of the CCA regions which remain.

Discard any regions which did not have an elongated rectangle underneath the circle (floating circles e.g. Sign posts)

Discard any regions which did not correspond with the average rectangularity and elongatedness of the crowd (not people)

Constraint the search to the CCA areas which are circles on top with an elongated rectangle underneath.

Apply this constraint to the original RGB video.

Convert the video frames to greyscale and perform a Sobel Edge Detection to give a gradient of magnitude and direction.

Discard any horizontal edges (irrelevant to backpack straps).

Perform non-maxima suppression and thresholding on the gradients to get an edge map.

Perform the Hough Transform for straight lines on the edge map within the rectangle.

Backpack straps can be identified when there are four straight lines identified within the rectangle.

This is a forward-facing person wearing a backpack

Locate the traffic lights and report location & colour

Dé hAoine 5 Nollaig 2025 20:12

This is a moving video camera (dashcam).

Convert the RGB video to HLS space

Traffic lights are designed to be bright. Perform a threshold on the luminance channel.

This will pick up false positives e.g. White road markings, brightly coloured cars, bright sky, fluorescent lights

Perform a dilation operation on the foreground, as initially traffic lights will begin far away and become closer as the car approaches them.

Perform CCA on each of the bright/luminant foreground regions

For each CCA region, perform a measure of circularity on the regions to eliminate the sky, cars, road markings.

This may still pick up false positives e.g. Brightly coloured signposts which are circular

Sign posts typically have a white or black element to them, where traffic lights inherently do not.

Perform a threshold on the saturation channel. Road signs will now appear as hollow circles (bright red border in the foreground, text/symbol in the background), whereas traffic lights will appear as fully filled circles (fully saturated colour).

Regions which are not fully saturated with colour should be discarded.

In the moving video camera, traffic lights are located to the left of the vehicle in question. Therefore the circle should be moving more left and down as frames continue.

Perform a mean shift tracking operation on these regions in the RGB frame, this tracks the direction in which the histogram of the object is moving through the scene. The object should be moving left and down.

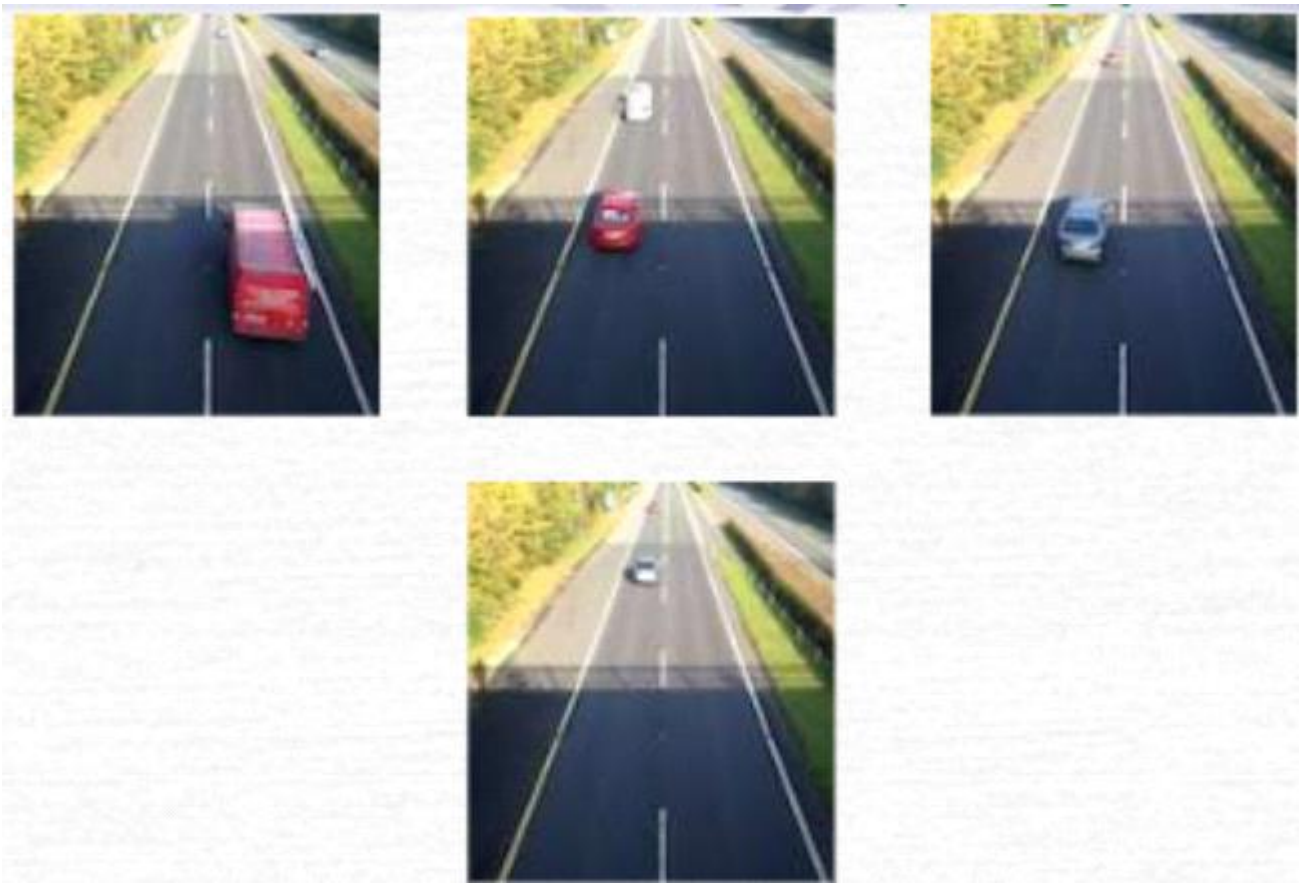
This confirms an oncoming traffic light.

To confirm the colour of the traffic light perform a threshold on the green channel. Red, orange and green all contain elements of the green channel.

Use a low threshold to account for red and orange.

Determine the colour of each vehicle passing by

Dé hAoine 5 Nollaig 2025 20:34



This is a static video camera where a shadow is being cast from the bridge at different times per day.

1. Use a **Gaussian Mixture Model** to account for the changes in lighting and outdoor environment.
   This eliminates the video to just the moving pixels, where the vehicles are located.

2. Perform **Connected Components Analysis** on the regions identified by the binary video.

3. Perform **Contour Following** on the CCA regions.

4. For each region encapsulated by the contour following, create a **Colour Histogram** for each region.

5. Normalise the **luminance** of each colour by taking a weighted average, this is to account for the lighting changes across the scene, such that the same vehicle does not become registered as two different colours.

6. Analyse the peaks of the colour histogram for each vehicle

Track play on the tennis court

Dé Máirt 9 Nollaig 2025 21:35

Find the ball:

1. This is an outdoor scene. Perform a **Gaussian Mixture Model** on the video to account for subtle changes such as wind blowing branches. This results in a binary video where moving pixels are in the foreground and non-moving pixels are in the background.

2. Perform **Connected Components Analysis** on the foreground objects. The ball will not be the only thing moving. The players will also be moving, there may be birds which fly through the view of the scene. This results in a labelling of the independent binary regions, by grouping together pixels which are connected to each other by 4 or 8 adjacency.

3. The ball may be held by the player while the player is serving. Therefore we must also track the players. As the ball will become a connected component of the player in the video frames where the player holds the ball. To locate the players and the ball perform a **measure of rectangularity** and a **measure of elongatedness** on the binary objects. Humans are taller than they are wide, therefore a human will be represented by an elongated vaguely rectangular shape. To locate the ball, perform a **measure of circularity** on the binary objects. These measures assign values with how rectangular, elongated or circular the region is, and can be used for analysis.

4. To ensure only the ball is tracked and not other vaguely circular objects e.g. raquets, perform a **non-maxima suppression** on the results of the circularity measure. Tennis balls should be represented by near-perfect circles. Non-maxima suppression will suppress any results from detection which are not in the local maxima by comparing the results to their neighbours.

5. To further eliminate any false positives**, count pixels** associated with these maximally circular objects. Tennis balls are small objects, any large circles should be discarded.

6. The ball and players have been located. Perform a **mean shift tracking** operation on these located areas to represent them as histograms and track them by the direction in which the histogram is moving.

Find the court to track play:

1. With the tracked players and ball, the court should be translated to a view where it can be analysed by a computer. First, convert the video frames to **greyscale**.

2. Perform an **Otsu Thresholding** as the lines of the court are a bright white. This also includes the clouds/sky. It produces a binary image.

3. Perform a **Moravec Corner Detection** on this binary image which looks at the variation around a point and can therefore operate on the binary image. Perform **non-maxima suppression** on

this cornerness map to maintain only the strongest corners.

4. By taking the binary image's top-leftmost, top-rightmost, bottom-leftmost and bottom-rightmost strong corners, we have identified four feature points. Perform a **Perspective Geometric Transform** on these four feature points to obtain a birds-eye view of the court, where the tracked players and ball can be extrapolated onto this view to track the play

Track the basketball.

Dé Máirt 9 Nollaig 2025 22:44

1. This video is taking place in a crowded auditoriam where spectators may be making subtle movements. Perform a **Gaussian Mixture Model** on this video to obtain a binary video where the significant, clear movements are in the foreground and non-moving pixels are in the background.