**Experiment-2.2**

**Aim of the Experiment :**

 Image matching using RANSAC (Random Sample Consensus) algorithm.

Problem Description :

RANSAC, which stands for Random Sample Consensus, is an iterative algorithm commonly used in computer vision and computer graphics to estimate parameters of a mathematical model from a set of observed data points that may contain outliers. The algorithm is particularly robust in the presence of outliers and can be applied to solve problems like fitting a model to noisy data. The RANSAC algorithm involves several steps, and while there isn't a specific formula for the entire algorithm, I can outline the key components and some formulas involved in each step:

1. Random Sampling: Select a minimal subset of data points to form an initial model hypothesis.
2. Model Fitting: Depending on the type of model being fitted, you'll use different formulas to estimate the model parameters. For example:

For a line (ax + by + c = 0), you might use the formula for the line equation y=mx+b. For a plane, you might use a formula like ax+by+cz+d=0.

1. Inlier Verification: Calculate the distance between each data point and the model. The formula for distance depends on the type of model and could be, for example:

For a line, you might use the perpendicular distance from a point to a line formula.

For a plane, you might use the distance from a point to a plane formula.

1. Model Evaluation: Count the number of inliers based on a predefined threshold.
2. Iteration: Repeat the random sampling, model fitting, inlier verification, and model evaluation steps for a certain number of iterations or until a termination condition is met.
3. Best Model Selection: Choose the model with the highest number of inliers as the final estimate.

It's important to note that the specific formulas will vary depending on the problem you are solving and the type of model you are trying to fit. The success of RANSAC often relies on selecting appropriate formulas for model fitting and inlier verification based on the characteristics of the data and the underlying model.

**Code :**

% Load two images to be merged

image1 = imread('image 6.jpg');

image2 = imread('image 5.jpg');

% Convert images to grayscale if they are not already

grayImage1 = rgb2gray(image1);

grayImage2 = rgb2gray(image2);

% Detect and extract features from both images

points1 = detectSURFFeatures(grayImage1);

points2 = detectSURFFeatures(grayImage2);

[features1, validPoints1] = extractFeatures(grayImage1, points1);

[features2, validPoints2] = extractFeatures(grayImage2, points2);

% Match features between the two images

indexPairs = matchFeatures(features1, features2);

matchedPoints1 = validPoints1(indexPairs(:, 1), :);

matchedPoints2 = validPoints2(indexPairs(:, 2), :);

[tform, inlierPoints1, inlierPoints2] = estimateGeometricTransform(matchedPoints1, matchedPoints2, 'affine');

% Refine the transformation using all inlier points

[tform, ~] = estimateGeometricTransform(inlierPoints1, inlierPoints2, 'affine');

% Use the transformation matrix to stitch the images

outputView = imref2d(size(grayImage1));

panorama = imwarp(image1, tform, 'OutputView', outputView);

panorama = imfuse(panorama, image2, 'blend', 'Scaling','joint');

% Display the stitched image

subplot(2,2,4);

imshow(panorama);

title('Stitched Image');

% Optionally, display the individual images and matched points

subplot(2, 2, 1);

imshow(image1);

title('Image 1');

subplot(2, 2, 2);

imshow(image2);

title('Image 2');

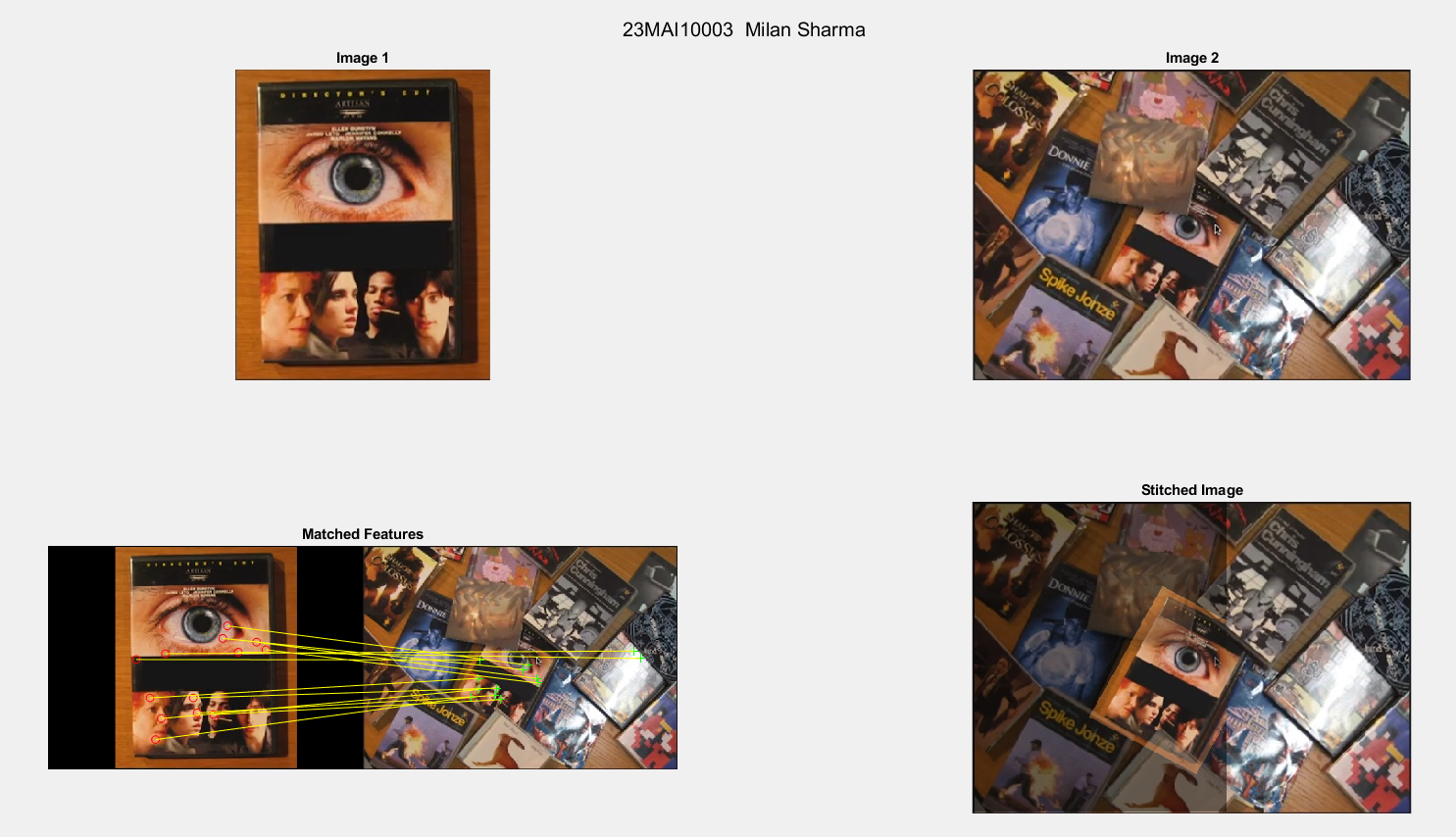
subplot(2, 2, 3);

showMatchedFeatures(image1, image2, matchedPoints1, matchedPoints2, 'montage');

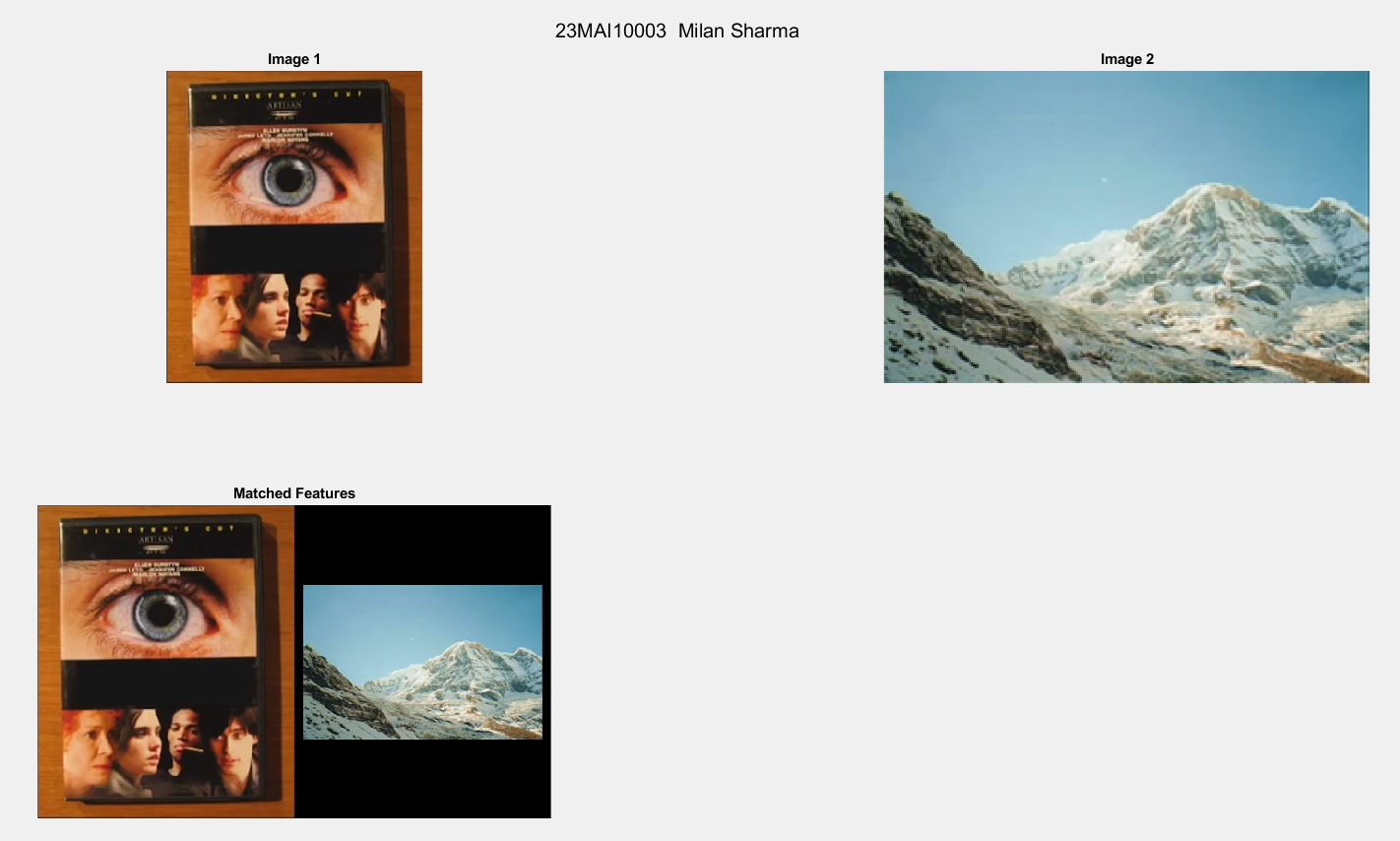
title('Matched Features');

sgtitle("23MAI10003 Milan Sharma")

**Output (For Matching Images) :**



**Output (For Unmatched Images):**



**Learning outcomes (What I have learnt):**

**1. Learnt about RANSAC (Random Sample Consensus).**

**2. Learnt how the key components are matched in the algorithm.**

**3. Learnt how to make matched points using RANSAC.**

**4. Learnt how to make Stitched image using two or more matched image.**