**Content**

1. Method - 1: STDFusionNet
2. Method – 2: Wavelet (dwt)
3. Method – 3: Wavelet (wavedec2)

**Method – 1: STDFusionNet**

## **Loading and Preprocess Infrared and Visible Image:**

Both infrared and Visible images were loaded and converted to gray scale and then to double for further processing. And plotting the histogram of infrared grayscale image.

**Code:**

% Load Input Images

IR = imread('manWalkIR.jpg');

VIS = imread('manWalkVB.jpg');

figure(1)

imshow(IR); title('Original Infrared Image');

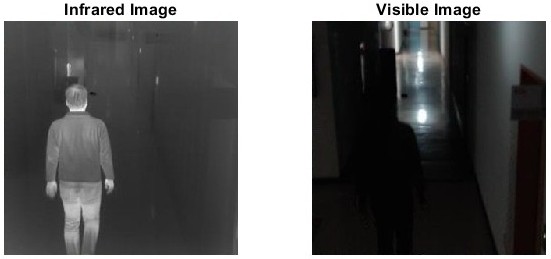
% Preprocess Infrared Image

grayIR = rgb2gray(IR);

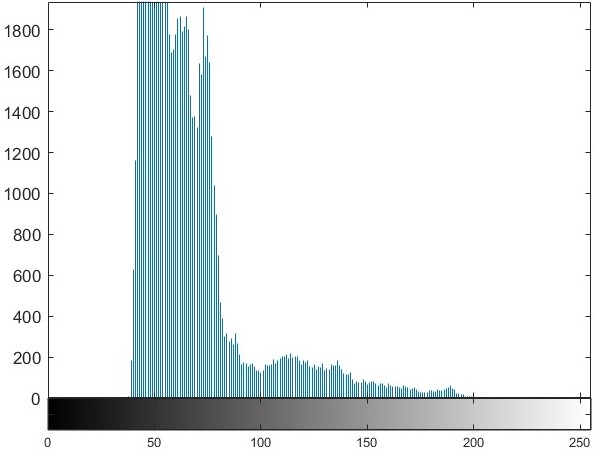
figure(2)

imhist(grayIR); title('Histogram of Infrared Grayscale Image');

**Output:**



**Fig. 1: Original Images**

****

**Fig. 2: Original Images**

## **Otsu Threshold**

Here we applied the Gaussian filter with variance of 2 to smoothen the Infrared image and calculate the Otsu Threshold for creating the binary mask.

**Code:**

smoothedIR = imgaussfilt(grayIR, 2); % Gaussian smoothing

level = graythresh(smoothedIR); % Otsu threshold

threshold = round(level \* 255);

fprintf('Computed Otsu Threshold: %d\n', threshold);

binaryMask = smoothedIR > threshold;

binaryMask = imclose(binaryMask, strel('disk', 5)); % Fill gaps

binaryMask = bwareaopen(binaryMask, 100); % Remove small fragments

## **Applying Binary Mask to Infrared Image:**

Now we applied the Binary Mask obtained above to the Infrared Image

**Code:**

% Apply Mask to IR Image

maskedIR = IR;

maskedIR(repmat(~binaryMask, [1 1 3])) = 0;

figure(3)

imshow(maskedIR); title('Masked IR Image (Auto ROI)');

**Output:**



**Fig. 3: Masked Infrared Image**

## **Salient Target and Background Mask:**

Now we created the Salient Target Mask for Thermal details and Background Mask for Background Structure.

**Code:**

% Create STM and BM Masks

stm = uint8(binaryMask) \* 255;

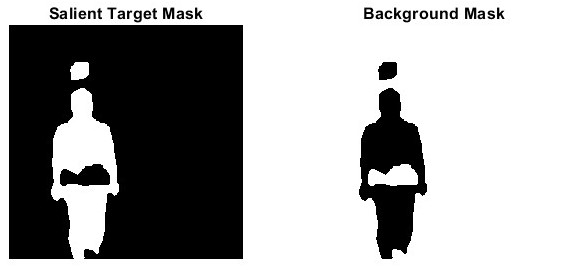
bm = uint8(~binaryMask) \* 255;

figure(4)

subplot(1,2,1); imshow(stm); title('Salient Target Mask');

subplot(1,2,2); imshow(bm); title('Background Mask');

**Output:**



**Fig. 4: Salient Target Mask and Background Mask**

## **Combining Masks with Infrared Image:**

Here we combined the Salient Target Mask and Background Mask with Infrared Image.

**Code:**

greyI = rgb2gray(IR);

result1 = greyI .\* uint8(binaryMask);

figure(5)

imshow(result1); title('Salient × Infrared');

result2 = greyI .\* uint8(~binaryMask);

figure(6)

imshow(result2); title('Background × Infrared');

**Output:**



**Fig. 5: Salient Target Mask with Infrared Image**

****

**Fig. 6: Background Mask with Infrared Image**

## **Pixel and Gradient Loss:**

Here we calculated the Pixel and Gradient Loss for the Salient region and Background areas.

**Code:**

% Visible Image Fusion

greyVIS = rgb2gray(VIS);

stmDouble = double(stm) / 255;

VIS\_double = double(VIS);

Id = uint8(stmDouble .\* double(greyVIS) + (1 - stmDouble) .\* VIS\_double);

## **Final Fusion:**

Here we combined the Salient Target Mask and Visible Image to obtain the desired result.

**Code:**

% Ensure RGB format

if size(Id, 3) == 1

Id\_rgb = cat(3, Id, Id, Id);

else

Id\_rgb = Id;

end

if size(maskedIR, 3) == 1

masked\_rgb = cat(3, maskedIR, maskedIR, maskedIR);

else

masked\_rgb = maskedIR;

end

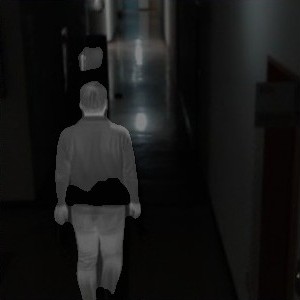
% Final Fusion

fusedFinal = uint8(0.5 \* double(masked\_rgb) + 0.5 \* double(Id\_rgb));

figure(7)

imshow(fusedFinal); title('Final Fused Output (Auto ROI + Otsu)');

**Output:**



**Fig. 7: Final Fused Image**

## **Enhancement Using Convolution:**

Here we applied the Gaussian Filter with variance 1 to the Final Fused Image.

**Code:**

% Simulated Convolutional Enhancement

conv1x1\_1 = fusedFinal; % Simulated 1×1 conv

conv3x3 = imgaussfilt(conv1x1\_1, 1); % Simulated 3×3 conv

conv1x1\_2 = conv3x3; % Simulated 1×1 conv

convEnhanced = uint8(0.5 \* double(fusedFinal) + 0.5 \* double(conv1x1\_2));

figure(8)

imshow(convEnhanced); title('Simulated Convolutional Enhancement Output');

**Output:**



**Fig. 8: Final Fused Image with Enhancement**

## **Loss Function Evaluation:**

**Code:**

% Loss Function Evaluation

fusedGray = rgb2gray(convEnhanced);

refGray = rgb2gray(VIS); % Reference can be VIS, IR, or maskedIR

% SSIM Loss

ssimVal = ssim(fusedGray, refGray);

L\_ssim = 1 - ssimVal;

% Gradient Loss

Gx\_fused = imgradient(fusedGray, 'sobel');

Gx\_ref = imgradient(refGray, 'sobel');

L\_grad = mean(abs(double(Gx\_fused) - double(Gx\_ref)), 'all') / 255;

% Total Loss

L\_total = L\_ssim + L\_grad;

fprintf('\n--- Fusion Loss Evaluation ---\n');

fprintf('SSIM Loss : %.4f\n', L\_ssim);

fprintf('Gradient Loss : %.4f\n', L\_grad);

fprintf('Total Loss : %.4f\n', L\_total);

**Output:**

--- Fusion Loss Evaluation ---

SSIM Loss : 0.3162

Gradient Loss : 0.0599

Total Loss : 0.3762

**Method – 2: Wavelet (dwt)**

## Loading and Preprocess Infrared and Visible Image:

Both infrared and Visible images were loaded and converted to gray scale and then to double for further processing.

**Code:**

% Read and preprocess images

IR = imread("manWalkIR.jpg");

VIS = imread("manWalkVB.jpg");

% original images

figure(1)

subplot(1,2,1); imshow(IR, []); title('Infrared Image');

subplot(1,2,2); imshow(VIS, []); title('Visible Image');

% Convert to grayscale if necessary

if size(IR,3)==3

IR = rgb2gray(IR);

end

if size(VIS,3)==3

VIS = rgb2gray(VIS);

end

% Resize to same size

[rows, cols] = size(IR);

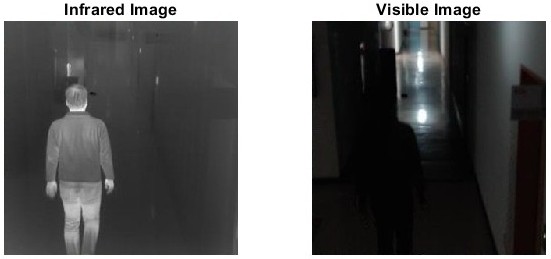
VIS = imresize(VIS, [rows cols]);

% Convert to double

IR = im2double(IR);

VIS = im2double(VIS);

**Output:**



**Fig. 1: Original Images**

## **Applying single level Discrete Wavelet Transform:**

Here we applied the single level discrete wavelet transform to both the infrared and visible images. By doing so we got the Approximation, Horizontal, Vertical and Diagonal components of both the images. Here we used the ‘db2’ wavelet.

**Code:**

% Apply single-level DWT

[LL\_IR, LH\_IR, HL\_IR, HH\_IR] = dwt2(IR, 'db2');

[LL\_VIS, LH\_VIS, HL\_VIS, HH\_VIS] = dwt2(VIS, 'db2');

% infrared image components

figure(2)

subplot(2,2,1); imshow(LL\_IR, []); title('Approximation (LL)');

subplot(2,2,2); imshow(LH\_IR, []); title('Horizontal Detail (LH)');

subplot(2,2,3); imshow(HL\_IR, []); title('Vertical Detail (HL)');

subplot(2,2,4); imshow(HH\_IR, []); title('Diagonal Detail (HH)');

% visible image components

figure(3)

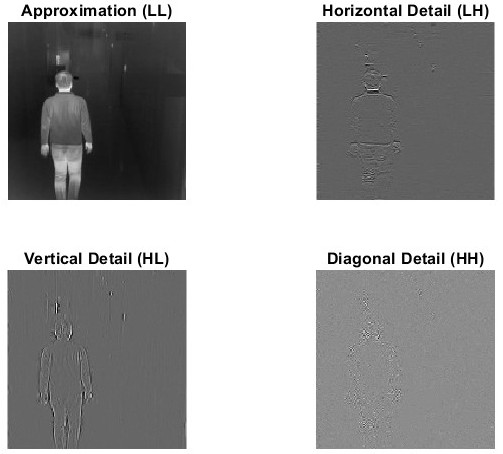
subplot(2,2,1); imshow(LL\_VIS, []); title('Approximation (LL)');

subplot(2,2,2); imshow(LH\_VIS, []); title('Horizontal Detail (LH)');

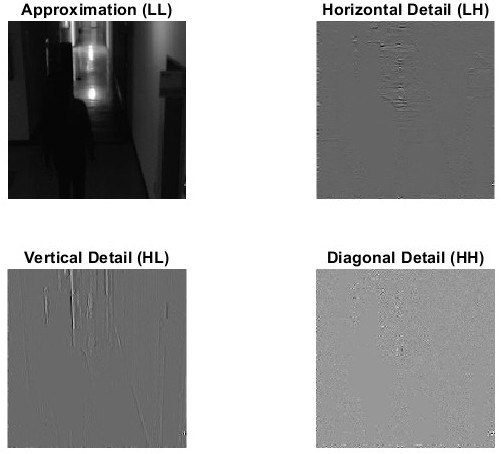
subplot(2,2,3); imshow(HL\_VIS, []); title('Vertical Detail (HL)');

subplot(2,2,4); imshow(HH\_VIS, []); title('Diagonal Detail (HH)');

**Output:**



**Fig. 2: Components of Infrared Image**



**Fig. 3: Components of Visible Image**

## **Fusion of coefficients:**

Now we fused the components of both the images by taking the maximum among infrared and fused image components.

**Code:**

% Fuse coefficients

LL\_fused = (LL\_IR + LL\_VIS) / 2; % average of approximations

LH\_fused = max(LH\_IR, LH\_VIS); % max for detail coefficients

HL\_fused = max(HL\_IR, HL\_VIS);

HH\_fused = max(HH\_IR, HH\_VIS);

## **Reconstruction of Fused image:**

Here we applied the inverse discrete wavelet transform to the fused components and combine them together to form the final fused image.

Code:

% Reconstruct fused image

Fused = idwt2(LL\_fused, LH\_fused, HL\_fused, HH\_fused, 'db2');

## **Displaying the results:**

**Code:**

% Display results

% fused image

figure(4)

imshow(Fused, [])

title("Fused Image")

**Output:**



**Fig. 4: Final Fused Image**

1. **Evaluation Matrix**

|  |  |  |  |
| --- | --- | --- | --- |
| **Methods** | **EN** | **MI** | **SF** |
| STDFusionNet (Paper) | 7.1978 ± 0.4793 | 3.7416 ± 0.5181 |  |
| STDFusionNet (Self) | 5.6256 | 4.2588 |  |
| Wavelet (dwt) | 6.8057 | 3.2315 |  |
| Wavelet (wavedec2) | 7.9723 | 2.9511 |  |