AASD 4004
Machine Learning - II

Applied AI Solutions Developer Program

Module 11 Smoothing Thresholding

Vejey Gandyer

Agenda

- Smoothing
- Averaging Blur
- Gaussian Blur
- Median Blur
- Bilateral Blur
- Simple Thresholding
- Adaptive Thresholding
- OTSU Thresholding
- Riddler-Calvard Thresholding

Image Smoothing

What is it?

Image Blurring

It’s what happens when your camera takes a picture **out of focus**

Sharper regions in the image lose their detail, normally as a disc/circular shape

Practically, each pixel in the image is **mixed** in with its surrounding pixel intensities

This “mixture” of pixels in a neighborhood becomes a blurred pixel

While this effect is usually unwanted in our photographs, it’s actually quite helpful when performing image processing tasks

Types of Smoothing

Types

- Averaging Blur
- Gaussian Blur
- Median Blur
- Bilateral Blur

Averaging Blur

Define a **k × k sliding window** on top of our image, where k is always an odd number

- This window is going to slide from left-to-right and from top-to-bottom
- The pixel at the center of this matrix (we have to use an odd number, otherwise there would not be a true "center") is then set to be the **AVERAGE** of all other pixels surrounding it

blur( )

Gaussian Blur

Gaussian blurring is similar to average blurring, but instead of using a simple mean, we are now using a **weighted mean**

Weighted Mean: Neighborhood pixels that are closer to the central pixel contribute more "**weight**" to the average

Result: Our image is less blurred, but **more naturally blurred**, than using the average blur method

GaussianBlur( )

Median Blur

Median blurring is similar to average blurring, but instead of using a simple mean, we are now replacing the central pixel with the **median** of the neighborhood

Median blurring is more effective at removing **salt-and-pepper** style noise from an image because each central pixel is always replaced with a pixel intensity that exists in the image

MedianBlur( )

Bilateral Blur

Bilateral blurring **preserves edges** with **two Gaussian distributions**

The first Gaussian function only considers **spatial neighbors** pixels that appear close together in the (x, y) coordinate space of the image

The second Gaussian then models the **pixel intensity** of the neighborhood pixels with similar intensity are included in the actual computation of the blur

BilateralFilter( )

Image Thresholding

What is it?

Image Thresholding

Thresholding is the **binarization** of an image

A simple thresholding example would be as follows:

- Selecting a pixel value p
- Setting all pixel intensities less than p to zero
- Setting all pixel values greater than p to 255

Types of Thresholding

Types

- Simple
- Adaptive
- Otsu and Riddler-Calvard

Simple Thresholding

Applying simple thresholding methods requires human intervention - **threshold( )**

- Specify a threshold value T
- All pixel intensities below T are set to 0
- All pixel intensities greater than T are set to 255

Inverse Binarization

- Setting all pixels below T to 255
- Setting all pixel intensities greater than T to 0

Simple Thresholding

Drawbacks

- Manual supply of Threshold value T
- Requires lots of experiments with different T

Solution: **Adaptive thresholding**

Adaptive Thresholding

Consider **small neighbors** of pixels and then find an **optimal threshold value** T for each neighbor

Uses

- High range of pixel intensities in the image
- Optimal value of T may change for different parts of the image

OTSU Thresholding

Otsu’s method assumes there are **two peaks** in the grayscale histogram of the image It then tries to find an optimal value to **separate** these two peaks – thus our value of T

OTSU Thresholding

- Find the optimal threshold for the image
  - Use mahotas.thresholding.otsu
- Apply threshold
  - Copy the image
  - Set to white pixel if pixel intensity greater than T
  - Set to black pixel if pixel intensity less than 255
  - Invert the thresholding
    - Use **cv2.bitwise_not**

Riddler-Calvard Thresholding

- Find the optimal threshold for the image
  - Use mahotas.thresholding.rc
- Apply threshold
  - Copy the image
  - Set to white pixel if pixel intensity greater than T
  - Set to black pixel if pixel intensity less than 255
  - Invert the thresholding
    - Use cv2.bitwise_not

Further Reading

Digital Image Processing 4th edition

Rafael Gonzalez & Richard Woods