ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ
UNIVERSITY OF PATRAS

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING
MSc in Biomedical Engineering

PROJECT IN MEDICAL IMAGE PROCESSING AND ANALYSIS

STRUCTURE-PRESERVING FILTERING FOR FUSION OF MULTY-PARAMETRIC MRI BRAIN IMAGES

STUDENT: Maria Revythi

Data fusion is the process of integrating information across different sources, such as imaging modalities. For example, multi-modal image fusion allows to combine structural and functional information in the same spatial domain, thereby facilitating treatment design. During fusion the most important information is retained from each modality. Structure-preserving image filtering allows to preserve small-scale details during image fusion.

The main aim of this project is to extract salient regions from the different MRI sequences with a low computational complicity. Secondly, the iterative joint filtering approach is implemented in order to perform structure-aware MR image fusion.

1. Salient structure extraction

The salient structure extraction is the crucial problem to decide which portion of each source image are selected. This process and also the code are described below:

Step 1: Image import and its display

```
I1 = niftiread('C:\Users\maria\Documents\MARIA\BioMed\SECOND_SEMESTER\EIKONA\project2\MICCAI_BraTS2020_Trai
I2 = niftiread('C:\Users\maria\Documents\MARIA\BioMed\SECOND_SEMESTER\EIKONA\project2\MICCAI_BraTS2020_Trai
I1=I1(:,:,75);
I1=squeeze(I1);
figure;imagesc(I1); colormap(gray);
I2=I2(:,:,75);
I2=squeeze(I2);
figure;imagesc(I2); colormap(gray);
```

Step 2: Image normalization with the aim of equation 8 from the paper

```
G1 = mat2gray(I1); %image normalization eq8
G2 = mat2gray(I2);
```

Step 3: Calculation of the adaptive local wiener filter (equation 9)
This filter is used in order to reduce the influence of noise and the irrelevant information (functions boxfilter and smoothing are described in the appendix)

(radius of the filtering window is set to 3 and the regularization parameter is set to λ=0.01 from the paper.)

```
%addaptive modified wiener filter
 r = 3;  % radius of the filtering window
 lambda = 0.01; % regulation parameter
[hei, wid] = size(G1); %definition of hei and wid
N = boxfilter(ones(hei, wid), r); %   a sliding window centered at the pixel p
% (it is defined as Ω in the paper)
Ga = smoothing(G1, r, lambda,N); %eq9 adaptive local wiener filter
Gb = smoothing(G2, r, lambda,N); %eq9
```

Step 4: Calculation of the gradient magnitude by absolute values (equation 13)

```
h = [1 -1];
MA = abs(conv2(Ga,h,'same')) + ...
     abs(conv2(Ga,h','same')); %eq 13 magnitude of Ga
MB = abs(conv2(Gb,h,'same')) + ...
     abs(conv2(Gb,h','same')); %eq 13 magnitude of Gb
```

Step 5: Decision map (equation 14)

Salient patches produce larger D metrics, and flat patches produce smaller D metrics. So a larger positive value in the decision map D implies that a pixel in GA sharper changed while a smaller negative implies that a pixel in GB sharper changed.

```
D = MA - MB; %eq14 decision map
```

Step 6: Calculation of salient structure (equation 15&16)

A mean filter (equation 15) is used to ensure a pixel with a larger gradient magnitude influences the decision of its neighborhood more and offers a means of spreading the sharp changed pixel to its neighborhood.

The salient structure is a binary matrix, where number 1 implies that a pixel in GA sharp changed while number 0 implies that a pixel in GB sharp changed.

```
IA = boxfilter(D,r) ./ N>0; %eq 15&16 calculation of salient structure
```

2. Iterative joint filter

In general, IJF can be used to recover small-scale details in the neighborhood of large-scale structures, which renders the final fusion image clearer and the patches along the structures look more natural. They are needed 5 parameters for the IJF, which are salient structure (IA), normalized image(GA), a space domain scaling parameter, is a range domain scaling parameter and the iterative time.

The main steps, which describes the structure-aware image fusion algorithm are the following:

Input: Two image modalities

Output: Fusion result F

Step 1: Normalization

Step 2: Smoothing

Step 3: Calculation of gradient magnitude

Step 4: Calculation of salient structures

The matlab code for the above steps is described before.

Step 5: for t ∈ [1, T] do
Filter IA to obtain IT by (19).
end for

```
for t = 1:3
    IA = double(IA > 0.5); %eq 19 step(It - 0.5)
    IA = RF(IA, 10, 0.2, 1, GA); %10 is is a space domain scaling parameter
    %0.2 is a range domain scaling parameter
    %1 is the number of iterations
end
```

This method uses equation 19 instead of 17 in order to transfer the details fast.

Step 6: Obtain the output F by equation 18.

```
F = IA.*GA + (1-IA).*GB; %eq18 final fusion result
F = uint8(255*F);
F=squeeze(F);
```

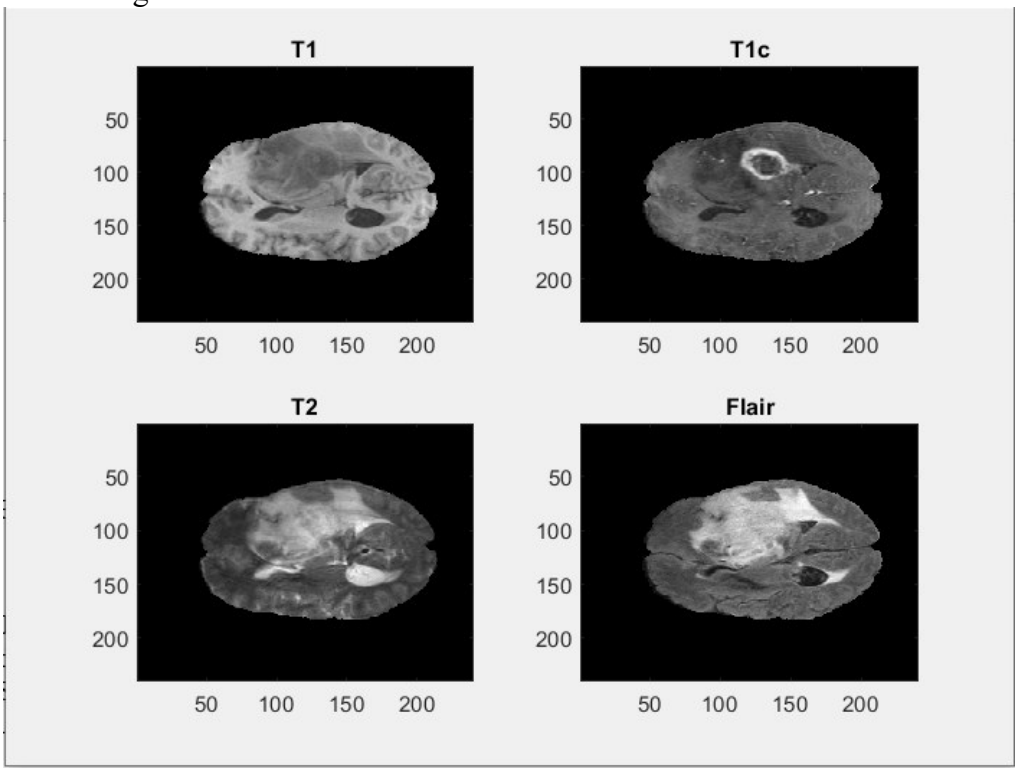
F is a linear combination of intensities in the two image modalities.

3. Extension to several images

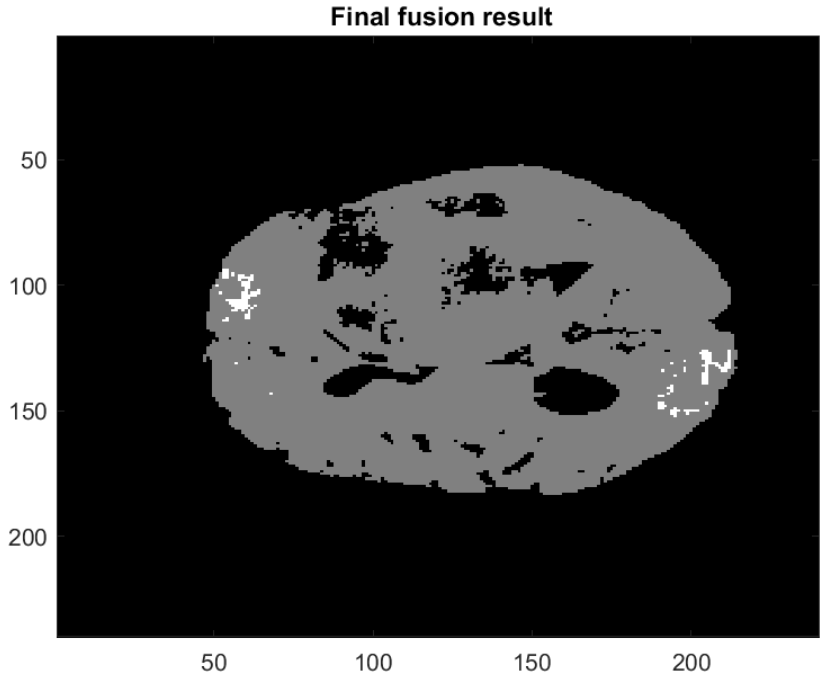
They are 4 image modalities in my dataset. In order to extend the algorithm to these datasets, I first perform the algorithm on the first two source images in the set. Secondly, the algorithm is performed between the fusion image and one of remaining images and so forth.

4. Images

The 4 image modalities are described below



The final fusion result is the following


Final fusion result
50
100
150
200
50
100
150
200

APPENDIX

Some functions are used in the main code, and they are explained here.

I. Boxfilter

```
function imDst = boxfilter(imSrc, r)

%   BOXFILTER   O(1) time box filtering using cumulative sum
%
%   - Definition imDst(x, y)=sum(sum(imSrc(x-r:x+r,y-r:y+r)));
%   - Running time independent of r;
%   - Equivalent to the function: colfilt(imSrc, [2*r+1, 2*r+1], 'sliding', @sum);
%   - subdivides imSrc into regions of size 2*r+1 block-by-2*r+1 block blocks to save memory.
%Column-wise neighborhood operations
%   - But much faster.
%Box Filter is a low-pass filter that smooths the image by making each output pixel the average of the surround
[hei, wid] = size(imSrc);
imDst = zeros(size(imSrc)); %definition of matrix imDst

% Y axis
imCum = cumsum(imSrc, 1); %calculation of cumulative sum over  Y axis

imDst(1:r+1, :) = imCum(1+r:2*r+1, :);%difference over Y axis
imDst(r+2:hei-r, :) = imCum(2*r+2:hei, :) - imCum(1:hei-2*r-1, :); %constant part
imDst(hei-r+1:hei, :) = repmat(imCum(hei, :), [r, 1]) - imCum(hei-2*r:hei-r-1, :);

% X axis
imCum = cumsum(imDst, 2); %calculation of cumulative sum over X axis

imDst(:, 1:r+1) = imCum(:, 1+r:2*r+1); %difference over X axis
imDst(:, r+2:wid-r) = imCum(:, 2*r+2:wid) - imCum(:, 1:wid-2*r-1);
imDst(:, wid-r+1:wid) = repmat(imCum(:, wid), [1, r]) - imCum(:, wid-2*r:wid-r-1);
end
```

This function calculates the sliding window centered at a specific pixel. The result is calculated using a matrix(imSrc) and the radius of the filtering window (r). This function creates a Temporary Matrix for Sliding Neighborhood and it is used as a mean filter.

II. Smoothing

```
function q = smoothing(I, r,eps,N) %adaptive local wiener filter
    mean_I = boxfilter(I, r) ./ N; %the mean of the image I in the window
    mean_II = boxfilter(I.*I, r) ./ N; %the mean of the square of I in the window
    var_I = mean_II - mean_I .* mean_I; %calculation of the variance of image I in the window
    a = var_I ./ (var_I + eps); %calculation of constant k in the paper eq12
    q = mean_I+ a.*(I-mean_I); %eq10
end
```

This function calculates the adaptive local wiener filter. This filter is calculated from the equation 10. This function needs a normalized image(I), the radius of the filtering window(r), the regulation parameter(eps) and a sliding window centered at pixel p (N).

If we tune the regularization parameter eps in a range of the intensity scale of noise, the noisy pixels in flat patches can be smoothed by (10). i.e., if I changes a lot within the window, we have var_I > eps, so k=1, then I = q. If I is almost constant in a window located at a flat patch, we have var_I < < *eps*, so k=0, then $q = mean\_I$. In this way, the major structures (e.g. edges and corners) can be preserved and noise in flat regions can be smoothed.

III. RF

```
function F = RF(img, sigma_s, sigma_r, num_iterations, joint_image)
    I = double(img);
    J = double(joint_image);
    [h, w, num_joint_channels] = size(J);
    dIcdx = diff(J, 1, 2); %calculation of Differences along x-axis
    dIcdy = diff(J, 1, 1); %calculation of Differences along y-axis

    dIdx = zeros(h,w);
    dIdy = zeros(h,w);
    % Calculation l1-norm distance of neighbor pixels.
    for c = 1:num_joint_channels
        dIdx(:,2:end) = dIdx(:,2:end) + abs( dIcdx(:,:,c) );
        dIdy(2:end,:) = dIdy(2:end,:) + abs( dIcdy(:,:,c) );
    end
     % Calculations of  the derivatives of the horizontal and vertical domain transforms.
    dHdx = (1 + sigma_s/sigma_r * dIdx); %eq 11 in the second paper
    dVdy = (1 + sigma_s/sigma_r * dIdy); %is the distance between neighbor samples
    % The vertical pass is performed using a transposed image.
    dVdy = dVdy';
```

```
    %% Perform the filtering.
    N = num_iterations;
    F = I;

    sigma_H = sigma_s;

    for i = 0:num_iterations - 1
        % calculation of  s the standard deviation for the kernel used in the i-th iteration
        sigma_H_i = sigma_H * sqrt(3) * 2^(N - (i + 1)) / sqrt(4^N - 1); %eq 14 from the second paper

        F = TransformedDomainRecursiveFilter_Horizontal(F, dHdx, sigma_H_i);
        F = TransformedDomainRecursiveFilter_Horizontal(F', dVdy, sigma_H_i);
        F = F';
    end
end
```

```
    %% Recursive filter.
    function F = TransformedDomainRecursiveFilter_Horizontal(I, D, sigma)


    a = exp(-sqrt(2) / sigma); %calculation of  RF feedback coefficient (appendix of second paper)

    F = I;
    V = a.^D;

    [~, w, num_channels] = size(I);

    % Left -> Right filter.
    for i = 2:w
        for c = 1:num_channels
            F(:,i,c) = F(:,i,c) + V(:,i) .* ( F(:,i - 1,c) - F(:,i,c) ); %eq 21 in the second paper
        end
    end

    % Right -> Left filter.
    for i = w-1:-1:1
        for c = 1:num_channels
            F(:,i,c) = F(:,i,c) + V(:,i+1) .* ( F(:,i + 1,c) - F(:,i,c) ); %eq 21 in the second paper
        end
    end

end
```

In general, this function calculates a recursive edge-preserving filter. More specific this function calculates the joint filter. As input, is needs the one image (in our case the salient structure), the guided image (in our case the normalized image), a space domain scaling parameter(sigma_s), a range domain scaling parameter(sigma_r) and the number of iterations).

Bibliography

Oliveiray, E. S. (s.d.). Domain Transform for Edge-Aware Image and Video Processing.

Wen Lia, Y. X. (s.d.). Structure-aware image fusion.