FUSION OF INFRARED AND VISIBLE IMAGE USING DIFFERENT METHODS

SHUBHAM KUMAR SAH	2341019010
JATIN JIHIRSU	2341010101

Content

1. Method – 1: STDFusionNet

1.1 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:

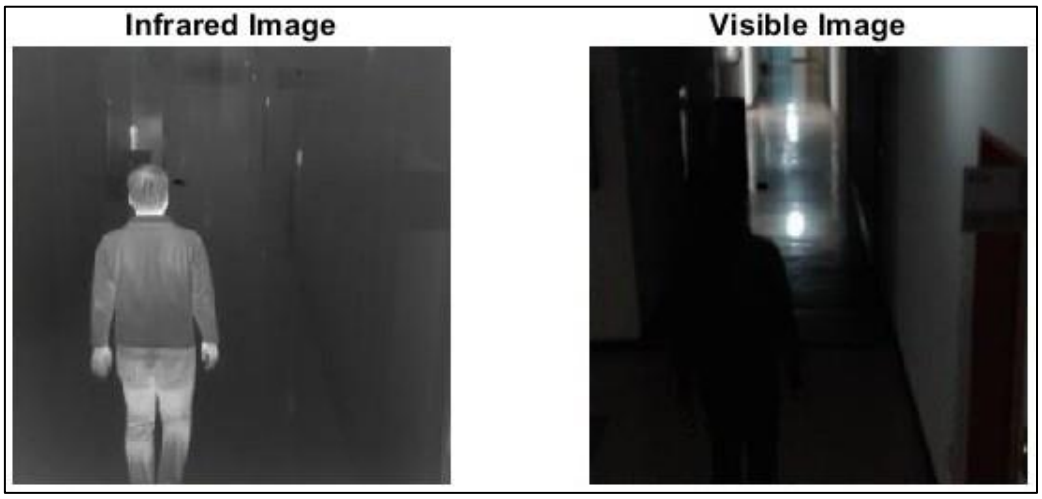

Fig. 1.1: Original Images

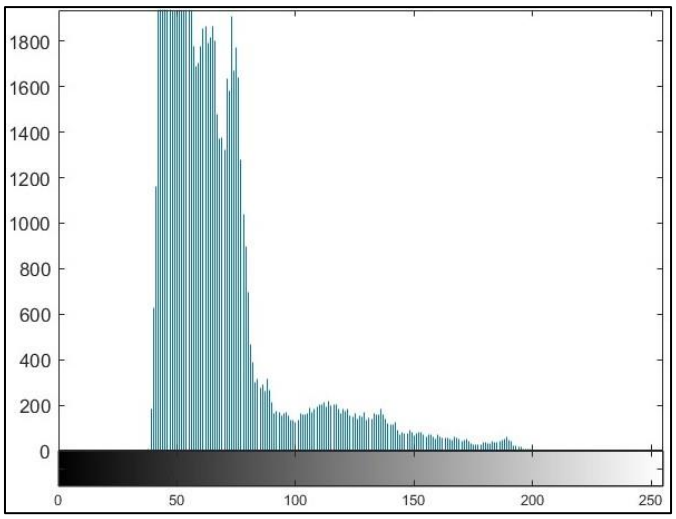

Fig. 1.2: Original Images

1.2 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
```

1.3 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:

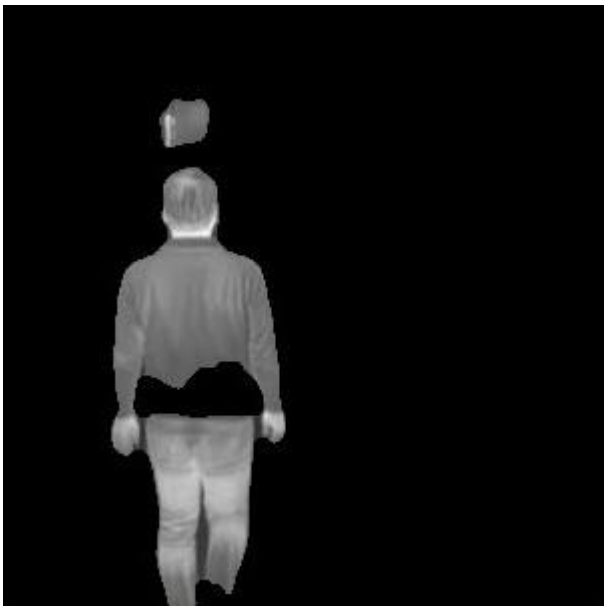

Fig. 1.3: Masked Infrared Image

1.4 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:

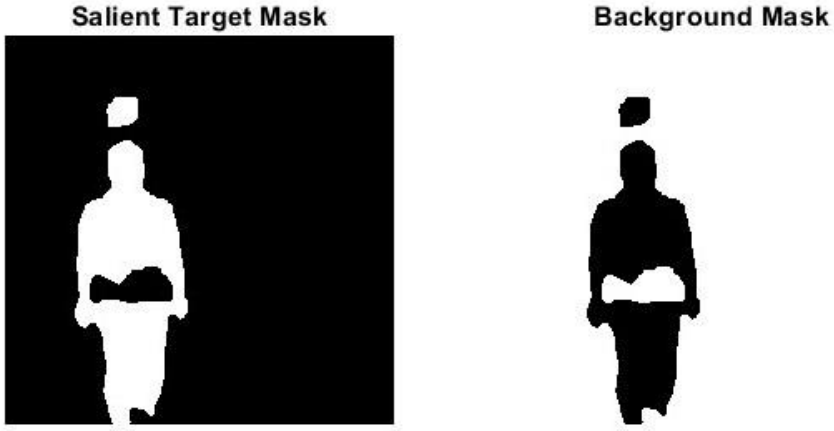

Fig. 1.4: Salient Target Mask and Background Mask

1.5 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:

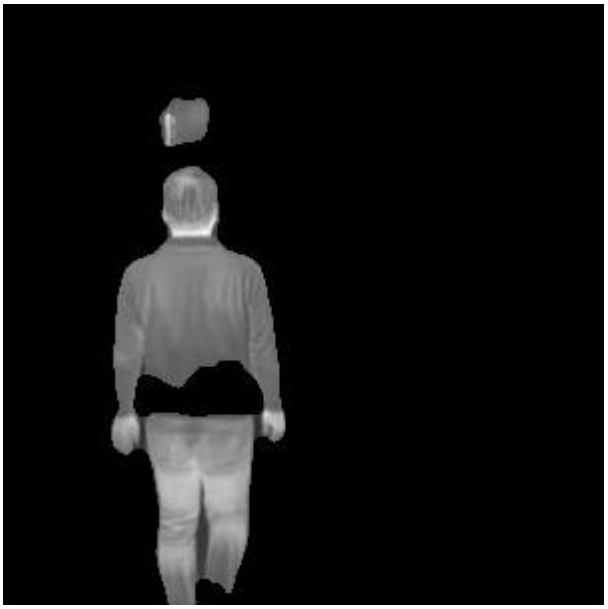

Fig. 1.5: Salient Target Mask with Infrared Image

Fig. 1.6: Background Mask with Infrared Image

1.6 Blend of Visible part with Salient Target:

Here we mix the visible image based on the important area (the mask). Where the object is present, it keeps the grayscale version (to match with infrared). Where the background is, it keeps the colour version.

Code:

```
% Visible Image Fusion
greyVIS = rgb2gray(VIS);
stmDouble = double(stm) / 255;
VIS_double = double(VIS);

Id = uint8(stmDouble .* double(greyVIS) + (1 - stmDouble) .* VIS_double);
```

1.7 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. 1.7: Final Fused Image

1.8 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. 1.8: Final Fused Image with Enhancement

1.9 Loss Function Evaluation:

Here we calculated the SSIM (Structural Similarity Index Measure) i.e., how similar is the fused image is with visible image and Gradient Loss i.e., edges and details of both images. And finally the Total Loss.

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
```

2. Method – 2: Wavelet (dwt)

2.1 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:

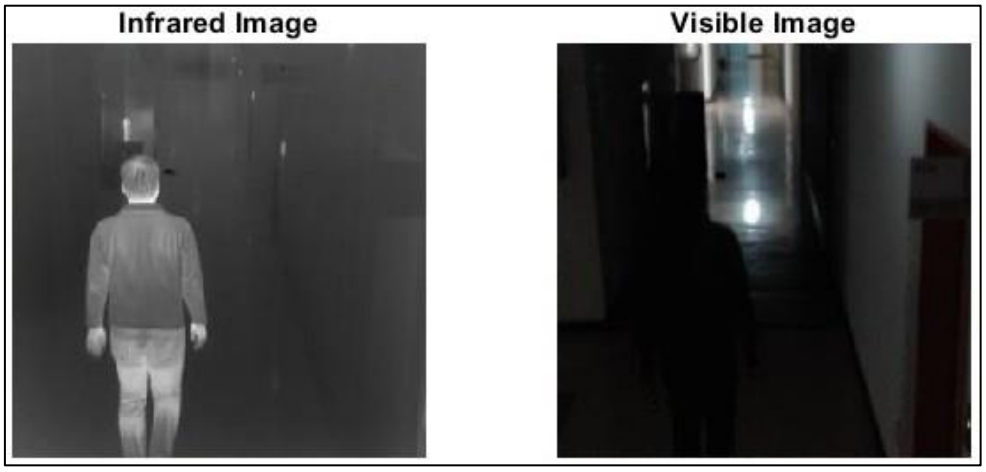

Fig. 2.1: Original Images

2.2 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:

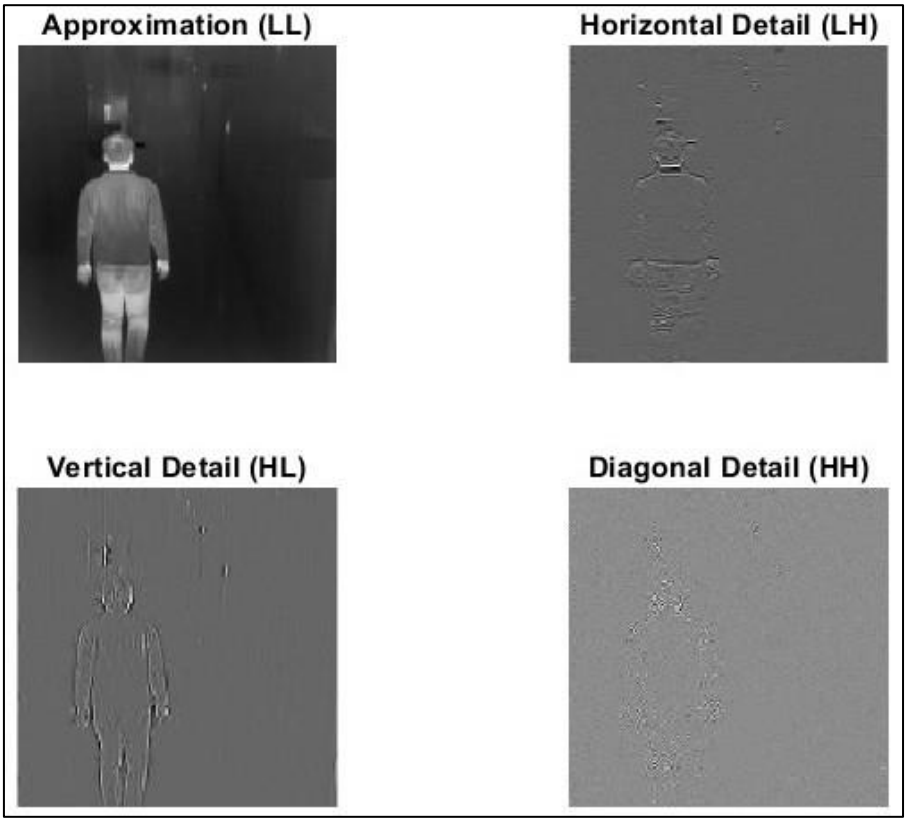

Fig. 2.2: Components of Infrared Image

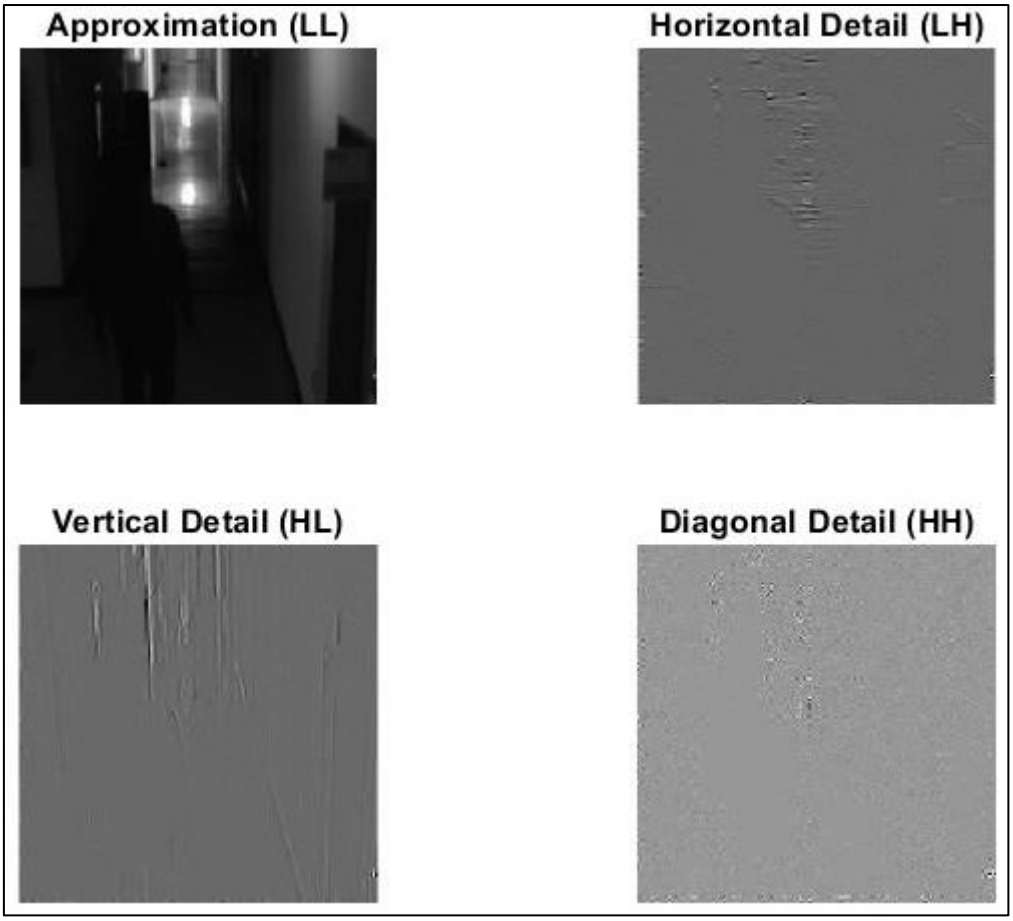

Fig. 2.3: Components of Visible Image

2.3 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);
```

2.4 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');
```

2.5 Displaying the results:

Code:

```
% Display results

% fused image
figure(4)
imshow(Fused, [])
title("Fused Image")
```

Output:

Fig. 2.4: Final Fused Image

3. Method – 3: Wavelet (wavedec2)

3.1 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");
if size(IR,3)==3
    IR = rgb2gray(IR);
end

if size(VIS,3)==3
    VIS = rgb2gray(VIS);
end

VIS = imresize(VIS, size(IR));
IR = im2double(IR);
VIS = im2double(VIS);

figure(1)
subplot(1,2,1); imshow(IR, []); title('Infrared Image');
subplot(1,2,2); imshow(VIS, []); title('Visible Image');
```

Output:

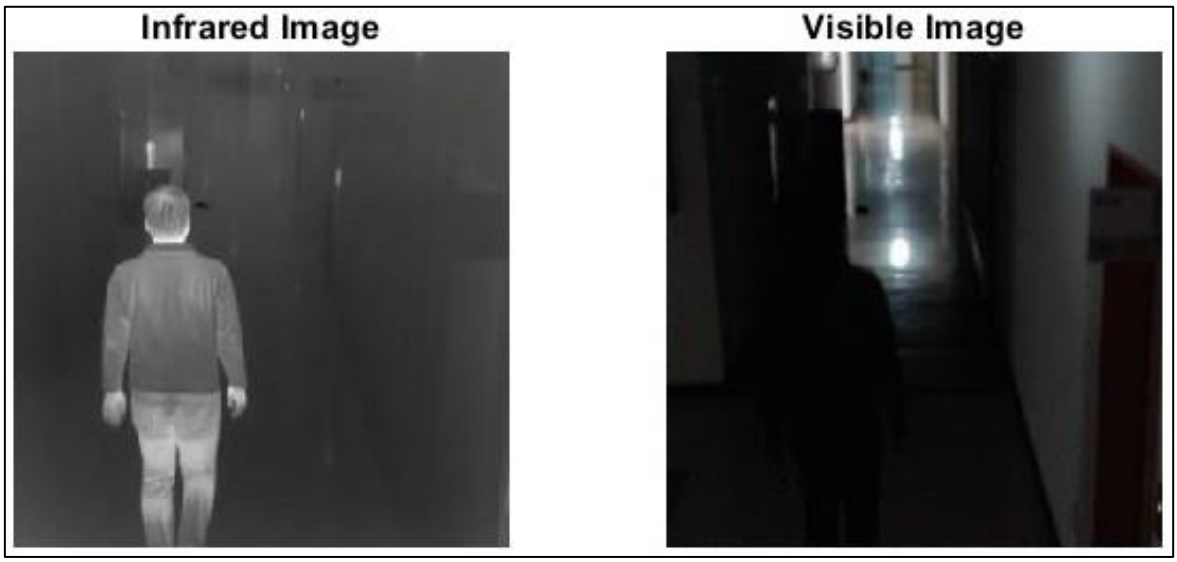

Fig. 3.1: Original Images

3.2 Multilevel Decomposition:

Here we performed 2-level wavelet decomposition using Daubechies-2 (db2) wavelet. Each image is decomposed into Approximation (LL) and Details (LH, HL and HH).

Code:

```
% Multi-level wavelet decomposition
level = 2;
waveletName = 'db2';
[C_IR, S_IR] = wavedec2(IR, level, waveletName);
[C_VIS, S_VIS] = wavedec2(VIS, level, waveletName);
```

3.3 Fusion of Approximation:

Now we fused the Approximation (LL) of both the images by taking the average of infrared and fused image components.

Code:

```
% Fuse coefficients
C_fused = C_IR; % Start with IR coefficients

% Fuse approximation (LL) coefficients
approx_len = prod(S_IR(1,:));
A_IR = C_IR(1:approx_len);
A_VIS = C_VIS(1:approx_len);
A_fused = (A_IR + A_VIS) / 2;
C_fused(1:approx_len) = A_fused;
```

3.4 Fusion of Details:

Now we fused the Details (LH, HL and HH) and of both the images by taking the average of infrared and fused image components.

Code:

```
% Fuse detail coefficients level by level
start = approx_len + 1;
for i = 1:level
    sz = S_IR(i+1,:); num = prod(sz);

    % Horizontal detail indices (LH)
    H_IR = C_IR(start : start+num-1);
    H_VIS = C_VIS(start : start+num-1);
    H_fused = max(abs(H_IR), abs(H_VIS)) .* sign(H_IR + H_VIS);
    C_fused(start : start+num-1) = H_fused;
    start = start + num;
```

```
    % Vertical detail indices (HL)
    V_IR = C_IR(start : start+num-1);
    V_VIS = C_VIS(start : start+num-1);
    V_fused = max(abs(V_IR), abs(V_VIS)) .* sign(V_IR + V_VIS);
    C_fused(start : start+num-1) = V_fused;
    start = start + num;

    % Diagonal detail indices (HH)
    D_IR = C_IR(start : start+num-1);
    D_VIS = C_VIS(start : start+num-1);
    D_fused = max(abs(D_IR), abs(D_VIS)) .* sign(D_IR + D_VIS);
    C_fused(start : start+num-1) = D_fused;
    start = start + num;
end
```

3.5 Displaying the results:

Code:

```
% Reconstruct fused image
Fused = waverec2(C_fused, S_IR, waveletName);

figure(2)
imshow(Fused, []); title('Fused Image (Raw)');
```

Output:

Fig. 3.2: Final Fused Image

3.6 Displaying the Enhanced Result:

Code:

```
% Enhance and display
Fused_eq = histeq(Fused);

figure(3)
imshow(Fused_eq, []); title('Fused Image (Enhanced)');
```

Output:

Fig. 3.3: Final Fused Image

4. 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	