* **Fusion of Infrared and Visible Image**

In this paper we propose a method for fusion of Infrared and Visible Image by preserving the thermal targets from infrared image and background structure from visible image.

Here is the step used in fusion of infrared and visible image:

1. **Displaying Infrared and Visible Image.**

* **Code**

IR = imread("manWalkIR.jpg");

VIS = imread("manWalkVB.jpg");

* **Output**



1. **Generating Histogram of Infrared image.**

Converting Infrared image to gray scale for thresholding and then generating its histogram.

* **Code**

% Convert IR to grayscale only for thresholding (safe for both RGB or gray)

grayIR = im2gray(IR); % replaces rgb2gray safely

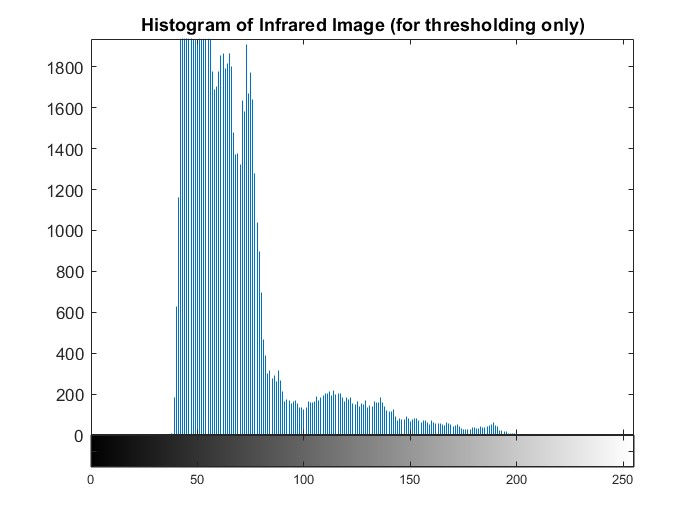
% Display histogram

figure(2)

imhist(grayIR);

title('Histogram of Infrared Image (for thresholding only)');

* **Output**



1. **Gaussian Filter and Otsu Threshold**

Here we apply the gaussian filter to the infrared image to reduce the noise in infrared image. Then we computed the Otus Threshold which helped in creating a binary mask. This binary mask is applied to the infrared image so that the target object in the infrared image is separated from the original infrared image.

* **Code**

% Apply Gaussian smoothing to reduce noise (on grayIR)

smoothedIR = imgaussfilt(grayIR, 2); % sigma = 2

% Compute Otsu threshold

level = graythresh(smoothedIR); % returns normalized threshold [0,1]

threshold = round(level \* 255); % scale to [0,255]

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

% Create binary mask using threshold

binaryMask = smoothedIR > threshold;

% Morphological closing to fill gaps

binaryMask = imclose(binaryMask, strel('disk', 5)); % fill small holes

% Remove small fragments

binaryMask = bwareaopen(binaryMask, 100); % remove objects < 100 pixels

% Apply mask to IR image

if size(IR,3) == 3

maskedIR = IR;

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

else

maskedIR = IR;

maskedIR(~binaryMask) = 0;

end

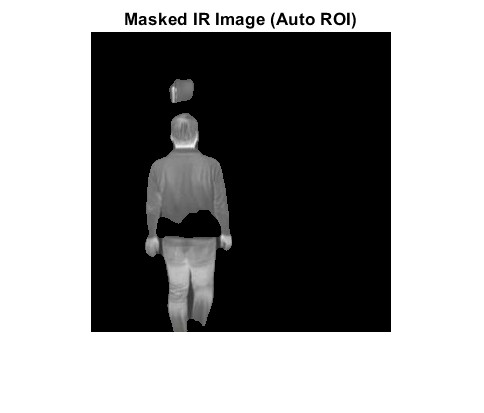
% Display masked IR image

figure(3)

imshow(maskedIR);

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

* **Output**



1. **Salient Target Mask and Background Target Mask**

Here we have salient target mask is created to preserve the thermal targets of infrared image and background target mask to preserve the background structure of visible image.

**Code:**

% Create Salient Target Mask (STM) and Background Mask (BM)

stm = uint8(binaryMask) \* 255;

bm = uint8(~binaryMask) \* 255;

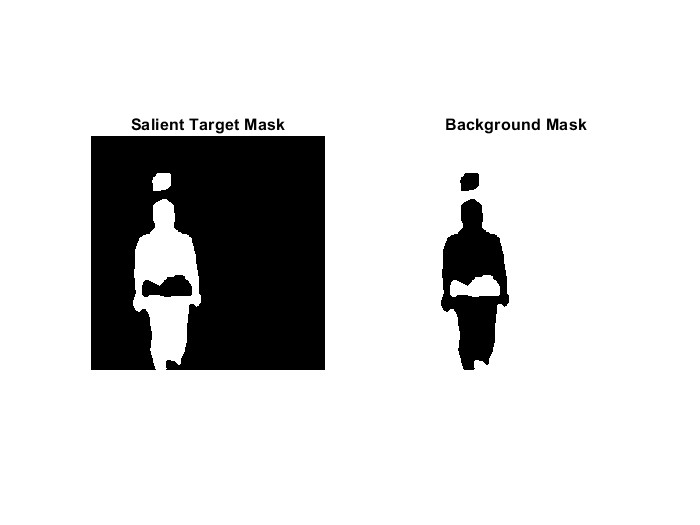
% Display masks

figure(4)

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

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

**Output:**



1. **Combining Salient Target mask and Background mask with Infrared image**

**Code:**

% Element-wise multiplication with IR image

if size(IR,3) == 3

result1 = uint8(double(IR) .\* repmat(binaryMask, [1 1 3]));

result2 = uint8(double(IR) .\* repmat(~binaryMask, [1 1 3]));

else

result1 = uint8(double(IR) .\* double(binaryMask));

result2 = uint8(double(IR) .\* double(~binaryMask));

end

figure(5)

imshow(result1);

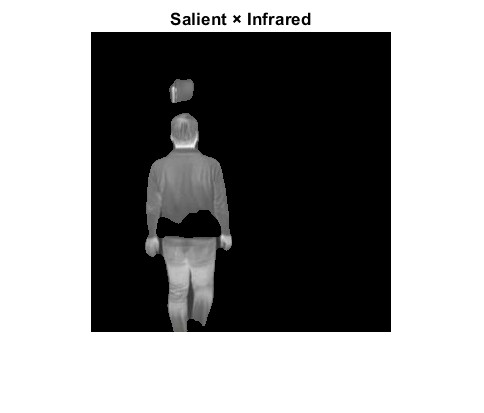
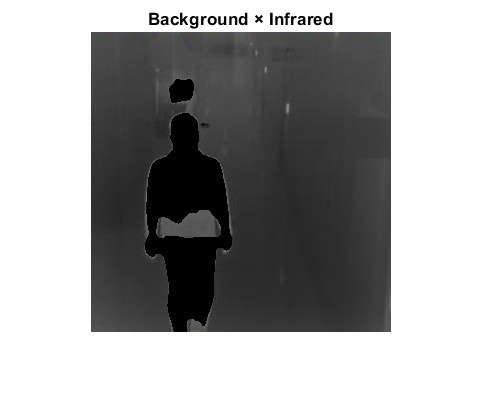
title('Salient × Infrared');

figure(6)

imshow(result2);

title('Background × Infrared');

**Output:**

1. **Fusion of Salient Target Mask and Visible image**

Here we take the Visible image and do fusion using Salient Target mask, so that it combines the Background Texture and Thermal Target and increase the visibility of the image.

**Code:**

% Prepare visible image

VIS\_double = double(VIS);

stmDouble = double(stm) / 255;

% Match mask dimensions to visible image

if size(VIS,3) == 3

stmRGB = repmat(stmDouble, [1 1 3]);

else

stmRGB = stmDouble;

end

% Fusion using mask

Id\_rgb = uint8(stmRGB .\* VIS\_double + (1 - stmRGB) .\* VIS\_double);

% Weighted fusion

if size(maskedIR,3) == 3

masked\_rgb = maskedIR;

else

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

end

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

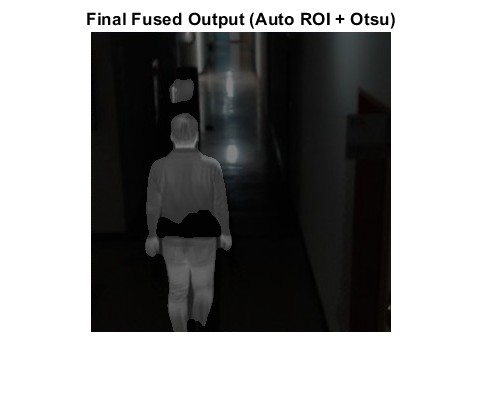
% Display final fused image

figure(7)

imshow(fusedFinal);

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

**Output:**



1. **Increasing the brightness of fused image**

Here we use the log transform to increase the brightness of the image so that the darker background region is more clearly visible.

**Code:**

% Brighten

image = double(fusedFinal);

c = 1;

image\_bright = c \* log(1 + image);

image\_bright = mat2gray(image\_bright); % Normalize

figure(8)

imshow(image\_bright);

title('Brighten');

**Output:**



1. **Sharpening image**

Finally, we sharpen the image to enhance its fine details and edges.

**Code:**

% Sharpen

image\_sharp = imsharpen(image\_bright, 'Radius', 2, 'Amount', 1);

figure(9)

imshow(image\_sharp);

title('Sharp');

**Output:**

