Adaptive Median Filter

(Noise removal in Image processing)

# Standard Median filter

In image processing median filter is usually necessary to perform a high degree of noise reduction in an image before performing higher-level processing steps. The **median filter** is a non-linear digital filter technique, often used to remove [speckle](http://en.wikipedia.org/wiki/Speckle_noise)  ([salt and pepper](http://en.wikipedia.org/wiki/Salt_and_pepper_noise)) [noise](http://en.wikipedia.org/wiki/Signal_noise) from images. Median filtering is a common step in image processing.

The main idea of the median filter is to sort the pixel values in a neighborhood region with certain window size and then chose the median of these values and places it in the center of the window in a new image, see figure 1. This process is repeated for all pixels in the original image.

Original Image

Old Value

Sort and chose median

New Value

……

New Image

……….

Window

Figure 1: Main idea of median filter

As the window size increased, the effect of the filter is increased, as shown in figure 2.

## Implementation steps of Standard Median filter on Image

The idea is to calculate the median of neighboring pixels' values. This can be done by repeating the following steps for **each pixel** in the image:

1. Store the values of the neighboring pixels in an array. The array is called the window, and it should be odd sized.
2. Sort the values in the window in ascending order.
3. Pick the **median** from the window as the new pixel value and place it in the center of the window in the new image, see figure 1.

## Notes

* We work on gray-level images. So, each pixel has a value ranged from 0 to 255. Where 0 is the black pixel and 255 is the white pixel.
* The median is the middle value in the window array after sorting.



Image with salt & pepper noise Median filter with window 3×3



Median filter with window 5×5 Median filter with window 7×7

Figure 2: Effect of the standard median filter with different window size

# Adaptive Median Filter

The standard median filter has the following drawbacks:

1. It fails to remove salt and pepper noise with large percentage (greater than 20%) without causing distortion in the original image.
2. It usually has a side-effect on the original image especially when it’s applied with large mask size, see figure 2 with window 7×7.

Adaptive median filter is designed to handle these drawbacks by:

1. Seeking a median value that’s not either salt or pepper noise by increasing the window size until reaching such median.
2. Replace the noise pixels only. (i.e. if the pixel is not a salt or a pepper, then leave it).

This is clear in figure 3 and figure 4. Compare the effect of both filters in each case. Note that both can remove the noise, but adaptive filter don’t cause large distortion on the original image as the standard filter do.

|  |  |
| --- | --- |
| S_P_0.2.bmp  Original image corrupted with salt and pepper noise with percentage ≈ 30% | S_P_0.2 Med7.bmp  Standard median filter with window 7×7 |
| AdaptiveMedian7.bmp  Adaptive median with max window 7×7 |

Figure 3: Effect of adaptive vs. standard median filter on small percentage of salt and pepper noise

|  |  |
| --- | --- |
| SP 0.25.bmp  Original image corrupted with salt and pepper noise with percentage ≈ 50% | Median7 -.bmp  Standard median filter with window 7×7 |
| AdaptiveMedian7 -.bmp  Adaptive median with max window 7×7 |

Figure 4 Effect of adaptive vs. standard median filter on large percentage of salt and pepper noise

## Implementation steps of Adaptive Median filter on Image

Adaptive median filter has variable window size, and the procedure of updating the pixel value is as follows:

For each pixel in the image:

try window sizes ranging from 3×3 to WS × WS, where WS is the maximum window size entered by the user, as follows:

### Step 0: Start by window size 3×3

### Step 1: Chose a non-noise median value

Sort the current window, and denote the following:

1. Zxy is the gray value of the current pixel value at location (x, y)
2. Zmax is the maximum gray value in the window.
3. Zmin is the minimum gray value in the window.
4. Zmed is the median gray value in the window.

A1 = Zmed – Zmin

A2 = Zmax – Zmed

If A1 > 0 and A2 > 0 then we found a non-noise median

Go to Step 2

Else

Increase window size by 2

If new window size ≤ WS then

Repeat Step 1 again

Else

NewPixelVal = Zmed

### Step 2: Replace the center with the median value, or leave it

B1 = Zxy – Zmin

B2 = Zmax – Zxy

If B1 > 0 and B2 > 0 then

NewPixelVal = Zxy //leave the center pixel as it is

Else

NewPixelVal = Zmed //replace the center pixel with the median value

### Step 3: repeat the process for the next pixel starting from step 0 again

The steps above summarize what’s done through adaptive median filter implementation. The meaning of these steps is as follows:

### Step 1: Search for a true median

IF the current window has a true median (i.e. Zmed is different from Zmin and Zmax) THEN

//Execute Step 2

ELSE

IF the current window size is not the maximum

Increase it and repeat Step 1

ELSE

Let the output pixel be Zmed and move to the next pixel.

ENDIF

EndIF

### Step 2: if we have a true median

IF (Zxy is different from Zmin and Zmax) (i.e. not noise)

THEN

Let the output pixel be Zxy (i.e. not changed) and move to the next pixel.

ELSE

Let the output pixel be Zmed and move to the next pixel.

# Project Requirements

It is required to apply the adaptive median filter with three different algorithms:

1. Quick Sort
2. Counting Sort
3. Selecting Kth smallest element (median) in the array without sorting it (Textbook sec. 9.2). where K = index of median = (WS × WS) / 2
4. Selecting each of the min and max value **separately** without sorting.
5. Selecting both of them **efficiently**, without sorting, as described in Textbook sec.9.1.
6. Display two graphs to show the execution time **against different window sizes** (3, 5, 7,… Wmax), where Wmax is user input.
7. One graph for comparing the first three methods (quick, counting, selecting kth element for median + min & max separately)
8. Another graph for comparing the two sub-methods of selecting min & max, under selecting Kth element method.

## Notes:

* To calculate time of certain peace of code:

1. Get the system time before the code
2. Get the system time after the code
3. Subtract both of them to get the time of your code

To get system time in milliseconds, you can use System.Environment.TickCount

* To draw the graph, you can use the Z-graph library attached with the project materials.

## Given:

* TEMPLATE C# Code to

1. Open image & load it in 2D array stored in a global variable of type byte[,]called ImageMatrix, using the following function inside ImageOperations class

byte [,] OpenImage(string ImagePath)

1. Get width and height of the image matrix

int GetHeight(byte[,] ImageMatrix)

int GetWidth(byte[,] ImageMatrix)

1. Display an image on a given PictureBox control using the following function inside ImageOperations class

void DisplayImage(byte[,] ImageMatrix, PictureBox PicBox)

* Zed-Graph library to use it for drawing the graph, with sample code showing how to use it.

## Input:

1. Noisy image
2. Max window size of the adaptive median filter (Ws)
3. Max window size for the graph (Wmax)

# Deliverables

## Implementation

1. Adaptive median filter using 3 methods
   1. Counting sort
   2. Quick sort
   3. Select Kth smallest element to select median
      1. Select min & max separately
      2. Select both of them efficiently (sec. 9.1)
2. Display two graphs (one for the first 3 methods, other for 2 methods under selecting Kth element) to show the execution time **against different window sizes** (3, 5, 7,…).

## Document contains:

1. Determine which method is better **based on your results**? And **explain why**?

# Grades

|  |  |
| --- | --- |
| **Deliverable** | **Grade** |
| Implementation of adaptive median filter using 3 methods | 35% |
| Implementation of 2 methods for selecting min & max | 15% |
| Plotting two graphs for different window size | 30% |
| Document | 20% |

# BONUS TASK:

There are some heuristics that can achieve implementing the median filter in image processing in **MUCH** better performance than any sorting/selection algorithm. Try to find/search the idea and implement it.