Visual Computing Tutuorial Sessions
Summary

September 29, 2020

Chapter 1

Tutorial W2

Segmentation: The goal is to classify the pixels of the image and cluster them into similar arrays e.g if pixels are part of the same object we would like to classify them in the same cluster in the same class.

Semantic Segmentation: Every pixel of interest is assigned a semantic class e.g car,person road etc

Complete Segmentation: If Every pixel of the image is assigned a class.

Binary Segmentation: Each pixel can belong to one of two classes e.g forground/background

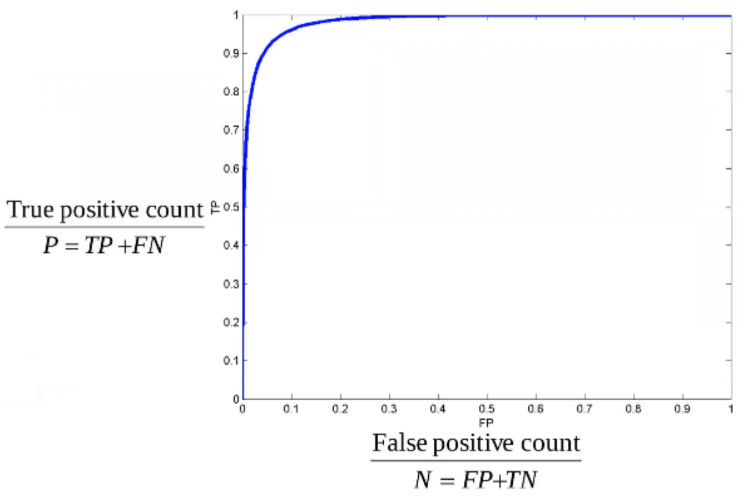
1.1 Threshholding:

The idea is that we set a pixel density threshold. All pixels with a grayscale above this threshold will be in the forground, and all pixels below will be in the background.

ROC (Reciever Operating Characteristic) curve Given a picture there are pixels which belong in the forground and pixels which belong to the background. We apply an algorithm on the picture to distinguish between these forground and background pictures. Positive is for forground and Negative for background, hence we get 4 possibilities for every pixel:

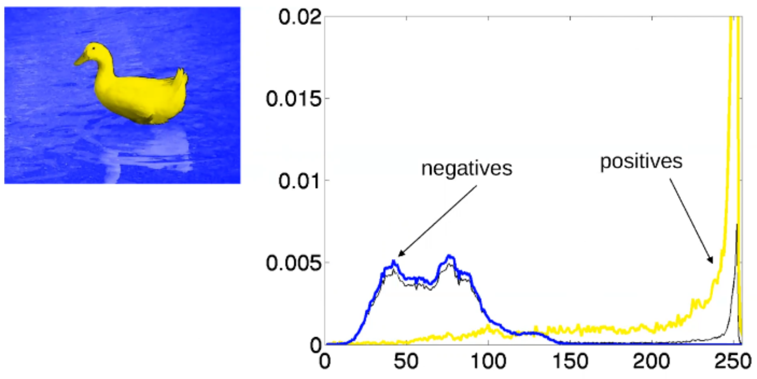
- **True Positive (TP)**: The algorithm labeled the pixel as positive and the pixel belongs in the forground
- **False Positive (FP)**: The algorithm falsely labeled the pixel as positive, when the pixel actually belongs to the background
- **True Negative (TN)**: The algorithm labeled the pixel as negative and the pixel belongs to the background
- **False Negative(FN)**: The algorithm labeled the pixel as negative, when the pixel actually belongs to the forground

We count the occurence of each of the 4 possibilities and create the ROC curve:



The ROC curve always passes through (0,0) and (1,1) and ideally is as close to a 90 degree angle as possible.

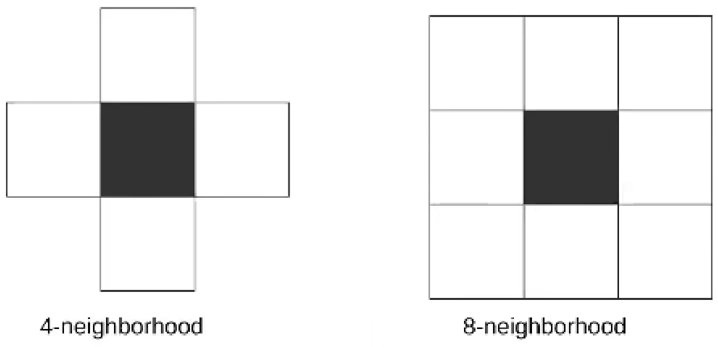
Graylevel Histograms In Graylevel each pixel is given a value between 0 and 255. In an image we can count the occurence of each value and compute the occording histogram where the x-axis is the grayscale value and the y-axis is the ratio of pixels with grayscale intensity x to the total amount of pixels in the image. The histogram will contain peaks and valleys depending on the number of light and dark pixels. This will allow us to visualize which value to take for our Threshold.



The limitation to the thresholding approach is that the forground and background pixels can be too entagled in which case there is no good threshold.

1.2 Region Growing

The idea is that in our image we start at a seed point (a pixel chosen by the user). The algorithm accepts this pixel as the first pixel belonging to the forground. The algorithm then expands by checking if the neighbours fulfill certain criterias e.g if the pixel value is close enough to the previous pixel. We define two neighbourhoods:



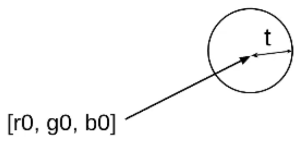
In contrast to thresholding, Region growing gives us a more precise control in which pixels we want in the forground.

1.3 Background Subtraction

The goal is to isolate the forground of a given image.

Approach 1: We represent the background with a single color value i.e the reference value. We then compare the distance of each pixel in the image to this chosen reference value, if this distance is lower than a threshold t then this pixel is assigned to the background otherwise forground. In contrast to Thresholding, this approach is more local i.e we assign pixels to the background if they are in a certain range from our reference. The equation is given by:

$$||[r, g, b] - [r0, g0, b0]|| < t \ (L_2 \text{ distance})$$



This method works if you have a homogeneous background.
Approach 2: This approach also uses a variation of thresholding, but instead of using the L_2 distance we use the **Mahalanobis distance**. Instead of only using a single background color as a reference we take multiple reference points. This lets us approximate the average value of the background. The Mahalanobis distance is given by:

$$(x - \mu)^T \sum^{-1} (x - \mu) > t$$

- μ := mean value across all pixels in the background (3x1 Column Vector)
- $\sum$:= Covariance Matrix acts as a weighting of the different rgb values (3x3 Matrix)
- x:= Random pixel in the image (3x1 Column Vector)

- t:= threshold

Approach 3: Pixel-wise Color Model The idea is that we no longer share mean and covariance between all pixels but instead define a mean and covariance for each individual pixel. For this to work we need a video of just the background and a second video with the object of interest in it. In the first video we can calculate the Mean and covariance using the values of that pixel from each frame i.e over time. We can then apply the Mahalanobis distance to chekc if the pixel is forground or background.

Chapter 2

Tutorial W3

2.1 Convolutions

Convolutions are an operator which map an image and a kernel to an images.

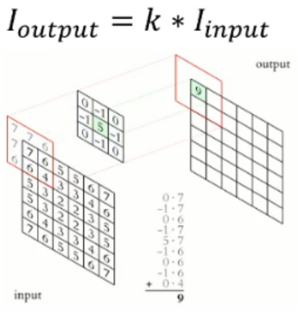
$$I_{out} = k * I_{in}$$

where * represents the convolution. Images are represented as matrices containing all the image samples. The Kernel is also a matrix containing weights for the convolution operator and is usually alot smaller than the image. Convolutions have the following properties:

- **Local:** $I_{out}[i,j]$ depends only on neighbours of $I_{in}[i,j]$
- **Linear:** $k * (\alpha l_1 + \beta l_2) = \alpha(k * l_1) + \beta(k * l_2)$
- **Associative:** $(k_1 * (k_2"l)) = ((k_1"k_2)"l)$ The right hand side is preferred because it has a lower computational cost.
- **Shift Invariant:** $shift(k * l) = k * shift$

A General form of the convolution respecting all of the above properties is given by:

$$I'(x,y) = \sum_{j=-k}^{k} \sum_{i=-k}^{k} K(i,j) I(x-i, y-j)$$



2.2 Edge Detection

<u>**Edges:**</u> Edges in images are areas with strong intensity contrasts.
In order to find an edge we plot the intensity of a line in the image. We take the derivative and see that the biggest change is the maximum magnitude or the point where the 2nd derivative is 0.

<u>**Gradient Method**</u> An image is a 2 dimensional domain. Hence our derivative is going to be represented by a **Gradient Vector**

$$g(x,y) = \begin{pmatrix} g_x(x,y) \\ g_y(x,y) \end{pmatrix} = \begin{pmatrix} (k_x * f)(x,y) \\ (k_y * f)(x,y) \end{pmatrix}$$

With the gradient Vector we can calculate the **Gradient Magnitude**, which indicates how large the gradient is

$$|g| = (g_x^2 + g_y^2)^{\frac{1}{2}}$$

and the **Direction**

$$\theta = tan^{-1}(\frac{g_y}{g_x})$$

For edge detection it is not only important to find the max Gradient Magnitude but also be able to detect which peaks are just noise as not every colorchange is necessarily an edge. As Images are not continuous but a set of samples i.e pixels we cannot calculate the derivative in the usual sense. We can create a Discrete approximation for the derivative:

$$\frac{\delta f}{\delta x]} \approx \frac{f(x+\Delta x, y) - f(x,y)}{\Delta x}$$

Δ replaces the ϵ in the usual continous derivatives and can be a discrete value. The discrete approximation is linear and hence a convolution. e.g if $\Delta x = 1$ we get the following convolution:

$$\frac{\delta I}{\delta x} = [-11] * I$$

For the convolution to be well-defined we must assign the center of the kernel which would be $f(x, y)$ i.e -1 in the example. In general it is better to use larger kernels as it will pick up less noise because it is taking the difference of more distant pixels. Examples of Kernels which approximate the 2D derivative of an image are: **Sobel Kernel:**

$$k_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad k_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Prewitt kernel

$$k_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \qquad k_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

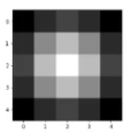
Gradient Thresholding Like Thresholding but instead of grayscale value we use the gradient value.

Canny Edge Detection Combination of noise reduction and edge enhancement. Compared to Gradient Thresholding we focus on having more continuous and thin edges. Canny Edge Detection works in three steps:

1. **Apply derivative of Gaussian filter**. The Gaussian filter suppresses high frequency noise. For efficiency we convolve the derivative and the filter first and then convolve the result with the image giving us the gradient image.

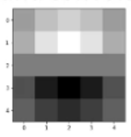
- Need smoothing to reduce noise prior to taking derivative

0.0121	0.0261	0.0337	0.0261	0.0121
0.0261	0.0561	0.0724	0.0561	0.0261
0.0337	0.0724	0.0935	0.0724	0.0337
0.0261	0.0561	0.0724	0.0561	0.0261
0.0121	0.0261	0.0337	0.0261	0.0121



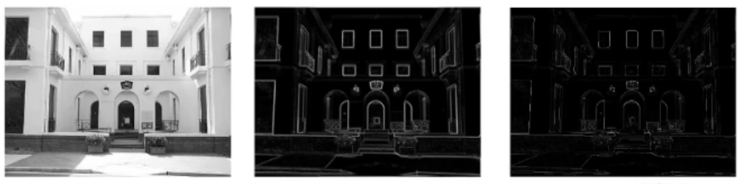
- We can use derivative of Gaussian filters
  - because differentiation is convolution, and convolution is associative:

$$D * (G * I) = (D * G) * I$$



2. **Non-maximum suppression**. Our goal is to eliminate the responses which are not on the peak of the gradient maximum. This allows us to avoid thick regions and get the edges down to single pixel width

- The edge direction angle is rounded to one of four angles representing vertical, horizontal and the two diagonals.
- Select the single maximum point across the width of an edge.
  - Maximum: The gradient magnitudes of the two neighbors in edge normal direction are smaller.



The right image is after Non-maximum suppression. Non-maximum suppression will not delete edges.

3. **Hysteresis** the idea is that real object usualy define continuous edges where as noise is usually disrupted instead. Hence we want to distinguish between both and also preserve continuous edges. In practice we define two thresholds $T_{low} < T_{high}$ and given a pixel from the non-maximum supression image with gradient value G we classify the three possible outcomes:
   - if $G < T_{low}$ then its definitely not an edge
   - if $G > T_{high}$ then its definitely a strong edge
   - if $T_{low} < G < T_{high}$ then its a weak edge if and only if it is connected to any strong edge through other weak edges