中值滤波的快速计算方法

姓名：伍斌

学号：17343124

中值滤波原方法

用一个中值滤波器窗口依次遍历整个图像，在每个位置都计算得到一个中值，然后将这些值拼在一起就得到了中值滤波后的图像。中值滤波可以很好地去除椒盐噪声。

代码（MATLAB）

```
% imgOrg = rgb2gray(imread("peppers.png"));
imgOrg = imread("picture.jpg");
imgOrg = imnoise(imgOrg, 'salt & pepper'); % Adding salt-pepper noise
subplot(1,3,1);
imshow(imgOrg);
title("Originall img");

% median filter
imgMedian = medianfilter(imgOrg, 3); % Generally, the size of a filter is an odd
num
subplot(1,3,2);
imshow(imgMedian);
title("Median filter img");

% Using Matlab API
subplot(1,3,3);
[~, ~, c] = size(imgOrg);
newImg = zeros(size(imgOrg));
```

```
for i = 1:c
    newImg(:,:,i) = medfilt2(imgOrg(:,:,i));
end
newImg = uint8(newImg);
imshow(newImg);
title("Medfilt img by API");

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function imgMedian = medianfilter(img,filterSize)
% img: image
% filterSize: if=5, it means 5*5

% Generate new blank image
[h, w, c] = size(img);
padSize = (filterSize-1)/2;
imgMedianT = zeros([h+2*padSize, w+2*padSize, c]);
imgMedianT(1+padSize:padSize+h, 1+padSize:padSize+w, :) = img;
imgMedian = zeros([h, w, c]);

for k = 1:c
    for i = 1:h
        for j = 1:w
            block = imgMedianT(i:i-1+filterSize, j:j-1+filterSize, k);
            imgMedian(i, j, k) = median(block, 'all');
        end
    end
end

imgMedian = uint8(imgMedian);

end
```

实验结果

运行用时：

函数名称	调用次数	总时间 (秒)	自用时间* (秒)	总时间图 (深色条带 = 自用时间)
main	1	9.102	0.010	
main>medianfilter	1	8.752	1.997	
median	786432	6.755	4.045	
median>isAllFlag	1572864	2.710	2.710	
imshow	3	0.144	0.042	
subplot	3	0.129	0.063	

其中中值滤波算法用时：8.752s

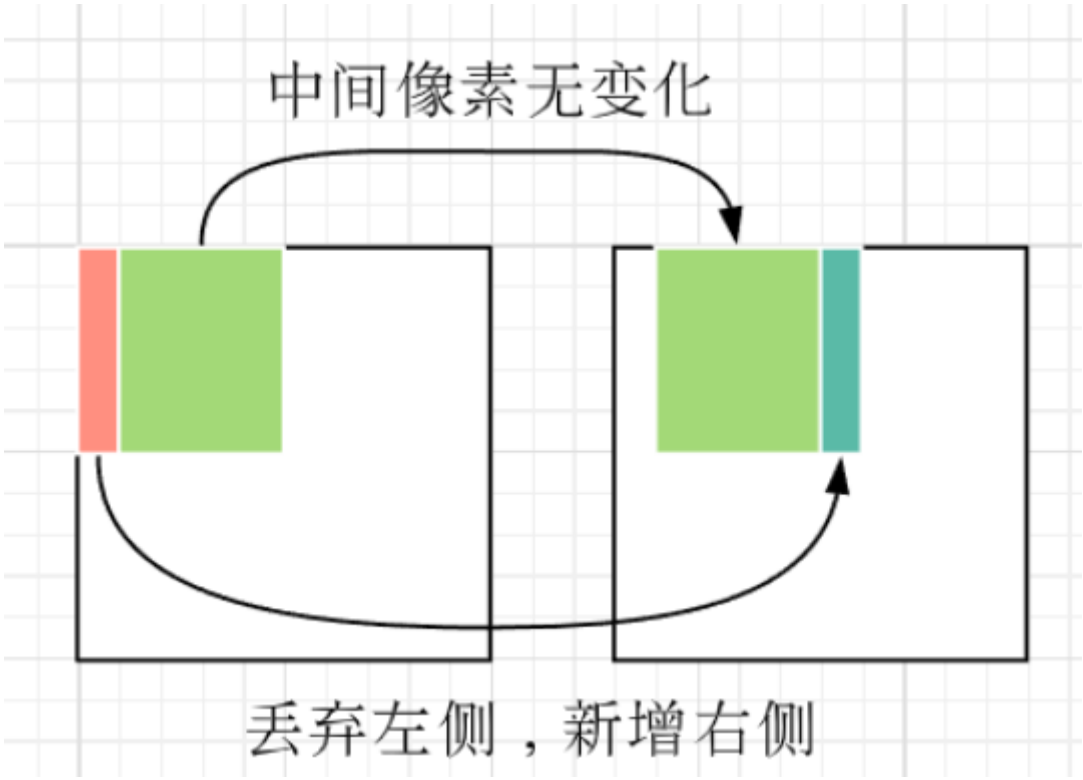
▾ 占用时间最长的行

行号	代码	调用次数	总时间(秒)	% 时间	时间图
41	`imgMedian(i, j, k) = median(block, 'all');`	786432	7.519	85.9%	
40	`block = imgMedianT(i:i-1+filterSize, j:j-1+filter…`	786432	1.191	13.6%	
42	`end`	786432	0.035	0.4%	
34	`imgMedianT(1+padSize:padSize+h, 1+padSize:padSize…`	1	0.004	0.0%	
46	`imgMedian = uint8(imgMedian);`	1	0.001	0.0%	
所有其他行			0.002	0.0%	
总计			8.752	100%	

可见由于每次移动滤波器窗口就要重新求一次中值，而要求中值需要进行排序，所以开销很大。

中值滤波的快速计算方法

仔细分析中值滤波算法发现，每一移动滤波器窗口时，该窗口的变化为：丢掉最左边的一列而新增最右边的一列，对于m*n的中值滤波窗口，有mn-2m个像素没有发生变化，因此不需要重新排序。



新算法

设滤波器窗口大小为x * y = 3 * 3，一个图像的像素分布为：

0	189	116	55
84	152	229	120
105	73	20	255
237	25	188	100

步骤一：

令$t = ceil(xy/2)$

例子中则为：$t = ceil(3 * 3/2) = 5$

t为排序后中位数所处的位置。取整可以避免不必要的浮点数运算。

步骤二：

将窗口移至一个新行的开始，对其内容排序。建立窗口像素的直方图H，确定其中值m，记下亮度≤m的像素数目n_m.

例中为当滤波器移至如图所示黄色区域窗口时：

0	189	116	55
84	152	229	120
105	73	20	255
237	25	188	100

n_m相当于当前中值的位置，例如下图3 * 3滤波器窗口排序好后，105为中值，此时n_m= 5，即此时中值的位置为5 号位置。

0	20	73	84	105	116	152	189	229

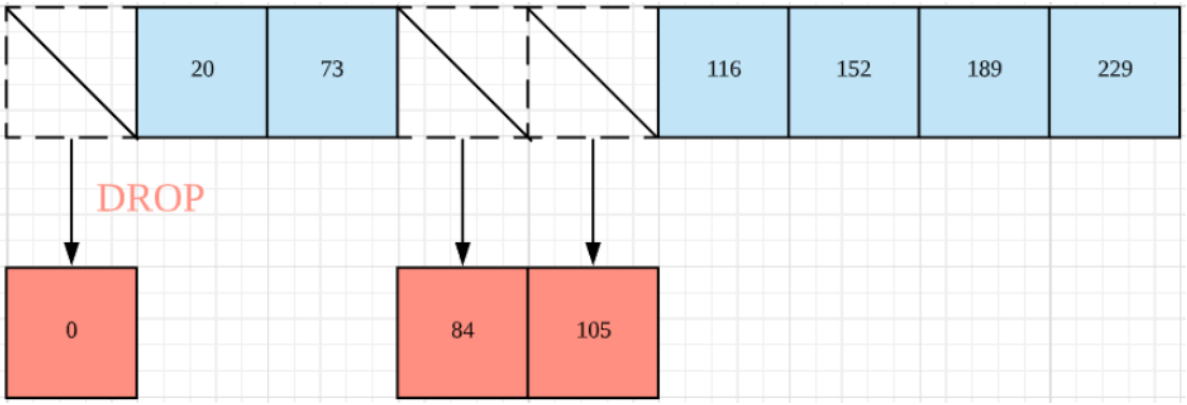
步骤三:

向右移动滤波器，对于移出的左列的每个亮度为p_g的像素 p ，做如下操作：$H[p_g] = H[p_g] - 1$，若 $p_g \leq m$， 则 $n_m = n_m - 1$

例中当滤波器移至如图示黄色区域窗口时：

0	189	116	55
84	152	229	120
105	73	20	255
237	25	188	100

最左侧（如红色部分）一列被移除，最右侧一列被移入。所以执行
$H[0] = H[0] - 1, H[84] = H[84] - 1, H[105] = H[105] - 1$，即：



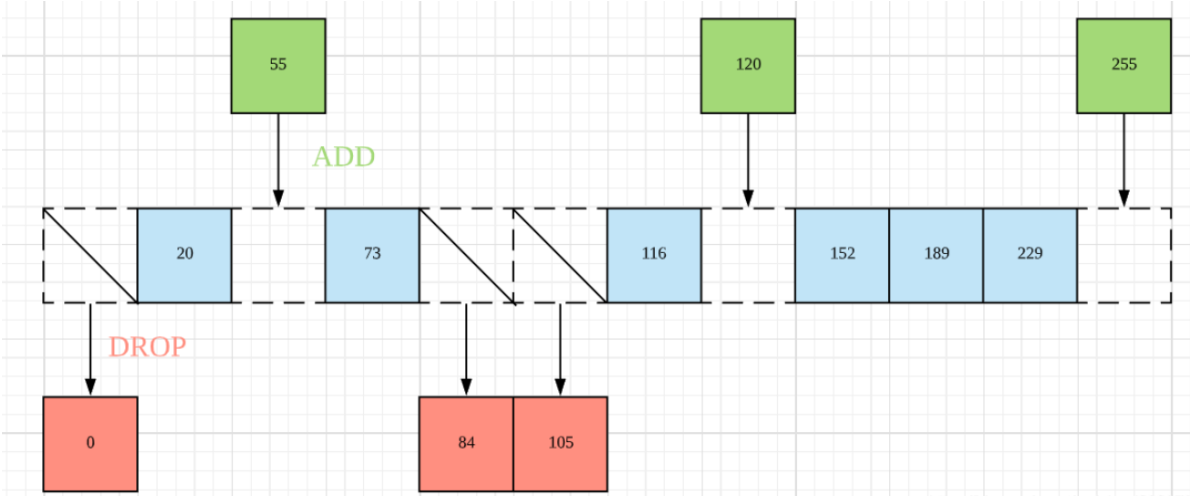
由于$m = 105$，$0 \leq m, 84 \leq m, 105 \leq m$，所以现在的$n_m = 5 - 1 - 1 - 1 = 2$

步骤四：

对于移入的右列的每个亮度为p_g的像素 p，做如下操作：$H[p_g] = H[p_g] + 1$，若$p_g \leq m$，则 $n_m = n_m + 1$

对于例中：

执行$H[55] = H[55] + 1, H[120] = H[120] + 1, H[255] = H[255] + 1$，即：



由于$m = 105, 55 \leq m$，所以现在的$n_m = 2 + 1 = 3$

步骤五：

若$n_m = t$，则跳至步骤八执行：移动滤波器后，没有改变中值，所以无需进行额外操作；

若$n_m < t$，则跳至步骤六执行：移动滤波器后，改变了中值的位置，所以需要进行额外修正操作；

若$n_m > t$，则跳至步骤七执行：移动滤波器后，改变了中值的位置，所以需要进行额外修正操作。

例中 $n_m = 3, t = 5, n_m < t$，所以执行步骤六。

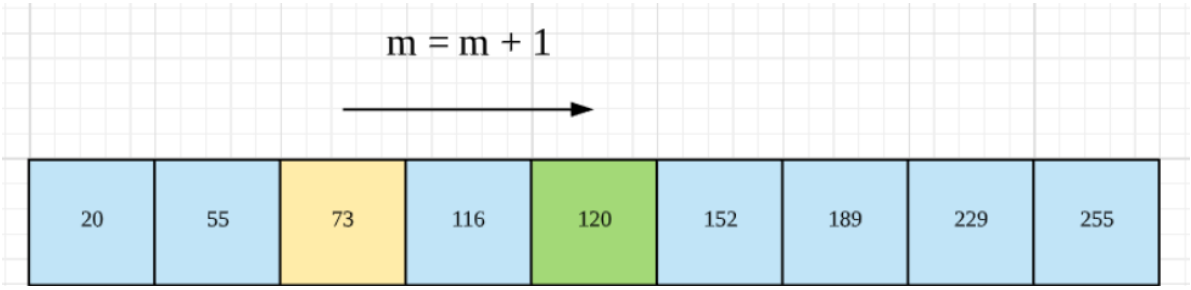
步骤六：

重复执行以下操作：

$m = m + 1$

$n_m = n_m + H[m]$

直到 $n_m \geq t$，跳至步骤八继续执行。



步骤七：

重复执行以下操作：

$n_m = n_m - H[m]$

$m = m - 1$

直到 $n_m \leq t$，跳至步骤八继续执行。

步骤八：

记录*m*为当前窗口的中值。

然后检查滤波窗口的右列是否为图像的右边界：若是，跳至步骤九；反之，跳至步骤三。

步骤九：

检查滤波窗口的底列是否为图像的下边界： 若是，图像遍历完成，算法结束； 反之，跳至步骤二。

中值滤波窗口需要遍历图像的每一行，一行滤波完成后换新行重复操作。

代码（MATLAB）

```
img = imnoise(imread("picture.jpg"),'salt & pepper',0.1);
filterSize = 5; % odd
tic;
newImg1 = effMedian(img, filterSize);
toc;
subplot(1,2,1);
imshow(img);
title("Original Image");
subplot(1,2,2);
imshow(newImg1);
title("Filtered Image");

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function imgMedian = medianfilter(img,filterSize)
% img: image
% filterSize: if=5, it means 5*5

% Generate new blank image
[h, w, c] = size(img);
padSize = (filterSize-1)/2;
imgMedianT = zeros([h+2*padSize, w+2*padSize, c]);
imgMedianT(1+padSize:padSize+h, 1+padSize:padSize+w, :) = img;
imgMedian = zeros([h, w, c]);

for k = 1:c
    for i = 1:h
        for j = 1:w
            block = imgMedianT(i:i-1+filterSize, j:j-1+filterSize, k);
            imgMedian(i, j, k) = median(block, 'all');
        end
    end
end

imgMedian = uint8(imgMedian);

end
```

实验结果

运行用时：

函数名称	调用次数	总时间 (秒)	自用时间* (秒)	总时间图 (深色条带 = 自用时间)
main	1	3.949	0.018	
effMedian	1	3.629	3.616	
subplot	2	0.119	0.055	
imshow	2	0.118	0.031	
newplot	4	0.029	0.006	
imread	1	0.028	0.001	

其中快速中值滤波算法用时：3.629s

▾ 占用时间最长的行

行号	代码	调用次数	总时间(秒)	% 时间	时间图
33	`moveOut = tempImg(h-ch:h+ch,w-1-cw,c);`	784896	0.851	23.4%	
34	`moveIn = tempImg(h-ch:h+ch,w+cw,c);`	784896	0.753	20.8%	
39	`n_m = n_m - length(find(tempBlock <= m)); % Step 3`	784896	0.379	10.4%	
44	`n_m = n_m + length(find(tempBlock <= m)); % Step 4`	784896	0.324	8.9%	
35	`tempBlock = moveOut(:);`	784896	0.151	4.2%	
所有其他行			1.171	32.3%	
总计			3.629	100%	

显然，快速中值滤波算法在实现效果好的同时，实现了更少的开销。