CMPT 414 review notes

1 Image Processing

In this section we outline methods that can be used to modify images. For our purposes we only concern ourselves with grayscale images. For color images, we can apply the same methods after transforming the color space into one that has separatable luminance and chrominance components (ex. YCbCr).

1.1 Histogram Transformations

An image histogram is constructed by the frequencies of pixel intensities. By modifying pixel intensities on a histogram-level we can alter and often improve the quality of images. The transformations are applied to the image histogram, but changes in histogram values reflect the mapping of pixel values in the image. We outline two methods below:

1.1.1 Linear Stretching

Sometimes the image histogram may only be occupying part of the intensity range (ex. mostly dark pixels or mostly light pixels). As a result, the image would have bad contrast.
One way to address this issue is to stretch the histogram so that it occupies the entire intensity range. To do this, we mark the top and bottom of the histogram (`a` and `b`, respectively) and for each pixel in the histogram, compute a proportion of where it stands between `a` and `b`. Based on these proportions, the histogram is then mapped to occupy the entire intensity range.
The algorithm is as follows. `p_0` and `p_m` are the minimum and maximum pixel intensities respectively.

```
function linear_stretch (img, a, b, p_0, p_m)
  for pix in img:
    if     pix <= a: pix = p_0
    elseif pix >= b: pix = p_m
    else:  pix = (p_m-p_0)/(b-a)*(pix-a)+p_0
end
```

One quick note is that we can set `a` and `b` to not be the max and min of the histogram, which then makes some pixels clip to the min or max.

1.1.2 Histogram Equalization

Histogram Equalization is another popular method that is often used for improving the constrast of the messages. The idea of Histogram Equalization is to transform the intensity histogram to be a uniform histogram. In practice, the resulting histogram will not be perfectly uniform.
In essence: by spacing out pixel intensities, regions in the image where there are lots of detail but only represented with a few intensities become easier to process.
The algorithm is as follows: `hist` is the original histogram of the image, and `n_factor` is the ratio between amount of pixels in the image and the levels of pixel intensities.

```
function hist_eq (img, hist, n_factor)
  for pix in img:
    pix = ceil(sum(hist[:p])/n_factor) - 1
end
```

1.2 Convolutions and filtering

A convolution is a linear and shift-invariant operator. Shift invariance implies that if $y(n)$ is the response to $x(n)$, then $y(n-k)$ is the response to $x(n-k)$. Linearity is defined in the usual manner. For our purposes, a 2D convolution is equivalent to filtering an image, where a filter is defined by a matrix that we call a mask.
For an image f and a mask t, a 2D convolution is defined as the following:

$$\begin{aligned} f * t &= \iint_{-\infty}^{\infty} f(x-x', y-y') \cdot t(x', y') dx' dy' \\ &= \sum_{x'=-M}^{M} \sum_{y'=-N}^{N} f(x-x', y-y') \cdot t(x', y') \end{aligned}$$

For the discrete case, the mask has to be in dimensions of $(2M+1) \times (2N+1)$.

1.2.1 Cross-Correlations

A cross-correlation between f and t is defined as

$$R_{ft}(x, y) = \sum_{x'=-M}^{M} \sum_{y'=-N}^{N} f(x+x', y+y') t(x', y')$$

The above definition is not invariant to the magnitude of f. A normalized R_{ft} is needed:

$$R_{ft} = \frac{\sum_{x',y'} f(x+x', y+y') t(x', y')}{\sqrt{\sum_{x',y'} f(x+x', y+y')^2 \cdot \sum_{x',y'} t(x', y')^2}}$$

1.2.2 Mask Design

Masks are usually matrices whose values are designed in a specific way to accomodate different uses.
In computer vision, we usually use masks that are symmetric. This implies that we do not need to concern ourselves with demarcating convolutions and cross-correlations.
When the same mask is applied twice (for example, applied onto itself), we will see that the mask grows in size. For example:

$$(1\ 1\ 1) * (1\ 1\ 1) = (1\ 2\ 3\ 2\ 1)$$

It is implied that mask values are normalized, so the previous example should really be

$$\left(\frac{1}{3}\ \frac{1}{3}\ \frac{1}{3} \right) * \left(\frac{1}{3}\ \frac{1}{3}\ \frac{1}{3} \right) = \left(\frac{1}{9}\ \frac{2}{9}\ \frac{3}{9}\ \frac{2}{9}\ \frac{1}{9} \right)$$

For a target 2D mask, it may be worthwhile to decompose into a kronecker product of two 1D masks, For example:

$$\begin{pmatrix} 1 & 0 & -1 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix}$$

A better visualization is the following, each entry in the matrix is the product of the row value and column value:

	1	**0**	**-1**
1	1	0	-1
2	2	0	-2
1	1	0	-1

The merit in decomposing the mask is to do with the time complexity of applying a filter. Each entry in the mask is used in a multiplication once for each pixel in the image (suppose the image is $N \times N$). Thus for a 2D $k \times k$ mask the time complexity will be $O(k^2N^2)$. However, for two 1D $k \times 1$ masks the time complexity will only be $O(kN^2)$.

1.2.3 Frequency Analysis

Images in the (x, y) domain can be decomposed into sums of spatial frequencies in frequency domain via an integral transform mechanism (ex. DCT, ADST, etc). Spatial frequencies can be categorized into high frequencies or low frequencies. Applying a filter that attenuates (reduces) high frequencies while passing low frequencies is called a low-pass filter. The opposite is called a high-pass filter. The demarcation for a filter to distinguish between low and high frequency is called a cut-off frequency.
Suppose we have an integral transform $\mathcal{I}$, there are important properties to exploit. In the foremost, a convolution in image space is translated to a multiplication in frequency space:

$$\mathcal{I}(f * t) = \mathcal{I}(f) \times \mathcal{I}(t)$$

1.3 Smoothing

Smoothing is a low-pass filtering operation. It is also called de-noising. While smoothing will remove noise, it will also remove the sharpness of the image, resulting in a blurred effect. In a sense, de-noising the image is achieved by smearing the image.

1.3.1 Unweighted Averaging

One naive approach to blur an image is to use a $k \times k$ mask that with all the same entries (i.e. all 1's; keep in mind that normalization is implied). For example, such a 4×4 mask would look like this:

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

1.3.2 K-nearest Neighbor Averaging

This method involves investigating a $m \times m$ neighborhood around each image pixel. K neighbors that are nearest to the centre image pixel value are taken and their average is preserved. This method is edge-preserving, but non-linear.

1.3.3 Median Filtering

This method also involves investigating a $m \times m$ neighborhood, but instead the median of the neighborhood is taken. This method is also very effective against salt and pepper (S&P) noise.
This method is also edge-preserving and non-linear, and additionally it does not preserve corners. One way to mitigate this issue is to choose a different set of values to observe in the neighborhood. For example, in a cross shape instead of the entire neighborhood.

1.3.4 Gaussian Filtering

Gaussian filtering is a popular method for denoising images, it is also used as a primitive in many other applications in computer vision. This is because the operation resembles optical blur, and a Gaussian mask has good mathematical properties.
The filter is parametrized with two parameters (σ_x^2, σ_y^2), but we will treat the filter as if it has only one parameter $\sigma^2 = \sigma_x^2 = \sigma_y^2$.
One important mathematical property of a Gaussian is that the integral transform of a Gaussian is still a Gaussian. For a Gaussian function $G(x)$ with variance σ^2, the integral transform response of the Gaussian $\mathcal{I}(G(x))$ has variance σ_f^2 where the relationship between σ and σ_f is described as $\sigma \cdot \sigma_f = (2\pi)^{-1}$. As such, the larger σ is, the lower σ_f is, which gives a lower cut-off frequency.

1.4 Sharpening

Sharpening is a high-pass filtering operation. It is the opposite of smoothing and as such it is sometimes called de-blurring.

1.4.1 Laplacian-based Methods

The Laplacian (2nd derivative) sharpening operator is defined as $f - \nabla \cdot \nabla f$. The Laplacian $\nabla \cdot \nabla f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$ is derived as follows:

$$\begin{aligned}\frac{\partial^2 f}{\partial x^2} &= f(x+1,y) - f(x,y) - (f(x,y) - f(x-1,y)) \\ &= f(x+1,y) + f(x-1,y) - 2f(x,y) \\ \frac{\partial^2 f}{\partial y^2} &= f(x,y+1) - f(x,y) - (f(x,y) - f(x,y-1)) \\ &= f(x,y+1) + f(x,y-1) - 2f(x,y)\end{aligned}$$

As such, $\nabla^2 f(x,y) = f(x+1,y)+f(x-1,y)+f(x,y+1)+f(x,y-1)-4f(x,y)$, which is similar to a blurred version of the image. The mechanism that a laplacian operator operates on is simply to subtract the noise from the image by subtracting a noisy version of the image: $\hat{f}(x,y) = 5f(x,y)-[f(x+1,y)+f(x-1,y)+f(x,y+1)+f(x,y-1)]$. This corresponds nicely to the following mask:

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

A common variation on this operator is to give weight to the laplacian, so the operator is defined as $f - w\nabla \cdot \nabla f$. While this method is effective at sharpening the image, it also creates lots of artefacts and noise.

1.4.2 Unsharp Masking

In a similar spirit to the laplacian operator, unsharp masking (USM) operates by adding a scaled and inverted version(inversion is the same as subtraction) of the blurred image to the original. The blurring mechanism is to simulate the noise model using a blurring mechanism in the image instead of using a derivative based method. USM generally performs better than the Laplacian operator, but it will take more time to run as it is more than just one simple filter.

2 Image Moments

For an image $f(x,y)$, the $(p,q)^{\text{th}}$ order moment is defined as the following:

$$\begin{aligned}m(p,q) &= \iint x^p y^q f(x,y) dx dy \\ &= \sum_x \sum_y x^p y^q f(x,y)\end{aligned}$$

Suppose we have a segmented image represented by a binary image B, image moments can tell us different statistics about the image.
$m(0,0)$ for the image tells us the amount of pixels that are 1 in B, or the area of 1's:

$$A = \sum_i \sum_j B[i,j]$$

The first moment in either axis helps us identify the centroid of 1's:

$$\bar{x} = \frac{1}{A}\sum_i \sum_j j \cdot B[i,j] \quad \bar{y} = \frac{1}{A}\sum_i \sum_j i \cdot B[i,j]$$

The axis of the least 2nd moment is the axis of elongation (orientation) for 2D regions.

3 Template Matching

Similarity between $M \times N$ image f and a $k \times s$ template t can be measured by their Euclidean distance, but the sum of squares itself will do:

$$\begin{aligned}&d^2(x,y) \\ =&\sum_{k,s}[f(x+k,y+s) - t(k,s)]^2 \\ =&\sum_{k,s}[f(x+k,y+s)^2 - 2f(x+k,y+s)t(k,s) + t(k,s)^2]\end{aligned}$$

If a perfect match is found at (x',y'), then $d^2(x',y') = 0$. Our goal is to minimize $d^2(x,y)$. $f(x+k,y+s)^2$ is independent from t, and $t(k,s)^2$ is a constant. Therefore we only need to focus on maximizing $\sum_{k,s} f(x+k,y+s)t(k,s)$ to minimize $d^2(x,y)$, which is in effect looking for maximas in a cross-correlation.

4 Edge Detection

We define edges as intensity discontinuities in images. Using edge information is useful for many computer vision applications. We will discuss many methods of edge detection in this section.

4.1 Gradient-Based Methods

In this category of methods, the image gradient is computed in one way or another to derive edge information.

$$\frac{\partial f}{\partial x} = f(x+n,y) - f(x,y) \quad \frac{\partial f}{\partial y} = f(x,y+n) - f(x,y)$$

$$S = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2} \quad \theta = \arctan\left(\frac{\partial f}{\partial y} \Big/ \frac{\partial f}{\partial x}\right)$$

Usually, $n = 1$ for computing the partial derivatives. In this context, S is the gradient magnitude, and θ is the directional information of the gradient. All of these are computed over the entire image, so there's a map of all of these values corresponding to each pixel.

4.1.1 Gradient Thresholding

Simply put, if the magnitude S is greater than a threshold t, then it is an edge pixel, and an edge map can be generated from it.

4.1.2 Roberts Operator

The roberts operator computes S and θ differently. Instead of using $\frac{\partial f}{\partial x}$ and $\frac{\partial f}{\partial y}$, it generates two filtered images with the following two masks:

$$G_x = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \qquad G_y = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

S and θ are derived by the following:

$$S = \sqrt{(G_x * f)^2 + (G_y * f)^2} \quad \theta = \arctan\left(\frac{G_y * f}{G_x * f}\right) - \frac{3\pi}{4}$$

The rest follows a simple thresholding procedure.

4.1.3 Prewitt and Sobel Operators

Prewitt uses 8 operators for each of the 8 ordinal-cardinal directions. The 0° and 45° operators are as follows:

$$\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 0 & 1 & 1 \\ -1 & 0 & 1 \\ -1 & -1 & 0 \end{bmatrix}$$

This method includes providing directional information and uses local averages to reduce noise. However, the edge width can be greater than 1, which means there isn't a single output.
The Sobel operator is similar to Prewitt's, except it places a stronger emphasis on the weighting of the direction each mask is meant for. Below are the corresponding masks to compare with their Prewitt counterparts:

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{bmatrix}$$

The 0° and 90° Sobel operators are often used to generate gradient information.

4.2 Laplacian-Based Methods

Instead of working with the first derivative, we can work with the second derivative (which is a laplacian) to derive edge information.

4.2.1 Zero Crossings

The laplacian edge operator we describe here is different from the sharpening operator. Namely, the laplacian edge operator is

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Whereas the laplacian sharpening operator has a midle entry of 5 instead of 4.
After applying the operator, we look for zero-crossings in the resulting image to determine where the edges are. Zero crossings are simply where there is a negative value and a neighboring positive value.

4.2.2 Marr's Operator

Marr's Operator applies a gaussian blur G to the image f before calculating the laplacian to find zero crossings:

$$\nabla^2(G(x, y) * f(x, y))$$

Due to a property for convolutions, the above is equivalent to

$$(\nabla^2 G(x, y)) * f(x, y)$$

Thus operator is also called a laplacian of gaussian (LOG) operator. The gaussian's σ can be tuned for different spatial frequencies.
Additionally, a laplacian of gaussian can be approximated by the difference between two gaussians, called a difference of gaussians (DOG).

4.3 Marr's Edge Operator

Given an image f(x,y), edges can be found by locating the zero crossings from $D^2(G(x, y) * f(x, y))$, where G is a Gaussian function and D^2 is the second derivative. Derivative rule for convolution: $D^2(G * f) = D^2 G * f$.

4.4 Canny Edge Operator

After discussing previous edge detection operators, we collect some thoughts to approach another edge operator.

4.4.1 Criteria for Good Edge Detection

- Always find real edges, not false ones
- Good localization, where found edges are as close to the original as possible
- Unique Output

4.4.2 Stages in Canny Edge Detection

1. **Smoothing**

   Apply Gaussian Blur to the image $I' = G * I$.

2. **Gradient Operator**

   Use the 0° and 90° Sobel operators to generate $\frac{\partial I'}{\partial x}$ and $\frac{\partial I'}{\partial y}$. Using these, generate S and $theta$.

3. **Non-maximum Suppression**

   Non-maximum suppression thins out the edges that result from the gradient operator.

   For each pixel p in S, examine two neighbors along the direction of θ. The pixel will be suppressed (set to 0) if p is not the largest value among its neighbors, and the directions of its neighbors don't differ from p by more than 45°.

4. **Double Thresholding**

   Double thresholding determines the strength of each pixel, whether it's weak, strong, or irrelevant.

   There are two thresholds, a high threshold Th_h to tell apart strong and weak edge pixels. A low threshold Th_s is then used to tell apart weak and irrelevant pixels. Strong pixels are given a value, and weak pixels are given another value. The resulting matrix will only have three unique values.

5. **Hysteresis Tracking**

   This is used to determine if weak edge pixels are going to be transformed into strong edge pixels and included in the final edge map. Otherwise they are discarded.

   If a weak edge pixel has at least one strong edge pixel surrounding it, then it is turned into a strong pixel.

   The final edge map is defined as the map of all the strong edge pixels.

5 Regions and Segmentation

A region is a group of connected pixels with similar properties in an image.

5.1 Image Segmentation

Image segmentation is a process to divide an image into regions. It can be done in a manner that is region-based, edge-based, or in a hybrid approach.
Some of the methods for Image Segmentation are as follows

- Thresholding
- Blob colouring (for binary images)
- Split and Merge
- K-Means
- Mean Shift

5.2 Gestalt Psychology and Human Perception

Grouping is the key to visual perception. The Law of Pragnanz in Gestalt psychology states that people perceive ambiguous or complex images as the simplest form possible.
Here are some of the heuristics that can be used to group items together.

- Proximity
- Similarity
- Common Fate
- Common Region
- Parallelism
- Symmetry
- Continuity
- Closure
- Familiar Shapes

6 Texture Analysis

Textures are defined as repeated patterns of local intensity variations, and it depends on the image resolution. There are three issues related to texture analysis:
There are three main tasks in texture analysis:

- texture classification
- texture segmentation
- shape from texture

6.1 Statistical Methods

6.1.1 Spatial Gray Level Dependence Method

The spatial gray level dependence method utilizes a gray level co-occurence matrix P. It is a $M \times M$ where M is the amount of gray levels. P serves as a tally of pixel-value pairs that follow a predefined pixel structure $\mathbf{d}$. Due to the particularity of $\mathbf{d}$, P is not symmetric. P is also normalized, and as such treated as a pmf.
$\mathbf{d}$ is defined as a pair of integral values (a, b) which is to be interpreted as "starting with pixel intensity i, move a units to the right then b units down to get to pixel intesity j". In other words, a valid pixel pair satisfying $\mathbf{d}$ is $f(x, y) = i$ and $f(x + a, y + b) = j$.
With P, we can define some related measures:

- Entropy: $-\sum_{i,j} P[i, j] \log P[i, j]$

  Entropy is the level of randomness in the data that is being processed. Entropy is highest when all the elements in P[i,j] are the same. In terms of images, this means that there is no preferred gray level.

- Energy: $\sum_{i,j} P[i, j]^2$

  Energy simply defined as the sum of all of the matrix values squared

- Contrast: $\sum_{i,j}(i-j)^2 P[i,j]$

  Contrast is a measure of variation in an image. The formula places greater weighting on entries away from the diagonal (due to the $(i-j)^2$ term), thus a high contrast would mean more pixel pairs satisfying **d** will be constructed with widely different gray level intensities.

- Homogeneity: $\sum_{i,j} \frac{P[i,j]}{1+|i-j|}$

  Homogeneity is an opposite measure to Contrast. It places greater weighting on entries on the diagonal (due to the $1+|i-j|$ denominator). A highly homogenous image would mean more pixel pairs satisfying **d** are constructed with similar gray level intensities.

One major downside of this method is that many **d**'s have to be tried before arriving at a good one. It is also not good for textures that are composed of primitives of a certain shape/structure.

6.1.2 Gray Level Run Length Method

The gray level run-length method utilizes a gray level run matrix P_θ, where $P_\theta(i,j)$ is the number of times an image contains a run of length j for pixels with value i in a certain direction of θ. Long runs can be expected for images with coarse texture.
With P_θ, we can define some related measures. We define the following as a normalization factor for our measures:

$$T_\theta = \sum_{i=1}^{N_g}\sum_{j=1}^{N_r} P_\theta(i,j)$$

where N_g is the number of gray levels and N_r is the number of run lengths

- Short run emphasis inverse moments

$$\frac{1}{T_\theta}\cdot\sum_{i=1}^{N_g}\sum_{j=1}^{N_r}\frac{P_\theta(i,j)}{j^2}$$

  This measure emphasizes short runs of a gray level image by dividing by j^2. Short run emphasis is large when there are lots of short runs of the same intensity.

- Long run emphasis moments

$$\frac{1}{T_\theta}\cdot\sum_{i=1}^{N_g}\sum_{j=1}^{N_r} j^2 P_\theta(i,j)$$

  This measure emphasizes long runs of a gray level image by multiplying by j^2. Long run emphasis is large when there are lots of long runs of the same intensity.

- Gray level nonuniformity

$$\frac{1}{T_\theta}\cdot\sum_{i=1}^{N_g}\left(\sum_{j=1}^{N_r} P_\theta(i,j)\right)^2$$

  The sum inside of the brackets is the total number of runs for a certain gray level value. This value is large if the runs are not evenly distributed across different intensities.

- Run length nonuniformity

$$\frac{1}{T_\theta}\cdot\sum_{j=1}^{N_r}\left(\sum_{i=1}^{N_g} P_\theta(i,j)\right)^2$$

  The sum inside the brackets is the total number of runs with a certain run length at a certain gray level. This value is large if intensities are not evenly distributed across different run lengths.

- Inverse of weighted average run length

$$T_\theta\left(\sum_{i=1}^{N_g}\sum_{j=1}^{N_r} j\cdot P_\theta(i,j)\right)^{-1}$$

  This measure puts more weight into longer run length.

6.2 Structural Methods

The basic assumption in structural methods is that textures are composed of primitives known as "texels".

- Works better than statistical methods if the primitives can be detected
- Geometric properties (size, elongation, orientation, etc.) of the primitives are important features
- Spatial relationship of the primitives based on the co-occurrence of these primitives can be further explored

For example, if we use an edge-based method, then the primitives are edges. The properties are orientation, gradient, and density. The spatial relationship is edge separations.

6.2.1 Tamura's Texture Measures

Tamura's features are based on psychophysical studies of the characterizing elements that are perceived in textures by humans. Some of those features include"

- Coarseness
- Contrast
- Directionality

7 Classical Hough Transform

The Classical Hough Transform is good for detecting lines and curves. It exploits the property that for analytical curves, dual spaces can be formed.

For detecting a line, the most naive parametrization of a line is $y = mx + c$. Suppose we have an (x, y) space and a (m, c) space. The line $y = mx + c$ in (x, y) space corresponds to a point in the (m, c) space. Similarly, a line in (m, c) space corresponds to a point in (x, y) space. There are at least two points needed in (x, y) space to identify a line, and as such two corresponding lines in (m, c) will intersect and form a point pinpointing the parameters (m, c) needed to form a line in (x, y) space.

The disadvantage of this parametrization is that a vertical line have $m = \infty$, and that would imply a huge (m, c) space.

7.1 Voting Algorithm

The hough transform uses a process of voting for potentional matches of the geometric object needed, and by indentifying the most voted parameters the object in question can be detected. The following outline follows line detection with the parametrization $y = mx + c$

1. Form a voting space with max and min values of c and m
2. For each edge point (x, y) with gradient magnitude S greater than a set threshold, increment all points in the voting space that satisfies $c = -xm + y$
3. Identify maxima and peaks in the voting space to find lines in the original image

7.2 Polar parametrization of Lines

As mentioned before, vertical lines cause $m = \infty$, which makes implementing a voting space difficult. As such, we can reparametrize a line to be following polar coordinates instead. A line will be described by the tangent line to a circle of radius ρ at angle θ. The relationship between x, y, ρ, θ is described as the following:

$$\rho = x \cos\theta + y \sin\theta$$

Depending on the range of ρ and θ, we can have $0 \leq \theta \leq 2\pi$ with $0 \leq \rho$. Alternatively, we can also have $0 \leq \theta \leq \pi$ with ρ being positive or negative.

Because images are discrete entities, the margin of error grows the bigger as a point (x, y) on a line goes further away from the origin: $\delta\rho \approx d\delta\theta$. One way to mitigate this issue is to perform this parametrization hierarchically by splitting the original image into quadrants and shifting the center point to minimize ρ.

7.3 Gradient Information

Using gradient information can be helpful because now a vote only needs to be casted in the direction of the gradient and not over all possible points.

7.4 Discussion

Hough transforms can recognize any curve of the form $f(\mathbf{x}, \mathbf{a}) = 0$, where $\mathbf{a}$ is a parameter vector. For example, for a circle defined as

$$(x - a)^2 + (y - b)^2 = r^2$$

There are three parameters and as such the parameter space will be 3-dimensional.

The merit in using the hough transform is that it works for broken lines and curves, and that it is insensitive to noise. However, with increasing amount of parameters it becomes computationally expensive to execute.

8 Generalized Hough Transform

While classical Hough Transforms are ideal for analytical curves, it is unable to work with arbitrary curves. Generalized Hough Transform (GHT) works by detecting objects based on the shape of its boundary. It is a generalized form of template matching. It contains two components: Modeling and Matching.

8.1 Modeling: R-Table Generation

A template that is given to GHT is transformed into an R-table. An R-table is a lookup table of differences from edge points to a reference point. An R-table is built in the following way:

1. Determine template centroid (x_c, y_c) as a reference point.
2. Initialize R-table as Q bins, where Q is the amount of intervals to divide $0°$ to $360°$ into.
3. For each edge point (x, y) in the template, derive its gradient angle and compute ϕ, which is the index of the bin that the entry belongs to.
4. place $(\Delta x, \Delta y) = (x_c - x, y_c - y)$ in bin ϕ,

In the original formulation, the entries are (r_i, θ_i) values that denote distances and angles from the centroid. Storing simple differences allows for simpler calculations for reconstruction of centroid locations when there is a scale s and rotation θ:

$$\begin{bmatrix} x_c \\ y_c \end{bmatrix} = s \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}$$

8.2 Matching: Voting Algorithm

After the R-table is generated, the voting algorithm is as follows:

1. Generate edge map and gradient angles of the source image
2. Initialize the Hough Space with the image dimensions
3. For each edge pixel in the edge map:
    - compute ϕ to determine the bin to look up in the model (with compensation for rotation)
    - For each entry in bin ϕ:
        - Compute a candidate location (with compensation for rotation and scaling) for the model
        - Cast a vote in the candidate location if it does not clip out of the image

8.3 Discussion

GHT is only able to detect with one orientation and scale at a time, which is kind of inefficient. Peak detection with the votes is also non-trivial. It can be used in multiple stages to construct more complex contours.

9 Representations of 2D Geometric Structures

9.1 Boundary Representations

9.1.1 Polyline

A polyline representation can be used to apporximate curves and contours accurately, it is also known as "vertex-based shape coding". It consists of an ordered array of (x_i, y_i) vertices that approximate the true contours, where each vertex denotes the next vertex in a closed loop in clockwise fashion.
The algorithm for curve approximation using polyline involves iteratively going through vertices, and adding new vertices if the maximum distance between the polyline and the shape is greater than T_{max}.
Applying rotation on for a polyline representation requires applying a rotation matrix on every point of the representation.
For a closed polyline representation of a contour, its area is defined as

$$\frac{1}{2}\sum_{i=0}^{n-1}(x_{i+1}y_i - x_i y_{i+1})$$

9.1.2 Chain Code

Chain code is an economic coding method where contours are approximated with a chain of lines that either go in the 4 or 8 directions (called 4-neighbors or 8-neighbors) in a counterclockwise fashion. Each direction is enumerated from 0 to 3 or 7, and a chain code is represented as a list of these directions.
The derivative of the chain code is equivalent to the difference of the chain code at a position with its previous position modulo 4 or 8.
Applying rotation for chain coding is easily achieved by uniformly adding a rotation value to each value in the chain-code.
For the area of an enclosed by a 4-neighbor chain code, it can be computed in the following manner:

> For each element of the chain code in counterclockwise order, there are 4 cases for each element:
>
> - Case 0: area = area - y
> - Case 1: y = y + 1
> - Case 2: area = area + y
> - Case 3: y = y - 1

9.1.3 Curvature Scale Space (CSS)

A curvature scale space (CSS) is a representation that identifies objects by inflection points (ie. zero crossings) on their contour. The location of the inflection points are expressed as a function of the arclength of the contour.
Additionally, The CSS of a contour is computed multiple times, each time after a gaussian filtering is applied to the arc-length function with increasing σ, until no more inflection points are found.
With more filtering, weaker details are preserved and as such less inflection points will be seen. The inflection points that last the longest often point toward the most prominent features.
After computing CSS with all of the increasing σ, we obtain a graph that has many curves that grow upwards. The change in position in each curve as it grows upwards reflects the change of location of its inflection point as greater σ is applied.
The merit in this representation is that it is scale invariant (inflection points are parametrized by arc-length), rotation invariant (the CSS of a item and its rotated version are the same but rotated as well), and it is robust to noise due to multiple stages of filtering when constructing the model.

9.1.4 Shape Context Descriptor

A shape context descriptor is a compact representation where the template image (containing its boundaries) is separated into partitions (bins). The number of points that fall within each bin forms a distribution.

To match an object with Shape Context Descriptors, solve for least cost assignment to get correspondences, and then use a template given by the correspondences.

9.2 Region Representations

9.2.1 Spatial Occupancy Array

This method is simple and results in a sparse array.

9.2.2 Axis-based Representations

This method involves putting images on a grid and figuring out which x values correspond with a specific y value of the image (vice versa for x-axis).
The representation is a list of lists. Each list starts with the value at the primary axis, and then each pair of values that comes after is the starting and end value of the other axis (ex. $(y, x_{in}, x_{out}, x_{in}, x_{out})$).
This method is inefficient for specific cases and usually requires a combination of both X- and Y-axis representations.

9.2.3 Quad-trees

Using quad trees is a space efficient method where more detail is added where needed. It is a hierarchical tree representation, but it depends on the chosen grid size. Quad tree representations also can't be easily shifted or scaled.
A quad tree is a complete 4-ary tree where node is either a grey, white, or black pixel. Black pixels denote the region is contained, white pixels denote that it is not a region, and only grey pixels have children.
To form a Quad tree representation of a region, we can start at an intermediate level, and apply a split and merge technique.
For splitting, the representation takes a particular node and colors it grey. Then the node forms 4 children and colors each accordingly based on the region at that location.
For merging, the intersection of two quad trees is performed. Here we outline the procedure for intersecting two quad trees, which takes as input two nodes R_1, R_2 (starting as root nodes, this recursively applies to the whole tree). If either node is white, then that color is white. If either is black, return the other node. Note that black has higher precedence than white. Otherwise if both nodes are grey pixels, then the intersection between all of their 4 children are computed in the same manner.

9.2.4 Medial Axis Transform (MAT)

With a given boundary, a medial axis transform can effectively compute the skeleton of the region.
The idea for how a representation works is that for each point p, if there are more than one equidistant neighbors on the region boundary, it is a medial axis and as such part of the region's skeleton.
The construction of such skeleton is as follows: At each point i in the region, find the largest disk centered at i and only contained by the region. A point p is maximal if its disk is not completely contained within another point i's disk. The union of all the disks represents the object.
While generating skeletons of regions is very helpful, this method is noise sensitive.

9.3 Distance Measures

For different n-dimensinoal vectors, it is often useful to discuss the distance between those vectors. We outline some key distance measures:

- L_p distance

  In L_p space, the distance between between two vectors $\mathbf{a}$ and $\mathbf{b}$ is defined as

  $$||\mathbf{x}||_p = \left(\sum_i |x_i|^p\right)^{\frac{1}{p}}$$

  where $\mathbf{x} = \mathbf{a} - \mathbf{b}$. If we are talking about a single vector, then the distance is considered a p-norm.

- City Block distance (Manhattan)

  The city block distance is a particular case of a L_p distance measure with $p = 1$. Thus it is also called a $L1$ distance.

  $$d_4(\vec{p}_1, \vec{p}_2) = |x_1 - x_2| + |y_1 - y_2|$$

- Euclidean distance

  The Euclidean distance is a particular case of a L_p distance measure with $p = 2$. Thus it is also called a $L2$ distance.

  $$d_e(\vec{p}_1, \vec{p}_2) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

- Chebyshev (Chess Board)

  The Chebyshev distance between two vectors is defined as the following:

  $$d_8(\vec{p}_1, \vec{p}_2) = max\{|x_1 - x_2|, |y_1 - y_2|\}$$

  It is also called a chess board because a distance of one is defined by any direction a king piece can move on a chess board.

9.4 Errors Measures

These error measures are commonly used and can also be used as loss functions. The following measures utilize a ground truth vector $\mathbf{y}$ and a prediction vector $\hat{\mathbf{y}}$

- Mean Square Error (MSE)

$$MSE(\hat{\mathbf{y}}, \mathbf{y}) = \frac{1}{n}\sum_{t=1}^{n}(\hat{y_t} - y_t)^2$$

- Mean Absolute Error (MAE)

$$MAE(\hat{\mathbf{y}}, \mathbf{y}) = \frac{1}{n}\sum_{t=1}^{n} \mid \hat{y_t} - y_t \mid$$

- Root Mean Square Error (RMSE)

$$RMSE(\hat{\mathbf{y}}, \mathbf{y}) = \sqrt{\frac{1}{n}\sum_{t=1}^{n}(\hat{y_t} - y_t)^2}$$

9.5 Feature Space

Often, objects can be represented with a series of tokens. Every token could represent some sort of feature such as position, color, texture, motion vector, size, or orientation.
A feature space is the choice of features and how they are quantified, where each token is represented by a point.
Token similarity is defined by a feature vector, which is the distance between points.

9.6 K-means Clustering

K-mean clustering is a method of classification of points in a feature space. effectively finding clusters of points. Given an input K for the amount of clusters that should be found, the algorithm is as follows:

1. Partition the N points in the feature space into K intial sets. Calculate the initial set centroids $m_1, ..., m_k$.
2. While $m_1, ..., m_k$ has not stabilized, do:
   (a) Assign each of the N points to their nearest m_i, making sure that no cluster is empty
   (b) Recompute mean m_i of each cluster

Effectively, for each cluster C_i, the k-means algorithm carries out gradient descent to minimize

$$\sum_i \sum_{j \in C_i} ||x_j - \mu_j||^2$$

A sample use case with K-means is to identify the most important colors in an image and apply color segmentation.

9.7 Mean Shift Algorithm

One fatal issue with K-means cluster is that the number of clusters needs to be known beforehand. Mean shift locates maxima of a density function, given discrete data sampled from that function locally at the current estimates.
For each point, their original position is taken as the initial estimate of the mean x. A kernel function $K(x - x_i)$ is applied at x against all x_i points in a neighborhood. A new weighted mean is computed as the following

$$m(x) = \frac{\sum_{x_i \in N(x)}[K(x_i - x) \cdot x_i]}{\sum_{x_i \in N(x)} K(x_i - x)}$$

The estimate x is now set as $m(x)$, and this process continues until convergence. The difference $m(x) - x$ is known as the mean shift vector, and it is proven to always point toward the maximum increase in density.

9.7.1 Applications

Mean Shift can be used for Edge Preserving Smoothing and Image Segmentation.
For Image Segmentation, the initial x's can be some random pixels from the grid. The feature space is the Space-Range domain, where space is the coordinates of each pixel, and range is the grayscale or color vector of the pixel. The kernel function would be the product of the two kernel functions $K_s(x_i^s - x^s) \cdot K_s(x_i^r - x^r)$ for each part of the domain, one for space and one for range.
After Mean Shift filtering, modes can be merged if they are closer than h_s in the Space domain and h_r in the Range domain. We can also merge regions that are too small.

9.8 Gaussian Pyramid

A Gaussian pyramid contains successively lower resolutions of an image.The process of generating the next image in the pyramid from the image before it is called a REDUCE operation. Both resolution and density are decreased.

$$G_l(i,j) = \sum_{m=-2}^{2}\sum_{n=-2}^{2} w(m,n)G_{l-1}(2i+m, 2j+n)$$

The function $w(m,n)$ is called a generating kernel (which is usually a matrix), and it has to satisfy the followng constraints:

1. Separable: $w(m,n) = \hat{w}(m)\hat{w}(n)$
2. Normalized: $\sum \hat{w}(m) = 1$
3. Symmetric: $\hat{w}(m) = \hat{w}(-m)$
4. Equal Contribution: $a + 2c = 2b$

After combining the second and fourth constraints, we have the following:

$$a = free$$
$$b = 1/4$$
$$c = 1/4 - a/2$$

The inverse of the REDUCE operation is called the EXPAND operation. This operation increases the size of an image from $M + 1$ to $2M + 1$. Let $G_{l,k}$ be the image obtained by applying EXPAND to G_l k times, then $G_{l,0} = G_l$ and

$$G_{l,k}(i,j) = 4 \sum_{m=-2}^{2} \sum_{n=-2}^{2} w(m,n) G_{l,k-1}(\frac{i+m}{2}, \frac{j+n}{2})$$

In this expression, only terms for when $(i + m)/2$ and $(j + n)/2$ are integers contribute to the sum $G_{l,k}(i, j)$.
Note that $G_{l,1}$ is the same size as G_{l-1}, and $G_{l,l}$ is the same size as the original image.
When EXPAND is applied to an image l times, $2^l - 1$ points are made between each pair of neighbouring samples.

9.9 Laplacian Pyramids

With a series of Gaussian Pyramid images, the corresponding laplacian pyramid images can be easily derived with the following algorithm:
For values l from 0 to N, the laplacian pyramid image L_l is defined as $G_l - G_{l+1,1}$, with $L_N = G_N$. In essence, each laplacian pyramid image is a difference of Gaussian pyramid images.
An important feature of the Laplacian pyramid is that the original image can be found by summing up pyramid levels. That is:

$$G_0 = \sum_{l=0}^{N} L_{ll}$$

Due to the fact that laplacian images have this property, and each EXPAND operation effectively subtracts an average of neighboring values, the image data is compressed with the pixel to pixel correlation removed and values shifted toward zero. This effectively reduces the information intropy of each image and thus reduces transmission bandwidths.

10 Representations of 3D Structures

10.1 Surface Representations

10.1.1 Polyhedral Surfaces

The winged-edged representation includes the following:

- face node: face characteristics (surface, normal, reflectance, color, etc.)
- edge node: topological information for faces, edges, and vertices, as well as additional information
- vertex node: 3D vertex location

The Theorem of Euler states that for every polyhedron, V-E+F=2, where V is the number of vertices, E is the number of edges, and F is the number of faces.
Also, there are B-splines, point clouds, and 3D (polygonal) meshes.

For shape distribution, there are 3 steps.

1. Distance is measured between two random points on the surface
2. A 1D histogram is generated to count the number of pairs at different distances
3. The histogram is normalized in the end

To spin an image, there are 3 steps.

1. For each vertex, an object centered coordinate (α, β) is computed based on the distance α along the axis and the distance β from the axis
2. A 2D accumulator array is created, and is incremented for each vertex that can be supported by the spin image
3. The final histogram is normalized

10.1.2 Generalized Cylinders (GC)

A Generalized Cylinder (GC) has a planar cross section, a space curve spine, and a sweep rule (constant, linear contraction, etc.).
In other words, a GC has a spine that defines the axis of the cylinder and has a cross section that is swept along that axis by a sweep rule.

10.2 Volumetric Representations

10.2.1 Spatial Occupancy Array

Voxels are the simplest representation, but are very wasteful. They can use sparse arrays, and one of the dimensions can be run-length coded.

10.2.2 Octree

Octrees are the 3D version of quadtrees, where each node has exactly 8 children. When given a $2^n \cdot 2^n \cdot 2^n$ array of voxels, the dimension of the array is 2^n. A node at level i for an octree that has an array with dimension of 2^n corresponds to an element which contains $2^i \cdot 2^i \cdot 2^i$ voxels.

"Void" = "white", "Full" = "black", "Mixed" = "gray", if we were to match the values in a quadtree to those in an octree.
Union, intersection, and complement of octrees can be solved recursively.

10.2.3 Constructive Solid Geometry

Solids are represented as compositions of other solids via set operations. Primitive solids are entities such as arbitrarily scaled rectangular blocks, cylinders, cones, and spheres of arbitrary radius.
A CSG representation involves primitive solids and set operators for combination and motion.
$< CSGRep >::=< primitivesolid > |MOVE < CSGRep > BY < MotionParams > | < CSGRep >< CombineOp >< CSGRep >$
Regularity is an important property of any set of points that models a solid. In a space, a set X is regular if X = kiX, where k and i are the closure and interior operators. This means that a regular set has no isolated or dangling boundary points.
Regularized operators are $X < OP > *Y = r(X < OP > Y)$
If primitives are unbounded, then it is difficult to check for the boundedness of an object. However, if they are bounded, then any CSG representation is a valid volume representation.

10.2.4 Shape Histogram

1. Build the volumetric representation by converting the surface into a set of voxels
2. The space is transformed to a spherical coordinate system around the center of mass of the model
3. A 3D spherical histogram is constructed for accumulating voxels in the volume representation
4. The histogram is normalized

10.3 Viewer-Centered Representation

Viewer-Centered Representation models 3D objects through a set of views ($2\frac{1}{2}$ sketch - depth map, etc.). It is a bottom-up approach, but each 3D object must be represented by many views.

10.4 Object-Centered Representation

Object-Centered Representation involves expressing object geometry with respect to a reference frame, such as GCs, Octrees, Shape Histograms, etc. It is a top-down approach, and it is often better for invariance (translation, rotation, and scale).