Table of Contents

```
% -------------------------------------------------
% Shannon–Fano Image Compression & Decompression
% -------------------------------------------------
% This script demonstrates Shannon–Fano coding for
% lossless grayscale image compression.
%
% Steps:
% 1. Select a grayscale image (portable method)
% 2. Compute pixel frequencies
% 3. Generate Shannon–Fano codes
% 4. Encode image
% 5. Decode image
% 6. Display original and decoded images
% 7. Show compression statistics
%
% Author: Sandesh + ChatGPT
% -------------------------------------------------

clc;
clear;
close all;
```

STEP 1: Read grayscale image (BEST FIX)

```
[file, path] = uigetfile({'*.jpg;*.png;*.bmp;*.tif', ...
                          'Image Files (*.jpg, *.png, *.bmp, *.tif)'}, ...
                          'Select a grayscale image');

if isequal(file,0)
    error('Image selection cancelled. Please select an image file.');
end

img = imread(fullfile(path, file));
```

```
if size(img,3) == 3
    img = rgb2gray(img);
end

img = uint8(img);
```

STEP 2: Flatten image into 1D array

```
pixels = img(:);
```

STEP 3: Compute symbol frequencies

```
symbols = unique(pixels);
freq = zeros(length(symbols),1);

for i = 1:length(symbols)
    freq(i) = sum(pixels == symbols(i));
end
```

STEP 4: Sort symbols by descending frequency

```
[freq, idx] = sort(freq, 'descend');
symbols = symbols(idx);
```

STEP 5: Initialize Shannon–Fano code map

```
codes = containers.Map('KeyType','double','ValueType','char');
for i = 1:length(symbols)
    codes(symbols(i)) = '';
end
```

STEP 6: Generate Shannon–Fano codes

```
codes = shannonFanoRecursive(symbols, freq, codes);
```

STEP 7: Encode image

```
encoded_bits = '';
for i = 1:length(pixels)
    encoded_bits = [encoded_bits codes(double(pixels(i)))]; %#ok<AGROW>
end
```

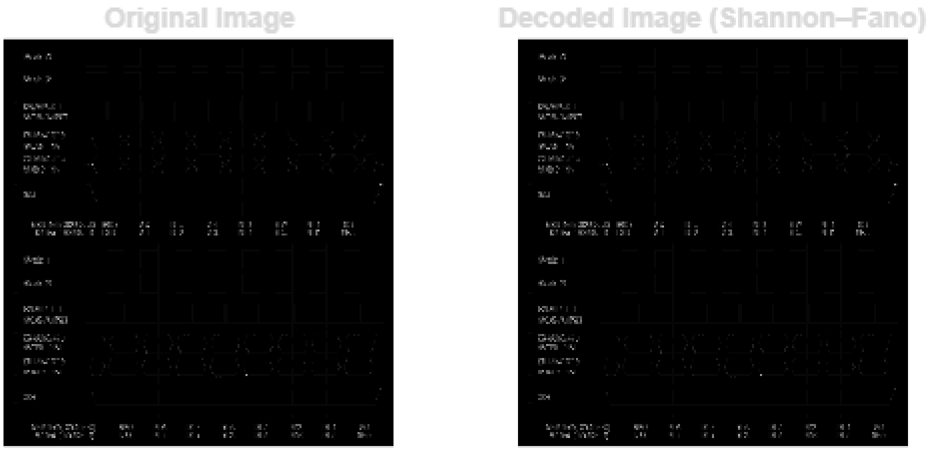
STEP 8: Decode bitstream

```
decoded_pixels = zeros(length(pixels),1,'uint8');
current_bits = '';
count = 1;
```

```
% Reverse code map
code_keys = keys(codes);
code_values = values(codes);
reverse_codes = containers.Map(code_values, code_keys);

for i = 1:length(encoded_bits)
    current_bits = [current_bits encoded_bits(i)];
    if isKey(reverse_codes, current_bits)
        decoded_pixels(count) = uint8(reverse_codes(current_bits));
        count = count + 1;
        current_bits = '';
    end
end

decoded_img = reshape(decoded_pixels, size(img));
```

STEP 9: Display results

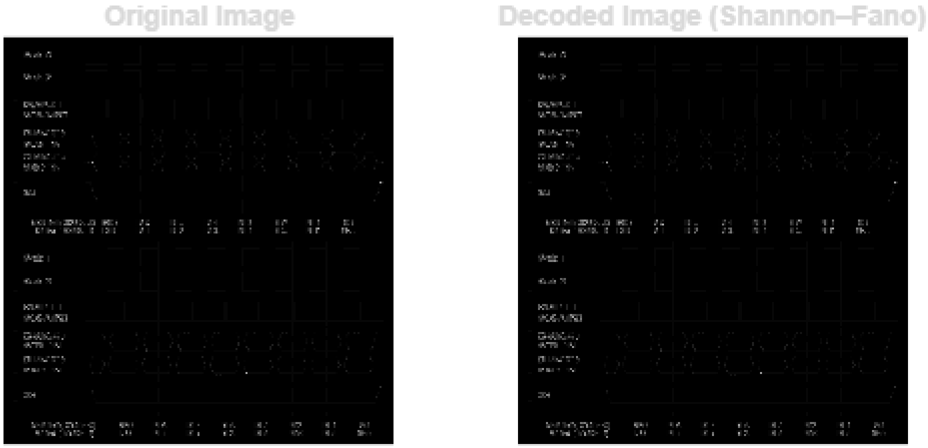
```
figure('Name','Shannon-Fano Image Compression','Color','w');

subplot(1,2,1);
imshow(img);
title('Original Image');
axis off;

subplot(1,2,2);
imshow(decoded_img);
title('Decoded Image (Shannon-Fano)');
axis off;
```



STEP 10: Compression statistics

```
original_size_bits   = numel(pixels) * 8;
compressed_size_bits = length(encoded_bits);
compression_ratio    = original_size_bits / compressed_size_bits;

fprintf('\n--- Compression Statistics ---\n');
fprintf('Original size (bits):   %d\n', original_size_bits);
fprintf('Compressed size (bits): %d\n', compressed_size_bits);
fprintf('Compression Ratio:      %.2f\n', compression_ratio);
fprintf('------------------------------\n');
```

```
--- Compression Statistics ---
Original size (bits):   3834768
Compressed size (bits): 687108
Compression Ratio:      5.58
------------------------------
```


Original Image

Decoded Image (Shannon–Fano)

Local Function: Shannon–Fano Recursive Split

```
function codes = shannonFanoRecursive(symbols, freq, codes)

    % Base case
    if length(symbols) <= 1
        return;
    end

    % Total frequency
    total_freq = sum(freq);

    % Find split point
    acc_freq = 0;
    split_index = 1;

    for i = 1:length(freq)
        acc_freq = acc_freq + freq(i);
        if acc_freq >= total_freq / 2
            split_index = i;
            break;
        end
    end

    % Assign 0 to first group
    for i = 1:split_index
        codes(symbols(i)) = [codes(symbols(i)) '0'];
    end

    % Assign 1 to second group
    for i = split_index+1:length(symbols)
        codes(symbols(i)) = [codes(symbols(i)) '1'];
    end

    % Recursive calls
    codes = shannonFanoRecursive(symbols(1:split_index), ...
                                 freq(1:split_index), codes);

    codes = shannonFanoRecursive(symbols(split_index+1:end), ...
                                 freq(split_index+1:end), codes);
end
```

Published with MATLAB® R2025b