Chapter 1

Edge point and Canny

Points in the image where brightness changes particularly sharply are often called edges or edge points.

Canny established the practice of choosing a derivative estimation filter by using the continuous model to optimize a combination of three criteria:

- Signal to noise ratio — the filter should respond more strongly to the edge at $x = 0$ than to noise.
- Localisation — the filter response should reach a maximum very close to $x = 0$.
- Low false positives — there should be only one maximum of the response in a reasonable neighbourhood of $x = 0$.

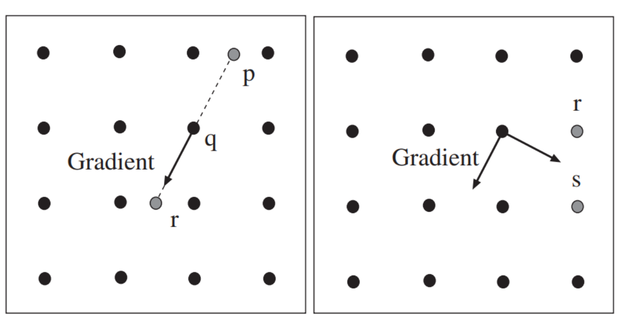
Once a continuous filter has been found, it is discretised. The criteria can be combined in a variety of ways, yielding a variety of somewhat different filters. It is a remarkable fact that the optimal smoothing filters that are derived by most combinations of these criteria tend to look a great deal like Gaussians — this is intuitively reasonable, as the smoothing filter must place strong weight on center pixels and less weight on distant pixels, rather like a Gaussian. In practice, optimal smoothing filters are usually replaced by a Gaussian, with no particularly important degradation in performance

Non maximum suppression

Obtaining edge points from gradient magnitude estimates involves a process called non-maximum suppression. The goal is to identify points along the gradient direction where the gradient magnitude is at a maximum.

The significant steps in non maximum suppression are:

- determining whether a given point is an edge point;
- and, if it is, finding the next edge point.



Left Side of Figure: Reconstructing Gradient Magnitude at Pixel q

The dots represent the pixel grid.

The algorithm is at pixel q, attempting to determine whether the gradient is at a maximum.

The gradient direction through q does not directly pass through any convenient pixels in the forward or backward direction.

Interpolation is performed to obtain the values of the gradient magnitude at points p and r.

Typically, linear interpolation is used. This means that the values at p and r are estimated by using the pixel values to the left and right of p and r, respectively.

If the value at q is larger than both interpolated values at p and r, then q is considered an edge point.

Right Side of Figure: Finding Candidates for the Next Edge Point (Given q as an Edge Point)

Assuming that q is an edge point, the search for candidates for the next edge point begins.

An appropriate search direction is perpendicular to the gradient direction at q.

Points s and t are considered as candidates for the next edge point.

Notice that, in principle, the algorithm doesn't need to restrict itself to pixel points on the image grid. The predicted position between s and t can be known, allowing interpolation to obtain gradient values for points off the grid.

This process of non-maximum suppression helps identify points along the gradient direction where the gradient magnitude is maximized, leading to a more accurate representation of edges in the image. The interpolation and search for candidates are crucial steps in refining the edge points and ensuring that the detected edges form meaningful chains.

```
While there are points with high gradient
that have not been visited

  Find a start point that is a local maximum in the
    direction perpendicular to the gradient
    erasing points that have been checked

  while possible, expand a chain through
    the current point by:
      1) predicting a set of next points, using
        the direction perpendicular to the gradient

      2) finding which (if any) is a local maximum
        in the gradient direction

      3) testing if the gradient magnitude at the
        maximum is sufficiently large

      4) leaving a record that the point and
        neighbours have been visited

      record the next point, which becomes the current point

  end

end
```

Chapter 2

Texture

Texture is a phenomenon that is widespread, easy to recognise and hard to define

There are three standard problems to do with texture

• Texture segmentation is the problem of breaking an image into components within which the texture is constant. Texture segmentation involves both representing a texture, and determining the basis on which segment boundaries are to be determined.

• Texture synthesis seeks to construct large regions of texture from small example images. We do this by using the example images to build probability models of the texture, and then drawing on the probability model to obtain textured images.

• Shape from texture involves recovering surface orientation or surface shape from image texture. We do this by assuming that texture “looks the same” at different points on a surface; this means that the deformation of the texture from point to point is a cue to the shape of the surface.

Analysis (and Synthesis) Using Oriented Pyramids

It deals with how images can be efficiently analyzed and synthesized using the Laplacian pyramid, which captures different scales with reduced redundancy. It also introduces oriented pyramids, an extension that adds information about image orientation, enabling detailed texture analysis.

Laplacian Pyramid:

1. **Computational Problem:**

   - Analyzing images using filter banks involves convolving an image with a large number of filters at different scales.
   - The Gaussian pyramid is an example of image analysis using a bank of filters, handling scale by subsampling the image after smoothing.

2. **Redundancy in Gaussian Pyramid:**

   - The Gaussian pyramid is redundant since each layer is a low-pass filtered version of the previous layer.
   - The Laplacian pyramid addresses this by storing only the errors in the prediction of the next finer scale layer.

3. **Laplacian Pyramid Representation:**
   - The Laplacian pyramid is a representation of different scales with lower redundancy, obtained by subtracting predictions from the Gaussian pyramid.
   - Each layer of the Laplacian pyramid can be seen as the result of a difference of Gaussian filters.

4. **Image Compression and Blending:**
   - Different levels of the Laplacian pyramid represent different spatial frequencies, making it useful for image compression.
   - Laplacian pyramids are employed for image blending.

5. **Synthesis from Laplacian Pyramid:**
   - It is easy to recover an image from its Laplacian pyramid.
   - The synthesis process involves reconstructing the Gaussian pyramid from the Laplacian pyramid and then obtaining the original image.

Oriented Pyramids:

1. **Limitations of Laplacian Pyramid:**
   - The Laplacian pyramid lacks information about orientation, crucial for reasoning about image texture.

2. **Oriented Pyramid Strategy:**
   - To address orientation, each layer of the Laplacian pyramid is further decomposed into components representing energy at distinct orientations.
   - The result is an oriented pyramid, providing detailed analysis of image texture.

3. **Design Considerations:**
   - Designing oriented pyramids involves considerations of the Fourier domain, with each layer's Fourier transform obtained by multiplying a difference of Gaussians.
   - Modification of the Fourier transform of the filter kernel is required to select orientations.

4. **Synthesis and Efficient Implementation:**
   - Synthesis in oriented pyramids should be easy. An efficient implementation of these pyramids is available.
   - Designing filters with oriented responses enables efficient reconstruction of layers and overall synthesis.

In summary, the Laplacian pyramid provides a representation of different scales with low redundancy, and the extension to oriented pyramids allows for detailed analysis of image texture, considering both scale and orientation. The design of filters plays a crucial role in achieving efficient synthesis and reconstruction.

Chapter 3

Clustering

Clustering is a technique that involves grouping similar data points together based on certain features or characteristics.

We can cluster in two ways:

Partitioning involves breaking down a large dataset into meaningful segments based on specific criteria, aiming to create coherent and internally associated subsets.

For example, we might:

– Decompose an image into regions which have coherent colour and texture inside them;

– Take a video sequence and decompose it into shots — segments of video showing about the same stuff from about the same viewpoint;

– Decompose a video sequence into motion blobs, consisting of regions that have coherent colour, texture and motion.

Grouping: Here we have a set of distinct data items, and wish to collect sets of data items that "make sense" together according to our model. Effects like occlusion mean that image components that belong to the same object are often separated.

Examples of grouping include:

– collecting together tokens that, taken together, forming an interesting object

– collecting together tokens that seem to be moving together.

Factors that postdate the main Gestalt movement:

• Proximity: tokens that are nearby tend to be grouped.

• Similarity: similar tokens tend to be grouped together.

• Common fate: tokens that have coherent motion tend to be grouped together.

• Common region: tokens that lie inside the same closed region tend to be grouped together.

• Parallelism: parallel curves or tokens tend to be grouped together.

• Closure: tokens or curves that tend to lead to closed curves tend to be grouped together.

• Symmetry: curves that lead to symmetric groups are grouped together.

• Continuity: tokens that lead to "continuous" — as in "joining up nicely", rather than in the formal sense — curves tend to be grouped.

• Familiar Configuration: tokens that, when grouped, lead to a familiar object, tend to be grouped together

Shot Boundary Detection

• Long sequences of video are composed of shots — much shorter subsequences that show largely the same objects.
• These shots are typically the product of the editing process.
• It is helpful to represent a video as a collection of shots; each shot can then be represented with a key frame. This representation can be used to search for videos or to encapsulate their content for a user to browse a video or a set of videos.
• Finding the boundaries of these shots automatically — shot boundary detection — is an important practical application of simple segmentation algorithms.
• A shot boundary detection algorithm must find frames in the video that are "significantly" different from the previous frame. Our test of significance must take account of the fact that within a given shot both objects and the background can move around in the field of view. Typically, this test takes the form of a distance; if the distance is larger than a threshold, a shot boundary is declared Algorithm:

```
For each frame in an image sequence

    Compute a distance between this frame and the
    previous frame

    If the distance is larger than some threshold,

      classify the frame as a shot boundary.

end
```

Background Subtraction algorithm

The Background Subtraction algorithm is used in scenarios where objects of interest appear on a relatively stable background, such as detecting items on a conveyor belt or counting cars on a road. The key idea is to segment the objects by subtracting an estimate of the background from the image and identifying significant changes.

One challenge is obtaining an accurate background estimate. Simply taking a static picture is not effective because backgrounds often change slowly over time due to factors like weather conditions or gradual environmental changes. Instead, a common and effective approach is to estimate background pixel values using a moving average.

In this method, the value of a background pixel is estimated as a weighted average of its previous values. Pixels from the distant past are given zero weight, and weights gradually increase. The moving average is designed to adapt to changes in the background, responding more quickly to rapid changes and gradually incorporating past values for slower changes.

The algorithm uses a filter to analyze changes over time, smoothing out rapid fluctuations and focusing on gradual shifts in the background. By suppressing fast changes and allowing slower ones, it distinguishes between moving objects and stable background, making it a reliable method to identify and separate foreground objects in dynamic environments.

Clustering and Segmentation by K-means

1. **Objective Function:**
   - Assume there are k clusters, each with a center c_i.
   - Each element, described by a feature vector x_j, is expected to be close to the center of its cluster.
   - The objective function aims to minimize the distance between elements and their assigned cluster centers.

2. **Algorithm Overview:**
   - The algorithm iterates between two main activities:
     - Given known cluster centers, allocate each point to the closest cluster center.
     - Given known allocations, compute new cluster centers by taking the mean of the points in each cluster.
   - The process starts with randomly chosen cluster centers and iterates until convergence.
   - This iterative approach seeks a local minimum of the objective function.

3. **Convergence and Guarantees:**
   - The algorithm is not guaranteed to converge to the global minimum.
   - It may not produce exactly k clusters unless modifications ensure each cluster has a non-zero number of points.

4. **Application to Image Segmentation:**
   - Challenge: Segments may not be connected and can be scattered widely in images.
   - Using pixel coordinates as features helps mitigate this issue.

- K-means can be applied with different values of k to search for an appropriate number of clusters.
- The result may need post-processing to improve segment connectivity.

5. **Limitation:**
- The algorithm's success depends on a good initialization, and it might converge to a local minimum.
- Choosing the right number of clusters (k) is often a trial-and-error process.

6. **Practical Considerations:**
- Commonly used for image segmentation and clustering data points.
- Can handle various types of features, making it versatile in different applications.

7. **Result Interpretation:**
- The final clusters represent groups of elements that are closer to each other in feature space.
- Applied iteratively, the algorithm refines cluster assignments and centers until convergence.

8. **Post-Processing for Image Segmentation:**
- To enhance segment connectivity, post-process the results, such as merging adjacent regions with similar characteristics.

In summary, K-means is a versatile clustering algorithm applied to various domains, including image segmentation, with considerations for initialization, convergence, and post-processing.

```
Choose k data points to act as cluster centers

Until the cluster centers are unchanged

    Allocate each data point to cluster whose center is nearest

    Now ensure that every cluster has at least
    one data point; possible techniques for doing this include .
    supplying empty clusters with a point chosen at random from
    points far from their cluster center.

    Replace the cluster centers with the mean of the elements
    in their clusters.

end
```