**CS 696 - Applied Computer Vision**

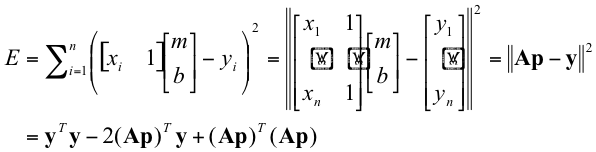
**INTRODUCTION :**

This project includes understanding the basics of single object tracking over time using KLT algorithm. Given the correspondences for the previous image and current image, we need to calculate the translation displacement matrix using Least Square Method.

**PART 1 : Estimate Translation displacement using Least Square Method**

Least Square Method is used to fit a line by minimizing the errors. Given n number of points, we need to convert them to standard form : Ax = b , in order to solve for translation matrix, solve x.

The following are the basic steps for Line fitting Algorithm :





**Filename : proj5.m**

%%%%%%%%%%%%%% Transformation matrix for previous image %%%%%%%%%%%%%%%

%%% step-4.1 : %%%

meanA = sum(matchedPoints1.Location)/size(matchedPoints1.Location,1);

meanA\_matrix = [1 0 -meanA(1,1); 0 1 -meanA(1,2); 0 0 1];

avgPointsA = matchedPoints1.Location - repmat(meanA,size(matchedPoints1.Location,1),1);

relDistA = sqrt(sum(avgPointsA.^2,2));

sa = sqrt(2)/mean(relDistA);

scaleA = [sa 0 0; 0 sa 0; 0 0 1];

%Transformation matrix Ta

Ta = scaleA \* meanA\_matrix;

%%%%%%%%%%%%%% Transformation matrix for current image %%%%%%%%%%%%%%%%

%%% step-4.2 : %%%

meanB = sum(matchedPoints2.Location)/size(matchedPoints2.Location,1);

meanB\_matrix = [1 0 -meanB(1,1); 0 1 -meanB(1,2); 0 0 1];

avgPointsB = matchedPoints2.Location - repmat(meanB,size(matchedPoints2.Location,1),1);

relDistB = sqrt(sum(avgPointsB.^2,2));

sb = sqrt(2)/mean(relDistB);

scaleB = [sb 0 0; 0 sa 0; 0 0 1];

%Transformation matrix Tb

Tb = scaleB \* meanB\_matrix;

%%%%%%%%%%%%%%%%%%%%%%%%%% Formulate A\_matrix %%%%%%%%%%%%%%%%%%%%%%%%%

%%% step-4.3 : %%%

A\_matrix = zeros(size(matchedPoints1\_norm,1),9);

for ii=1:size(matchedPoints1\_norm,1)

ua = matchedPoints1\_norm(ii,1);

va = matchedPoints1\_norm(ii,2);

ub = matchedPoints2\_norm(ii,1);

vb = matchedPoints2\_norm(ii,2);

A\_matrix(ii,:) = [ua\*ub va\*ub ub ua\*vb va\*vb vb ua va 1];

end

%%%%%%%%%%%%%%%%%%% Evaluate the equation using SVD %%%%%%%%%%%%%%%%%%%

%%% step-4.4 : %%%

[~, ~, V] = svd(A\_matrix);

f = V(:,end);

T = reshape(f,[3 3])';

% Converting to rank 2

[U, S, V] = svd(T);

S(3,3) = 0;

T\_matrix = U\*S\*V';

%%%%%%%%%%%%%%%%%%%%%%%% Transformation matrix %%%%%%%%%%%%%%%%%%%%%%%%

%%% step-4.5 : %%%

vecT = Ta\*Tb’\*T\_matrix;

**RESULTS: Using sequence 1**

| FIGURE 1 | FIGURE 2 | FIGURE 3 | FIGURE 4 |
| --- | --- | --- | --- |
| Initial Box - frame 3 |  | **Groundtruth box** | **Previous Image - frame 2** |
|  |  |  |  |
| Initial Box - frame 50 |  | **Groundtruth box** | **Previous Image - frame 49** |
|  |  |  |  |

**There are some cons of Least Square Method :**

1. Sensitive to outliers -> Bad matches, extra points
2. Doesn't allow to get multiple good fits

**Fix : Use RANSAC** along with Least square method to tackle outliers

Method 1 : Crop the image using - the box at the beginning.

Method 2 : Crop the image using - the box at the previous time (t-1).

**Comparison between Method 1 vs Method 2 in Least squares Method :**

| Least Square Method  Total error | Method 1 | Method 2 |
| --- | --- | --- |
| sequence 1 | 0.3789 | 0.5630 |
| sequence 2 | Works only till 8th frame.  Throws error as ->  Index exceeds matrix dimensions. | 1.0511 |

**Analysis :**

Method 2 works well for Sequence 2 , whereas method 1 works better for Sequence 1.

Method 1 reduces the error in sequence 1.

**PART 2 : Use RANSAC framework (Additional Work)**

RANSAC is a very general framework for model fitting in presence of outliers.

Basic steps :

1. Choose a small subset of points uniformly at random
2. Fit a model to that subset
3. Find all remaining points that are closer to the model and reject the rest as outliers
4. Repeat the above steps of a particular number of time and chose the best model

**Filename : proj5\_ransac.m**

%%%%%%%%%%%%%%%%%%%%%%% RANSAC ALGORITHM %%%%%%%%%%%%%%%%%%%%%%%%%

%% The input to the algorithm is:

% Pixel locations in the first img

image1\_points = matchedPoints1.Location;

image1\_points(:,3) = 1; % Add homogeneous coordinate

% Pixel locations in the second img

image2\_points = matchedPoints2.Location;

image2\_points(:,3) = 1;

base\_points = ones(n, 3); % Points we want to match

input\_points = ones(n, 3); % Points to be matched against

best\_model = eye(3);

best\_error = 0;

prev\_consensus = 0; % Number of inliers in the previous best model

%% Main loop

n = 1;

k = 100; % - the number of iterations to run

t = 0.005; % - the threshold for the square distance for a point to be considered as a match

verbose = true;

for ii = 0:k

% Take n random points from the first img and their matches in the

% second image

% Generate random array of size n with unrepeated indices up to

% numMatch

rand\_indices = randperm(numMatch, n);

for j = 1:n

base\_points(j, :) = image1\_points(rand\_indices(j), :);

input\_points(j, :) = image2\_points(rand\_indices(j), :);

end

% Create A matrix using the data

A = makeAmatrix(base\_points, input\_points);

% Solve the equations using SVD

[~, ~, V] = svd(A);

% The affine matrix transformation is the last column of the V matrix

% transposed

maybe\_model = reshape(V(:, end), [3, 3])';

% transformation with all the points - Check how good it is

consensus\_set = 0;

total\_error = 0;

for j = 1:numMatch

% Transform the point in image1 using the model TO Image2

image1PointTrans = maybe\_model \* image1\_points(j, :)';

%Make sure the last coordinate is homogeneus

image1PointTrans = image1PointTrans / image1PointTrans(3);

distError = norm(image2\_points(j, :) - image1PointTrans');

if distError < t

consensus\_set = consensus\_set + 1;

total\_error = total\_error + distError;

end

end

% Save this transformation if it includes more points in the consensus

% or the same number of points but with less error

if consensus\_set >= prev\_consensus

if consensus\_set > prev\_consensus

if verbose

fprintf('Improving the model, points match %d, prev mean error %2.2f, current mean error %2.2f\n', ...

consensus\_set, best\_error/prev\_consensus, total\_error/consensus\_set);

end

best\_model = maybe\_model;

best\_error = total\_error;

prev\_consensus = consensus\_set;

else if total\_error < best\_error

if verbose

fprintf('Improving the model, points match %d, prev mean error %2.2f, current mean error %2.2f\n', ...

consensus\_set, best\_error/prev\_consensus, total\_error/consensus\_set);

end

best\_model = maybe\_model;

best\_error = total\_error;

prev\_consensus = consensus\_set;

end

end

end

end

%% Convert the model into an affine transformation matrix

% Force 3,3 element to be 1

best\_model = best\_model / best\_model(3,3);

% Force bottom two values to be 0, they should already be close to zero but

% they are not, due to numerical errors in the homograph calculation and

% in the previous normalization step

best\_model(3,1:2) = 0;

**RESULTS: Using sequence 1**

| FIGURE 1 | FIGURE 2 | FIGURE 3 | FIGURE 4 |
| --- | --- | --- | --- |
| Initial Box - frame 10 |  | **Groundtruth box** | **Previous Image - frame 9** |
|  |  |  |  |
| Initial Box - frame 50 |  | **Groundtruth box** | **Previous Image - frame 49** |
|  |  |  |  |

**Comparison between Method 1 vs Method 2 in RANSAC Method :**

| RANSAC Method  Total error | Method 1 | Method 2 |
| --- | --- | --- |
| sequence 1 | 0.0800 | 0.3218 |
| sequence 2 | Works only till 3rd frame -> throws error (insufficient match) | 1.0469 |

**Analysis :**

Using RANSAC model, the error is reduced considerably . However for sequence 2 images using method 1 ( Method 1: use the box at the beginning) before cropping image produces error using both RANSAC and Least Square method due to less number of match available.

**Comparison between Least Square Method vs RANSAC :**

| Sequence 1 - images | Least Square Method error | RANSAC Method errror |
| --- | --- | --- |
| Method 1 | 0.3789 | 0.08 |
| Method 2 | 0.563 | 0.3218 |

**Conclusion :**

Thus I was able to construct the translation matrix using the matching correspondences between 2 images. I implemented both Least square method along with RANSAC framework to loop through different iterations to remove some outlier and improve the performance. The improvement in performance is evident from the decrease in error in the model.