GZ720J: Homework 4
Tracking Templates and Control Points

Name - Jiaxin Chen

November 11, 2016

1 The Car Tracker: Template Tracking with Lucas-Kanade (10 pts)

Question 1.1 Warmup (5pts)

1) The matrix A^TA is is approximate to the Hessian matrix.

Linearizing the objective function by first order Taylor expansion:

$$\sum_X \left[I(W(X,p)) + \nabla I(W(X,p)) \frac{\partial W}{\partial p} \Delta p - T(X) \right] \tag{1}$$

And we take the partial derivative of Δp to the expansion equal to 0, we obtain:

$$\begin{aligned} &\sum_X \left[\nabla I(W(X,p)) \frac{\partial W}{\partial p} \Delta p \right]^T \left[\nabla I(W(X,p)) \frac{\partial W}{\partial p} \Delta p \right] \Delta p \\ &= \sum_X \left[\nabla I(W(X,p)) \frac{\partial W}{\partial p} \Delta p \right]^T [T(X) - I(W(X,p))] \end{aligned} \tag{2}$$

which is

$$A^TA\Delta p = A^Tb \tag{3}$$

Therefore we can obtain A^TA, which is approximate to the Hessian matrix.

$$A^TA = \sum_X \left[\nabla I(W(X,p)) \frac{\partial W}{\partial p} \Delta p \right]^T \left[\nabla I(W(X,p)) \frac{\partial W}{\partial p} \Delta p \right] \tag{4}$$

2) Conditions: A^TA must satisfy positive definite and invertible($A^TA \neq 0$), which means all of its eigenvalues $\lambda_i \neq 0$. Besides, A^TA should be well-conditioned if it's large, so the template offset can be calculated reliably.

Question 1.2 Discussion (5pts)

The car tracker works perfectly during the 101 frames. And I consider about 3 scenarios and the corresponding solutions:

1) If the car is sheltered by another cars or objections partially or fully, the tracker may fail. The possible solution is to use robust tracking.

2) If there is a viewpoint change of the tracked car, the tracker may fail. The possible solution is to use SDM algorithm to solve it.

3) If the car moves too fast, the tracker also may fail. The possible solution is multi-scaled tracker.

2 The Pooh Tracker: Component-based Tracking (90 pts)

2.1 Tracking the Pooh with the LK Tracker

Question 2.1.1 Implementation (8 pts)

```
% Create by chenjx65 on 2016-11-1
clear all;
addpaths;

load('data/pooh/rects_frm992.mat');
poohPath = 'data/pooh/testing';

% Initialize and obtain frame
first = 992;
last = 3000;
length = last - first + 1;
rectInit = {rect_lear, rect_leye, rect_nose, rect_rear, rect_reye};
rectUpdate = rectInit;
It = cell(1, length);
for i = 1 : length
    It{i} = imread(fullfile(poohPath, sprintf('image-%04d.jpg', i + first ...
        - 1)));
end

% Open video writer
vidname = 'pooh_lk.avi';
vidout = VideoWriter(vidname);
vidout.FrameRate = 10;
open(vidout);
% Add frames to video
for i = 1 : length - 1
    % Compute displacement for 5 rectangulars by LK
    for j = 1 : 5
        [u, v] = LucasKanade(It{i}, It{i+1}, rectInit{j});
        rectInit{j} = rectInit{j} + [u, v, u, v];
        % Adjust rectangular and update to draw
        rectUpdate{j}(1) = max(0, rectInit{j}(1));
        rectUpdate{j}(2) = max(0, rectInit{j}(2));
        rectUpdate{j}(3) = min(size(It{i+1}, 2), rectInit{j}(3));
        rectUpdate{j}(4) = min(size(It{i+1}, 1), rectInit{j}(4));
    end
    hf = figure(1); clf; hold on;
    imshow(It{i+1});
    for j = 1 : 5
        rect = [rectUpdate{j}(1:2), rectUpdate{j}(3:4)-rectUpdate{j}(1:2)];
        drawRect(rect, 'r', 5);
    end
    frameNum = i + first;
    fprintf('Frame: %i\n', frameNum);
    text(80, 100, ['Frame ', num2str(frameNum)], 'color', 'y', ...
        'fontsize', 100);
    title('Pooh tracker with Lucas-Kanade Tracker');
    hold off;
    % Write and resize a frame to video
    frm = getframe;
    writeVideo(vidout, imresize(frm.cdata, 0.5));
```

```
50 end
51
52 % Close video writer
53 close(vidout);
54 close(1);
55 fprintf('Video saved to %s\n', vidname);
```

Question 2.1.2 Discussions (2 pts)

I cannot track until frame 3000.

The right eye of the Winne lost the track when it's frame 1410. I think the reasons are the accumulation of errors and the change of viewpoint such as zooming or rotating.

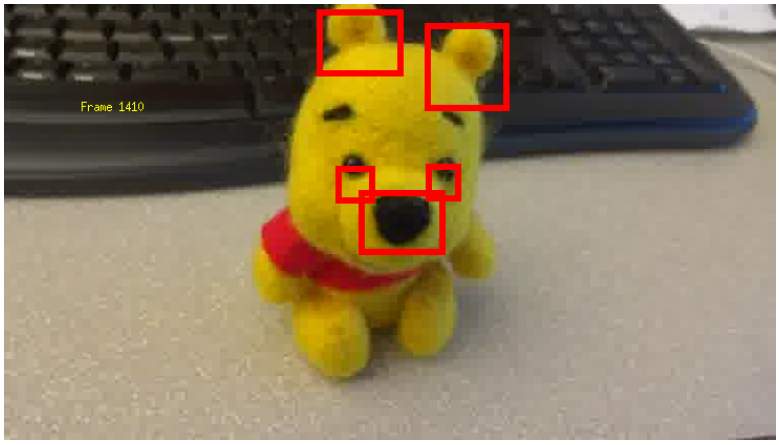

Figure 1: My tracker got lost on frame number 1410

Question 2.1.3 Extra Credit (10 pts)

In order to solve losing the track of right eye problem, I optimize my code as `runTrackPooh_LK_extra.m`. I invoke the `LucasKanade.m` twice to calculate the translation [u1, v1] between the previous frame and current frame, [u2, v2] between current frame and next frame. And I utilize the interpolation to calculate the average translation [u, v]. Therefore, my LK tracker can track the pooh of Winne perfectly until the frame 1726, the right ear of the Winne lost the track firstly as Figure 2. I save the tracking vedio as `pooh_lk_extra.avi`.

Figure 2: My tracker got lost on frame number 1726

```
% Create by chenjx65 on 2016-11-1
clear all;
addpaths;

load('data/pooh/rects_frm992.mat');
poohPath = 'data/pooh/testing';

% Initialize and obtain frame
first = 992;
last = 3000;
alpha = 0.3;
length = last - first + 1;
rectInit = {rect_lear, rect_leye, rect_nose, rect_rear, rect_reye};
rectUpdate = rectInit;

It = cell(1, length);
for i = 1 : length
    It{i} = imread(fullfile(poohPath, sprintf('image-%04d.jpg', i + first ...
        - 1)));
end

% Open video writer
vidname = 'pooh_lk_extra.avi';
vidout = VideoWriter(vidname);
vidout.FrameRate = 10;
open(vidout);
% Add frames to video
for i = 2 : length - 1
    % Compute displacement for 5 rectangulars by LK
    for j = 1 : 5

        % Question 2.1.3 Extra Credit
        %************************************************************
        % Utilize the combination of rectUpdate and rectInit
        [u1, v1] = LucasKanade(It{i-1}, It{i}, rectUpdate{j});
        [u2, v2] = LucasKanade(It{i}, It{i+1}, rectInit{j});

        % Use interpolation to combine the translation of u1, u2 and v1, v2
        u = alpha*u1 + (1-alpha)*u2;
        v = alpha*v1 + (1-alpha)*v2;
        rectInit{j} = rectInit{j} + [u, v, u, v];
        %************************************************************

        % Adjust rectangular and update to draw
        rectUpdate{j}(1) = max(0, rectInit{j}(1));
        rectUpdate{j}(2) = max(0, rectInit{j}(2));
        rectUpdate{j}(3) = min(size(It{i+1}, 2), rectInit{j}(3));
        rectUpdate{j}(4) = min(size(It{i+1}, 1), rectInit{j}(4));
    end
    hf = figure(1); clf; hold on;
    imshow(It{i+1});
    for j = 1 : 5
        rect = [rectUpdate{j}(1:2), rectUpdate{j}(3:4)-rectUpdate{j}(1:2)];
        %rect = [rectInit{j}(1:2), rectInit{j}(3:4)-rectInit{j}(1:2)];
        drawRect(rect, 'r', 5);
    end
    frameNum = i + first;
    fprintf('Frame: %i\n', frameNum);
    text(80, 100, ['Frame ', num2str(frameNum)], 'color', 'y', ...
        'fontsize', 100);
```

```
    title('Pooh tracker with Lucas-Kanade Tracker');
    hold off;
    % Write and resize a frame to video
    frm = getframe;
    writeVideo(vidout, imresize(frm.cdata, 0.5));
end

% Close video writer
close(vidout);
close(1);
fprintf('Video saved to %s\n', vidname);
```

2.2 Tracking the Pooh with Supervised Descent Method (SDM) Tracker

Question 2.2.1 Training step 1: perturbation (10 pts)
Code for `genPerturbedConfigurations.m`

```
% Create by chenjx65 on 2016-11-5
function perturbedConfigurations = ...
    genPerturbedConfigurations(singleFrameAnnotation,meanShape, n, ...
    scalesToPerturb)

% Initialize
perturbedConfigurations = zeros(4, n*5);

% Calculate random translations and scales
translateX = 10 * randn(1, 10*n);
translateY = 10 * randn(1, 10*n);
index = find((sqrt(translateX.^2 + translateY.^2)) ≤ 10);
translateX = translateX(index(1 : n));
translateY = translateY(index(1 : n));
translations = [translateX ; translateY]';
scales = scalesToPerturb(randsample(length(scalesToPerturb), n, true));

for i = 1:n
    % Perturb the meanShape by translation and scaling.
    centerShift = mean(singleFrameAnnotation) - mean(meanShape);
    updateMeanShape = bsxfun(@plus, meanShape, centerShift);
    updateMeanShape = updateMeanShape * scales(i);
    updateMeanShape = bsxfun(@plus, updateMeanShape, translations(i, :));
    scale = findscale(updateMeanShape, singleFrameAnnotation);

    % Generate perturbedConfigurations of x, y locations and scale of SIFT
    perturbedConfigurations(1:2, 5*(i-1)+1 : 5*i) = (updateMeanShape * ...
        scale)';
    perturbedConfigurations(3, 5*(i-1)+1 : 5*i) = [7 4 4 10 10] * scale;
end
end
```

Question 2.2.2 Training steps 2&3: prepare D and F (10 pts)
Code for `genDisplacementMatrix.m`

```
% Create by chenjx65 on 2016-11-5
```

```
function D = genDisplacementMatrix(perturbedConfigurations, annotations, ...
    m, n)

% Initialize
D =[];

for i = 1 : m
    % Obtain annotation and perturbedConfiguration for each frame
    annotation = reshape(annotations(i, 2 :  end), [2, 5])';
    perturbedConfiguration = perturbedConfigurations{i}(1:2, :)';

    % Calculate displacement and update D
    displacement = repmat(annotation, n, 1) - perturbedConfiguration;
    D = [D; reshape(displacement', [10, n])'];
end
end
```

Code for `genFeatureMatrix.m`

```
% Create by chenjx65 on 2016-11-5
function F = genFeatureMatrix(perturbedConfigurations, images, m, n)

% Initialize
F =[];

for i = 1 : m
    % Extract siftFeatures for each image and update F
    siftFeatures = siftwrapper(images{i}, perturbedConfigurations{i});
    F = [F; reshape(siftFeatures, 640, n)'];
end
end
```

Question 2.2.3 Training steps 4&5: linear mapping and update configuration (10 pts)

Code for `learnMappingAndUpdateConfigurations.m`

```
% Create by chenjx65 on 2016-11-5
function [updateConfigurations,W] = ...
    learnMappingAndUpdateConfigurations(perturbedConfigurations, D, F, m, n)

% calculate model weight and updateConfiguratons
W = learnLS(F, D);
updateConfigurations = perturbedConfigurations;

for i = 1 : m
    % Calculate siftFeature and displacement
    perturbedConfiguration = reshape(perturbedConfigurations{i}(1:2, :), ...
        [10, n])';
    siftFeature = F((i-1)*n+1 : i*n, :);
    displacement = siftFeature * W;
    updateConfiguration = perturbedConfiguration + displacement;
    updateConfigurations{i}(1:2, :) = reshape(updateConfiguration', [2, ...
        n*5]);
end
end
```

Question 2.2.4 Sequentially learn multiple mappings (10 pts)

Code for this step is deferred to `SDMtrain.m` in the next section.

The check for the loss function is Table 1. The Euclidean distances between the current estimation and the actual ground truth is decreasing, which means my implementation above is right.

Iteration	1	2	3	4	5
loss	2461.4	5.2245	0.7369	0.3795	0.2809

Question 2.2.5 Integration (15 pts)

Code for `SDMtrain.m`

```
function models = SDMtrain(mean_shape, annotations)
% CV Fall 2014 - Provided Code
% You need to implement the SDM training phase in this function, and
% produce tracking models for Winnie the Pooh
%
% Input:
%   mean_shape:    A provided 5x2 matrix indicating the x and y ...
    coordinates of 5 control points
%   annotations:   A ground truth annotation for training images. Each ...
    row has the format
%                  [frame_num nose_x nose_y left_eye_x left_eye_y ...
    right_eye_x right_eye_y right_ear_x right_ear_y left_ear_x left_ear_y]
% Output:
%   models:        The models that you will use in SDMtrack for tracking
%

% ADD YOUR CODE HERE

% Parameters
poohpath = 'data/pooh/training';
scalesToPerturb = [0.8:0.2:1.2];
n = 100;
mappingNum = 5;
m = size(annotations,1);

% Initialize
images = cell(mappingNum, 1);
perturbedConfigurations = cell(1, m);
models = cell(1, mappingNum);
loss = cell(1, mappingNum);

% Calculate perturbedConfigurations
for i = 1 : m
    images{i} = imread(fullfile(poohpath, sprintf('image-%04d.jpg', ...
        annotations(i,1))));
    singleFrameAnnotation = reshape(annotations(i, 2:end), 2, 5)';
    perturbedConfigurations{i} = ...
        genPerturbedConfigurations(singleFrameAnnotation, mean_shape, n, ...
        scalesToPerturb);
end

% Calculate D, F, models and loss.
for i = 1 : mappingNum
    D = genDisplacementMatrix(perturbedConfigurations, annotations, m, n);
```

```
39      F = genFeatureMatrix(perturbedConfigurations, images, m, n);
40      [perturbedConfigurations,W] = ...
            learnMappingAndUpdateConfigurations(perturbedConfigurations, D, F, ...
            m, n);
41
42      models{i} = W;
43      loss{i} = norm(D(:));
44
45      fprintf('loss: %d\n', loss{i});
46  end
47
48  end
```

Question 2.2.6 Implementation (20 pts)

Code for `SDMtrack.m`

```
1   function SDMtrack(models, mean_shape, start_location, start_frame, ...
        outvidfile)
2   % CV Fall 2014 - Provided Code
3   % You need to implement the SDM test phase in this function, and output a
4   % tracking video for Winnie the Pooh
5   %
6   % Input:
7   %   models:         The model you trained in SDMtrain.m
8   %   mean_shape:     A provided 5x2 matrix indicating the x and y ...
        coordinates of 5 control points
9   %   start_location: A initial location for the first frame to start ...
        tracking. It has the format
10  %                   [nose_x nose_y; left_eye_x left_eye_y; right_eye_x ...
        right_eye_y; right_nose_x right_nose_y; left_nose_x left_nose_y]
11  %   start_frame:    A frame index denoting which frame to start tracking
12  %   outvidfile:     A string indicating the output video file name (eg, ...
        'vidout.avi')
13
14      % Open video for writing
15      vidout = VideoWriter(outvidfile);
16      vidout.FrameRate = 20;
17      open(vidout);
18
19      % ADD YOUR CODE HERE
20
21      begin_shape = start_location;
22      current_shape = start_location;
23
24      mappingNum = length(models);
25
26      for iFrm = (start_frame+1) : 3000
27          % Load testing image
28          I = imread(sprintf('data/pooh/testing/image-%04d.jpg', iFrm));
29
30          % ADD YOUR CODE HERE
31          % Store your initial guess as a 5x2 matrix named begin_shape
32          % (1st column indicates x-coordinate, and 2nd column indicates ...
                y-coordinate).
33          % Store your final guess as a 5x2 matrix named current_shape (in ...
                the same format as begin_shape)
34          perturbedConf = zeros(4, 5);
35          center_shift = mean(current_shape) - mean(mean_shape);
```

```
36
37          mean_shape = bsxfun(@plus, mean_shape, center_shift);
38          scale = findscale(mean_shape, current_shape);
39          mean_shape = scale * mean_shape;
40
41          perturbedConf(1:2, :) = mean_shape';
42          perturbedConf(3, :) = [7 4 4 10 10]* scale;
43
44          for i = 1 : mappingNum
45              f = siftwrapper(I, perturbedConf);
46              displacement = f(:)' * models{i};
47              displacement = reshape(displacement, [2, 5]);
48              perturbedConf(1:2, :) = perturbedConf(1:2, :) + displacement;
49          end
50          begin_shape = mean_shape;
51          current_shape = perturbedConf(1:2, :)';
52
53          % Draw tracked location of parts
54          % Red crosses should track Pooh's nose, eyes and ears
55          figure(1);
56          if ¬exist('hh','var'), hh = imshow(I); hold on;
57          else set(hh,'cdata',I);
58          end
59          if ¬exist('hPtBeg','var'), hPtBeg = plot(begin_shape(:,1), ...
               begin_shape(:,2), 'g+', 'MarkerSize', 15, 'LineWidth', 3);
60          else set(hPtBeg,'xdata',begin_shape(:,1),'ydata',begin_shape(:,2));
61          end
62          if ¬exist('hPtcurrent_shape','var'), hPtcurrent_shape = ...
               plot(current_shape(:,1), current_shape(:,2), 'r+', ...
               'MarkerSize', 25, 'LineWidth', 5);
63          else set(hPtcurrent_shape,'xdata',current_shape(:,1),'ydata',...
64                  current_shape(:,2));
65          end
66          title(['frame ',num2str(iFrm)]);
67          if ¬exist('hFrmNum', 'var'), hFrmNum = text(30, 30, ['Frame: ...
               ',num2str(iFrm)], 'fontsize', 100, 'color', 'r');
68          else set(hFrmNum, 'string', ['Frame: ',num2str(iFrm)]);
69          end
70          %resized so that video will not be too big
71          frm = getframe;
72          writeVideo(vidout, imresize(frm.cdata, 0.5));
73      end
74
75      % close vidobj
76      close(vidout);
77  end
```

Question 2.2.7 Discussions (5 pts) I think the SDM tracker performs better because of 3 reasons:

1) The SDM tracker is more robust due to the meanShape, while the LK tracker assumes the displacement of the components between frames is nearly constant and small theoretically.

2) The SDM tracker uses sift features rather than the previous frame, which can avoid the track get lost.

3) The LK tracker utilizes the pure translation, which cannot handle the change of the viewchange such as zooming and rotating.

1. **LKtracker**

   1) Advantages

      - We don't need training phase and labled data, which is unsupervised and easier to implement.
      - We LK algorithm converges, it can predict more accurate than SDM because SDM use the generic descent direction.

   2) Disadvantages

      - The assumption of LK tracker is based on the nearly constant motion, which is not always true.
      - According to Reference [3], the Hessian matrix might not be positive definite, which is unable to implement LK algorithm.

2. **SDMtracker**

   1) Advantages

      - SDM don't need calculate Hessian matrix and can update displacement during training phase.
      - SDM utilizes SIFT features, so it's more robust to the change of viewpoint.

   2) Disadvantages

      - We need a set of training data to implement SDK, which means the commputation is very expensive during training phase.
      - If the point falls out of the SIFT features, the tracker may become unstable sometimes.

Question 2.2.8 Extra credits (3+3+3 pts) My tracker can overcome the following 3 challenges successfully to track from frame 992 to frame 3000. And I modified my code by following 2 ways:

1) In training phase `SDMtrain.m`, I use multiple scalesToPerturb = [0.8:0.2:1.2] instead of scalesToPerturb = [0.8 1.0 1.2]. So my tracker can handle the large change of the viewpoint.

2) In `genPerturbedConfigurations.m`, I calculate the random translations and only select the translations within 10 pixel translation. So my track can handle the big perturbation of the translation.

And the display of the 3 challenges is shown as Figure 3, Figure 4, Figure 5.

Figure 3: First challenge: frame 1410

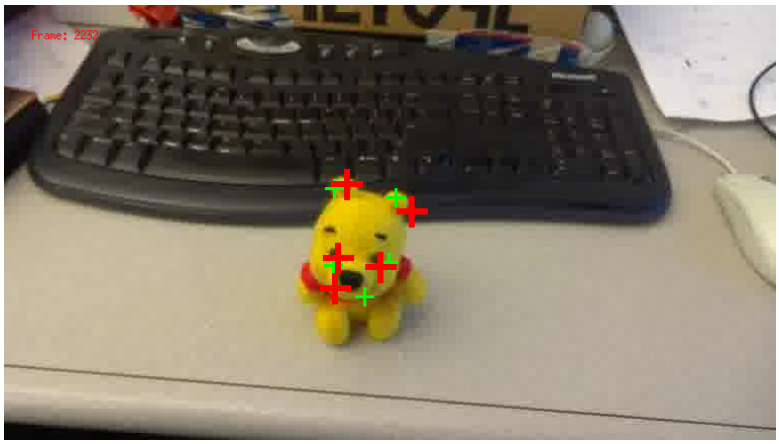

Figure 4: Second challenge: frame 2232

Figure 5: Third challenge: frame 2452