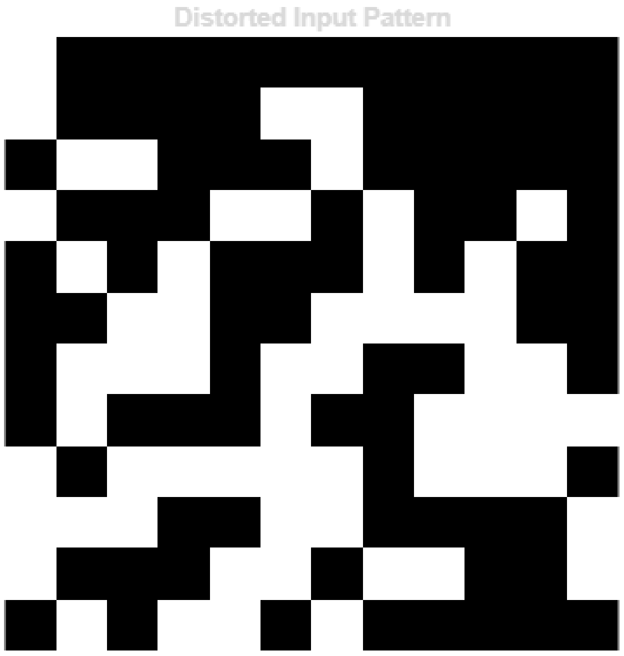
## Table of Contents

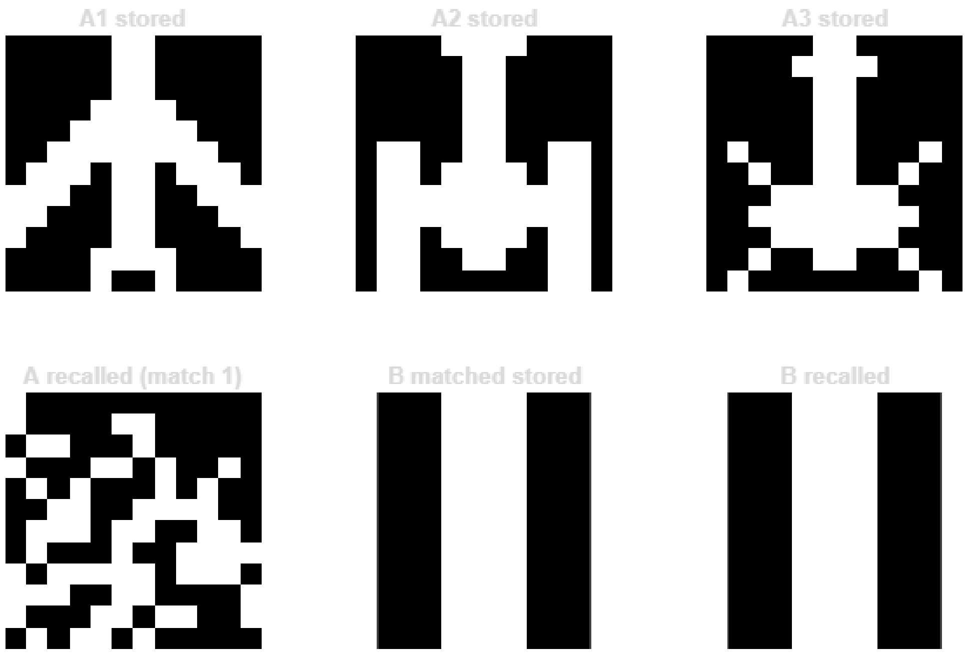
```
% FML_LAB_8
% Author: ARISH
% Q1) BAM for image patterns: load, preprocess, encode, recall

clear; clc; close all;

% Configuration
% Paths to paired images (replace these with your actual files)
% Group A: 12x12 images (arrow, robot, rocket)
A_paths = {
    'Plane.png'  % 12x12 arrow-like
    'Tank.png'  % 12x12 robot-like
    'Helicopter.png'  % 12x12 rocket/tower
};

% Group B: 12x10 images (bar, digit 2, digit 3)
B_paths = {
    'P1.png'  % 12x10 vertical bar
    'T2.png'  % 12x10 digit "2"
    'H3.png'  % 12x10 digit "3"
};

% Distorted image to identify (12x12)
distorted_path = 'D2.png';

% Preprocessing parameters
bw_threshold = 0.5;   % Otsu used if empty ([]); else fixed threshold in
[0,1]
invert = false;       % Set true if foreground/background are reversed
open_radius = 0;      % Morphological opening radius (0 = disable)

% Helpers
to_bipolar = @(M) 2*double(M) - 1;

%  Load and preprocess images
XA = []; YA = [];  % will hold bipolar vectors for groups A and B

for i = 1:numel(A_paths)
    IA = preprocess_to_grid(A_paths{i}, [12 12], bw_threshold, invert,
open_radius);
    IB = preprocess_to_grid(B_paths{i}, [12 10], bw_threshold, invert,
open_radius);

    XA(:,i) = to_bipolar(IA(:));  % 144x1
    YA(:,i) = to_bipolar(IB(:));  % 120x1
end
```

```
% Compute BAM weight matrix (outer-product encoding)
W = zeros(size(XA,1), size(YA,1));  % 144x120
for i = 1:size(XA,2)
    W = W + XA(:,i) * YA(:,i)';     % sum of outer products
end

% Load distorted image and bipolarize
ID = preprocess_to_grid(distorted_path, [12 12], bw_threshold, invert,
open_radius);
x = to_bipolar(ID(:));              % 144x1

% BAM recall dynamics
max_iter = 20;
for iter = 1:max_iter
    y = sign(W' * x);               % forward pass to B
    x_new = sign(W * y);            % backward pass to A
    if isequal(x_new, x), break; end
    x = x_new;
end

% Match recalled x to stored XA and y to YA
[matchA_idx, simA] = best_match(x, XA);
[matchB_idx, simB] = best_match(y, YA);

fprintf('Recalled A-pattern best match: #%d (similarity = %.0f)\n',
matchA_idx, simA);
fprintf('Associated B-pattern best match: #%d (similarity = %.0f)\n',
matchB_idx, simB);

%  Load distorted image and bipolarize
ID = preprocess_to_grid(distorted_path, [12 12], bw_threshold, invert,
open_radius);
x = to_bipolar(ID(:));              % 144x1

% Show distorted image before recall
figure('Name','Distorted Input','Color','w');
imshow(ID);
title('Distorted Input Pattern');

% Visualize
figure('Name','BAM Recall','Color','w');

subplot(2,3,1); imshow(reshape(to_binary_bipolar(XA(:,1)), [12 12]));
title('A1 stored');
subplot(2,3,2); imshow(reshape(to_binary_bipolar(XA(:,2)), [12 12]));
title('A2 stored');
subplot(2,3,3); imshow(reshape(to_binary_bipolar(XA(:,3)), [12 12]));
title('A3 stored');

subplot(2,3,4); imshow(ID); title('Distorted Input');

subplot(2,3,4); imshow(reshape(to_binary_bipolar(x), [12 12]));
title(sprintf('A recalled (match %d)', matchA_idx));
```

```
subplot(2,3,5); imshow(reshape(to_binary_bipolar(YA(:,matchB_idx)), [12
10])); title('B matched stored');
subplot(2,3,6); imshow(reshape(to_binary_bipolar(y), [12 10])); title('B
recalled');

% Utility functions
function Igrid = preprocess_to_grid(path, target_size, bw_threshold, invert,
open_radius)
    % Read
    I = imread(path);
    if size(I,3) > 1, I = rgb2gray(I); end
    I = im2double(I);

    % Resize to target grid
    I = imresize(I, target_size, 'nearest');

    % Binarize
    if isempty(bw_threshold)
        level = graythresh(I);  % Otsu
        BW = imbinarize(I, level);
    else
        BW = imbinarize(I, bw_threshold);
    end

    % Optional invert
    if invert
        BW = ~BW;
    end

    % Optional morphological opening to clean speckle
    if open_radius > 0
        se = strel('disk', open_radius, 0);
        BW = imopen(BW, se);
    end

    % Ensure logical
    Igrid = logical(BW);
end

function [idx, sim] = best_match(vec, bank)
    sims = sum(bank .* vec, 1);      % bipolar dot-product similarity
    [sim, idx] = max(sims);
end

function BW = to_binary_bipolar(bip)
    BW = bip > 0;                    % +1 -> 1, -1 -> 0
end
```

```
Recalled A-pattern best match: #1 (similarity = 144)
Associated B-pattern best match: #1 (similarity = 120)
```


Distorted Input Pattern


A1 stored
A2 stored
A3 stored
A recalled (match 1)
B matched stored
B recalled

## Q2) BAM Encoding and Recall with Bipolar Vectors

```
clear; clc;

% Original Binary Patterns
fprintf('\nQ2');
A1_bin = [0 1 0 1 0];
B1_bin = [1 0 0 1];

A2_bin = [1 1 0 0 0];
B2_bin = [0 1 0 1];

% Convert to Bipolar
to_bipolar = @(v) 2*v - 1;

X = [to_bipolar(A1_bin)', to_bipolar(A2_bin)'];  % 5×2
Y = [to_bipolar(B1_bin)', to_bipolar(B2_bin)'];  % 4×2

% Compute BAM Weight Matrix
W = zeros(size(X,1), size(Y,1));  % 5×4
for i = 1:size(X,2)
    W = W + X(:,i) * Y(:,i)';      % Outer product encoding
end

% i) Present X1 and find associate
x1 = X(:,1);
y1 = sign(W' * x1);                % Forward pass
fprintf('\ni) Associate of X1:\n');
disp(y1');

% ii) Probe patterns
X_star  = [-1; 1; 1; 1; -1];
X_star2 = [-1; -1; 1; 1; -1];

probe_set = {X_star, X_star2};
labels = {'X*', 'X**'};

for k = 1:length(probe_set)
    x_probe = probe_set{k};
    y_probe = sign(W' * x_probe);      % Forward pass
    x_recall = sign(W * y_probe);      % Backward pass

    fprintf('\nii) Probe %s:\n', labels{k});
    disp(['Recalled Y: ', mat2str(y_probe')]);
    disp(['Recalled X: ', mat2str(x_recall')]);

    % Compare Hamming distance to stored patterns
    for i = 1:size(X,2)
        hd = sum(x_probe ~= X(:,i));
        fprintf('Hamming distance to X%d: %d\n', i, hd);
    end
end
```

```
Q2
i) Associate of X1:
     1    -1    -1     1


ii) Probe X*:
Recalled Y: [1 -1 -1 1]
Recalled X: [-1 1 -1 1 -1]
Hamming distance to X1: 1
Hamming distance to X2: 3

ii) Probe X**:
Recalled Y: [1 -1 1 -1]
Recalled X: [-1 -1 1 1 1]
Hamming distance to X1: 2
Hamming distance to X2: 4
```

*Published with MATLAB® R2025a*