Computer Vision: EECE 5639
Project Extra
November 10, 2019
Due on December 10, 2019
Structure from Motion

By, John Nguyen

Abstract:

For the structure from motion experiment, the techniques for estimating three dimensional structures from two-dimensional image sequences is explored. From a dataset of 101 480x512 grayscale images, the objective is to perform 3D reconstruction from the set of 2D frames of a hotel. This is achieved by implementing the factorization method, finding key features from the Harris corner detector, and using the MATLAB KLT point tracker also known as the Kanade-Lucas-Tomasi algorithm.

Experiments and Algorithms:

To begin the experiment, the dataset is 101 images of the same hotel in 480x512 grayscale images. With these images, the first step is to use the Harris corner detection algorithm to extract key features which in this case is corners. As mentioneded in Project 2 and Project 3,

"The first algorithm used, the Harris Corner Detector, extracts corners from the input frame. Corners are useful for finding the correspondences between small sections of images so that the similarities between images can be observed. The basic methodology of corner detection is to find two adjacent strong edges. The Harris corner detector determines the amount of change in intensity in a small window when moving in directions:

E(u,v)=x,yw(x,y)[I(x+u,y+v)-I(x,y)]2 [Eq 1.]

Using the first order approximation of f(x,y), the previous equation can be rewritten as

$$E(u,v) \approx [\ u \quad v\ ] M \begin{bmatrix} u \\ v \end{bmatrix}$$

$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$ [Eq 2.]

The matrix M is calculated from the derivatives at each pixel. The presence of corners can be determined by calculating the eigenvalues 1,2of M: there is a corner when both are large. The corner response can be calculated as the following. R is large when there is a corner while it is negative when there is an edge. By thresholding R, we can get the corner pixels.

R=det M-k(trace M)2 [Eq 3.]

The R value matrix is rotation invariant which is advantageous since it does not need any pre alignment. Equation 3 describes the R value computation where k is set to .04. After getting the vertical and horizontal gradient vectors, the sums of the square individual gradients and the product of the gradient is used to create the R matrix at each respective pixel. A corner is denoted by a high R value as shown in the figure below.

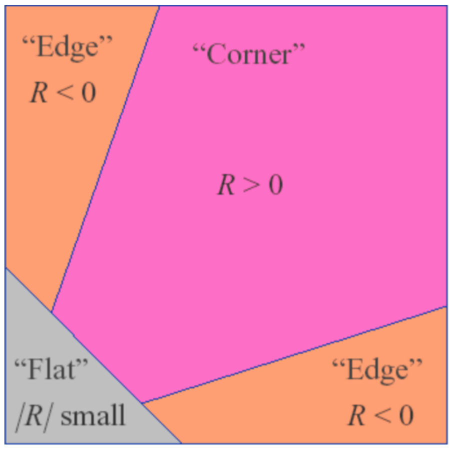

Figure 1 - R value response

The R values chosen are shown to be effective in identifying corners and edges." (Project 2). After applying this algorithm, Figure 2 shows the results on the first frame and Figure 3 shows additional features detected on a variety of other frames.

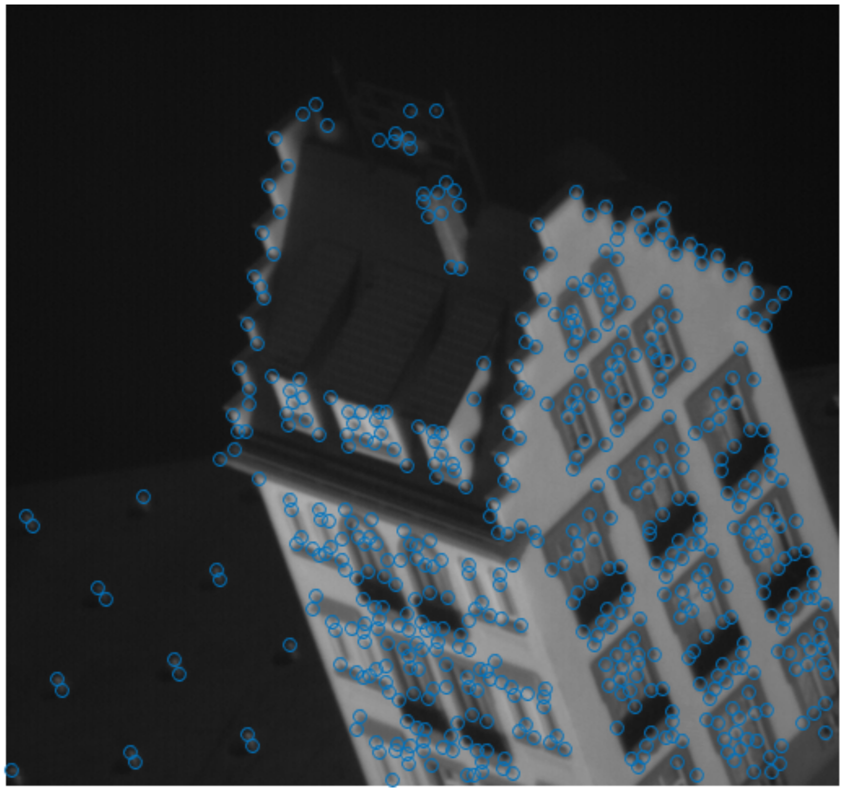
Figure 2 - Features detected in the initial frame

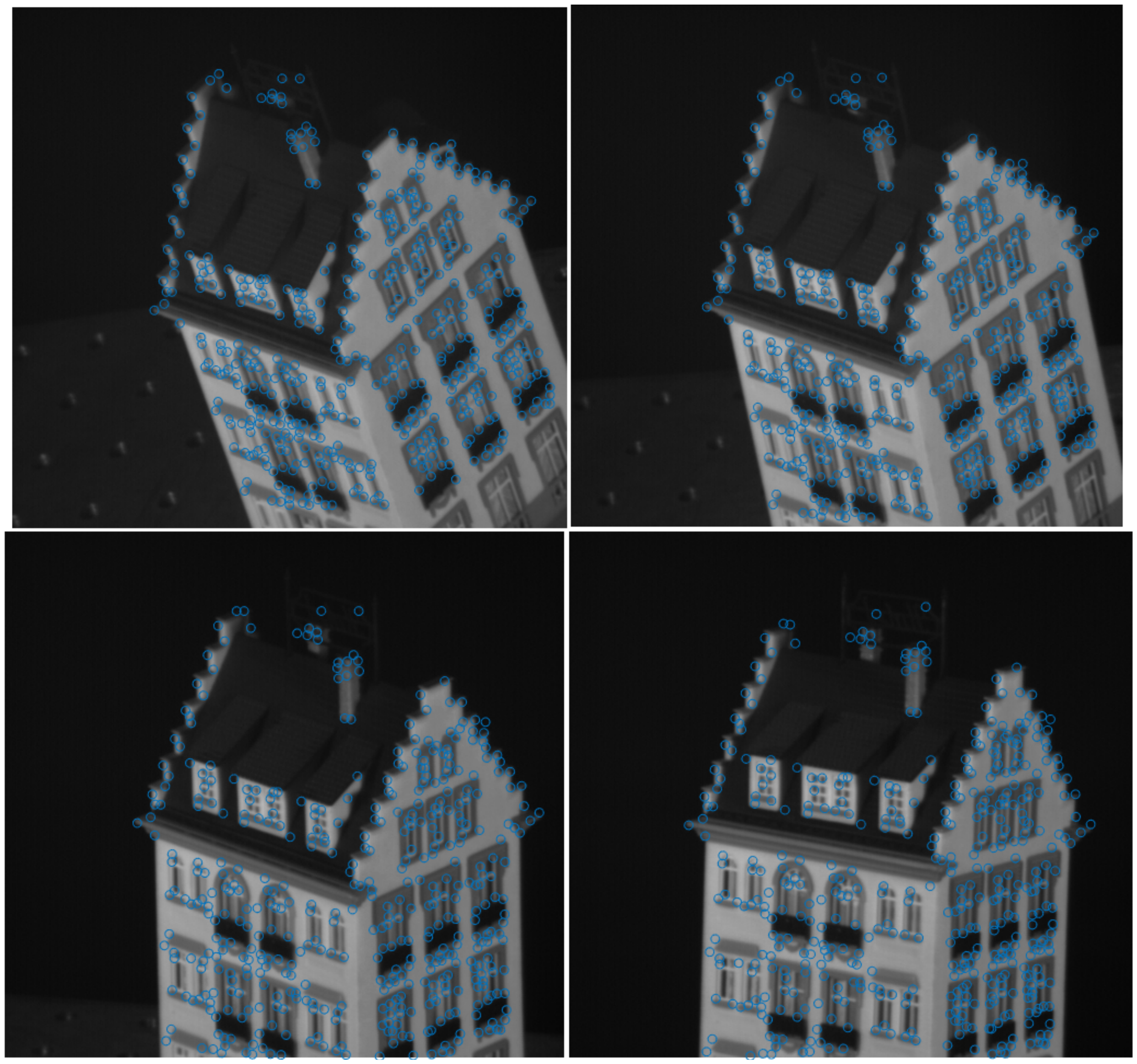
Figure 3: Tracked features in frame 1, 20, 50, 80

Through factorization, a fixed camera is used in tandem with a moving object to track features across frames to find structure and motion of the object. Using object coordinates, which moves with the object, a rotation and translation is applied to express object coordinates in the camera coordinate system and then the weak perspective model is applied. In summary with the motion and structure matrix, the measurement matrix can be computed as shown in the figure below.

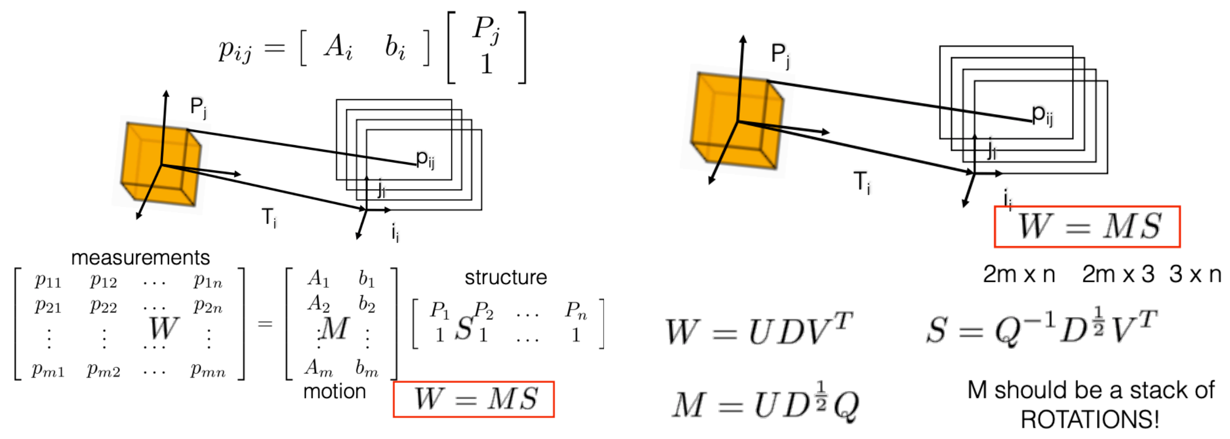
$p_{ij} = \begin{bmatrix} A_i & b_i \end{bmatrix} \begin{bmatrix} P_j \\ 1 \end{bmatrix}$
P
P_j
p_{ij}
T_i
j_i
i_i
measurements
$\begin{bmatrix} p_{11} & p_{12} & \dots & p_{1n} \\ p_{21} & p_{22} & \dots & p_{2n} \\ \vdots & \vdots & W & \vdots \\ p_{m1} & p_{m2} & \dots & p_{mn} \end{bmatrix} = \begin{bmatrix} A_1 & b_1 \\ A_2 & b_2 \\ \vdots M & \vdots \\ A_m & b_m \end{bmatrix} \begin{bmatrix} P_1 & P_2 & \dots & P_n \\ 1 & 1 & \dots & 1 \end{bmatrix}$
structure
S
motion
$W = MS$
$W = UDV^T$
$M = UD^{\frac{1}{2}}Q$
$W = MS$
2m x n 2m x 3 3 x n
$S = Q^{-1}D^{\frac{1}{2}}V^T$
M should be a stack of ROTATIONS!

Figure 4: Moving tracks of features

Using the point tracker from MATLAB, a set of new locations of the points from the input frame is obtained, and from there the measurement matrix is computed from subtracting the mean in each frame. Computed the singular value decomposition of the measurement matrix yields the motion and structure matrix where the Q is computed as an affinity so that MQ is a stack of rotation matrices. The following figure shows the moving features tracks obtained from the measurement matrix.

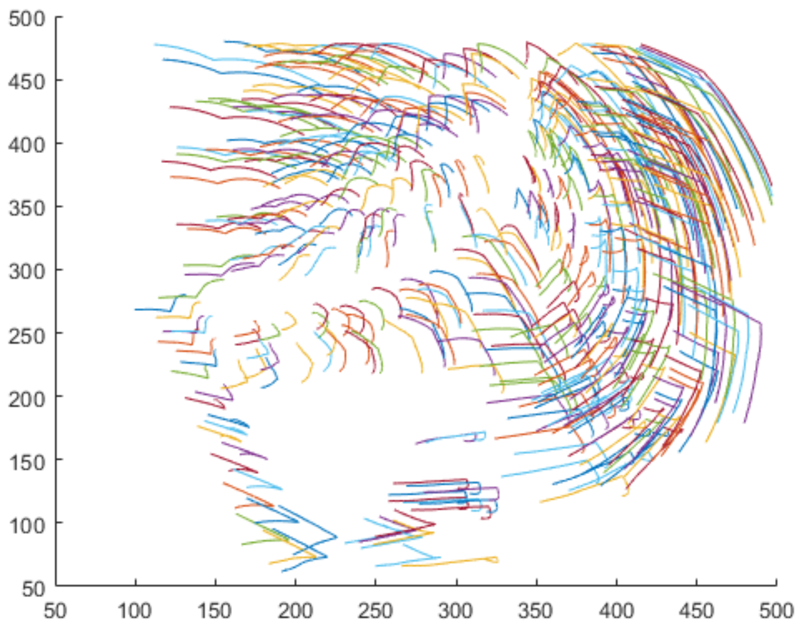

Figure 5: Moving tracks of features

3D reconstruction was attempted by generating a point cloud as shown below.

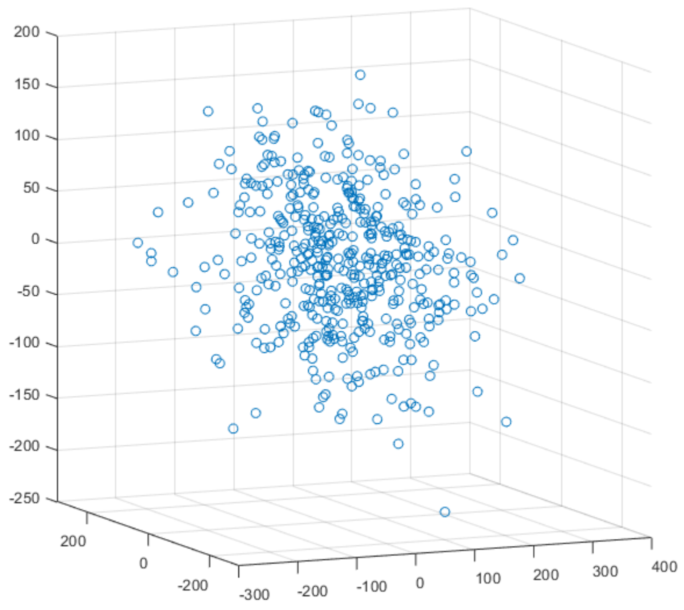

Figure 6: 3D structure reconstruction

The point cloud attempted has sources of error from susceptible noise from motion in addition to possible errors in quality of features detected and selected after thresholding. Given more time to debug and obtain better features, 3D reconstruction is hoped to be achieved.

Conclusion:
In conclusion, for the structure from motion, the techniques for estimating three dimensional structures from two-dimensional image sequences used are the factorization method, finding key features from the Harris corner detector, and using the MATLAB KLT point or Kanade-Lucas-Tomasi algorithm from a dataset of 101 480x512 grayscale images. The final point cloud generated resembles the hotel sequence, but it is assuming that there are sources of error from noise from and quality of features detected. As mentioned earlier, given more time to debug these issues and possible sources of error, 3D reconstruction with more desirable results is hoped to be achieved.

Appendix:

extraproject.m

```
clear all; close all; clc;

%% load images
init = imread("hotel/hotel/hotel.seq0.png");
for i = 1:100
    hotel{i} = imread("hotel/hotel/hotel.seq"+i+".png");
end

%% init tracker
corners = hcd(init);
tracker = vision.PointTracker('MaxBidirectionalError',1);
initialize(tracker,corners,init);

%% get measurement matrix
cornercnt = size(corners,1);
valid = true(cornercnt,1);
for i = 1:100
    [points,validity] = tracker(hotel{i});
    valid = valid & validity;
    pts{i} = points;
end
for i = 1:100
    pts{i} = pts{i}(valid, :);
    pts_avg{i} = pts{i} - mean(pts{i});
end
W = [];
for i = 1:100
    W = [W;pts_avg{i}'];
    WW = [WW;pts{i}'];
end

%% compute Q
[U,D,V] = svd(W);
M = U*sqrt(D); M = M(:,1:3);
S = sqrt(D)*V; S = S(1:3,:);
Q = solveQ(M);
P = inv(Q)*S;
scatter3(P(1,:),P(2,:),P(3,:));
```

hcd.m

```
function corners = hcd(Im1)
I = Im1;
deriv_filt =  .5*[-1, 0, 1];
```

```
w_size  =3;
window = ones(w_size);
frame_num = 1;
wind_sz = 5;
corners = {};
midpoint = floor(wind_sz/2);
for i = 1:frame_num
   Dx = conv2(I, deriv_filt', 'same');
   Dy = conv2(I, deriv_filt, 'same');
   Dx2 = Dx.^2; Dy2 = Dy.^2;
   DxDy = Dx .* Dy;
   Dx2 = conv2(Dx2, window,'same');
   Dy2 = conv2(Dy2, window,'same');
   DxDy = conv2(DxDy, window,'same');

   for x = 1:size(Dx, 1)
      for y = 1:size(Dy, 2)
         c_matrix = [Dx2(x, y), DxDy(x, y);  DxDy(x, y), Dy2(x, y)];
         lambda = eig(c_matrix);
         R_matrix(x,y,i) = (lambda(1)*lambda(2)) - (.04*(lambda(1)+lambda(2))^2);
      end
   end
end

%% Threshold
for i = 1:frame_num
   threshold = max(max(R_matrix(:, :, i)));
   threshold = threshold/2500;
   R_matrix(:, :, i) = R_matrix(:, :, i) .* (R_matrix(:, :, i) > threshold);
end
%% Dialate R matrix and threshold transform the highest value (NMS)
for i = 1:frame_num
  R_dialate = imdilate(R_matrix(:, :, i), ones(7));
  R_matrix(:, :, i) = R_matrix(:, :, i) .* (R_matrix(:, :, i) == R_dialate(:, :));
end

%find the pixels coordinates where corners occur
for i = 1:frame_num
   c = [];
   [c(:,2),c(:,1)] = find(R_matrix(:, :, i) > 0);
   c = c(c(:, 1) < size(I,2) - midpoint,:);
   c = c((c(:, 2) < size(I,1) - midpoint),:);
   c = c(c(:,1) > midpoint,:);
   c = c(c(:,2) > midpoint,:);
   corners = c;
end
```

solveQ.m

```
function Q = solveQ(U)
  H = []; y = [];
  for i = 1:100
      I = U(2*i-1,:);
      J = U(2*i,:);
      II = reshape(I'*I,1,9);
      JJ = reshape(J'*J,1,9);
      IJ = reshape(I'*J,1,9);
      JI = reshape(J'*I,1,9);
      H = [H;II;JJ;IJ;JI];
      y = [y;1;1;0;0];
  end
  x = inv(H'*H)*H'*y;
  R = reshape(x, 3, 3)';
  Q = chol(R)';
end
```

plottracks.m

```
function plottracks(W)
    for i=1:200
        A(i) = logical(mod(i,2));
        B(i) = not(A(i));
    end
    figure; hold on;
    for i=1:448
        plot(W(A,i),W(B,i));
    end
end
```

```
figure, imshow(hotel{20}), hold on, scatter(pts{20}(:,1),pts{20}(:,2)), hold off;
```