16-720J: Homework 4
Tracking Templates and Control Points

Wenbo Zhao
NetID: zhaowb7 AndrewID: wzhao1
zhaowb7@mail2.sysu.edu.cn

November 14, 2015

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

Question 1.1 Warmup (5pts)

(1) A^TA is used to compute the pseudo-inverse of A in case of A is not square matrix such that the solution to $A\Delta p = b$ is $\Delta p = -(A^TA)^{-1}A^Tb$.

(2) A^TA must be invertible. It happens when all the derivatives are zero, e.g. flat regions, or x and y derivatives are linearly correlated, e.g. lines.

Question 1.2 Discussion (5pts)

Scenarios could fail:

(1) change of illumination: iterate refinement

(2) change of scale: iterate refinement

(3) reflection: use other features

(4) rapid movement: use Pyramids coarse-to-fine resolution

(5) large displacement: iterate refinement

(6) camera rotation: stationary camera

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)
Be sure to include your matlab code using the commented `lsiinputlisting` below ...

```
% ----- 16720J CV HW4: LK-SDM-Tracker -----
% THIS script implements Lucas-Kanade pooh component-based tracker
%
% Author: Wenbo Zhao (wzhao1#andrew.cmu.edu)
% Log:
%
clear all, close all
% Set paths & load data & init
addpaths; % ./lib
addpath ./data/pooh/;
testPooPa = './data/pooh/testing';

load rects_frm992.mat % rect_nose, rect_leye, rect_reye, rect_lear, rect_rear
initRect = [rect_nose; rect_leye; rect_reye; rect_lear; rect_rear];
[nRec, rSize] = size(initRect);
% Read image
imgName = dir(testPooPa); % get all image names
fprintf('images range: %s to %s \n', imgName(3).name, imgName(end).name);

% Open video for writing
vidout = VideoWriter('pooh_lk.avi');
vidout.FrameRate = 20;
open(vidout);

for i = 1:length(imgName)
    if i==1 || i== 2
        continue % skip fisrt two, not image
    end
    if i==3
        img = imread(fullfile(testPooPa,imgName(i).name));
        drawPoo(img, initRect, i-2);
        text(80,100,'Ready?','color','r','fontsize',30); pause(1);
        hf = drawPoo(img, initRect, i-2);
        text(80,160,'GO!','color','g','fontsize',80); pause(.5);
        imgPre  = img;
        rectPre = initRect;
    else
        img  = imread(fullfile(testPooPa,imgName(i).name));
        rect = rectPre;
        for j = 1:nRec
            [u,v] = LucasKanade(imgPre,img,rect(j,:)); % compute the ...
               displacement using LK
            rect(j,:) = rect(j,:) + [u,v,u,v]; % move rectangle
        end
        hf      = drawPoo(img, rect, i-2); % draw frame
        imgPre  = img;
        rectPre = rect;
    end
%resized so that video will not be too big
frm = getframe;
writeVideo(vidout, imresize(frm.cdata, 0.5));
end

% close vidobj
close(vidout);
fprintf('Video saved to %s\n', vidname);
```

Also, be sure to save your output video as `pooh_lk.avi`

Question 2.1.2 Discussions (2 pts)

Can it track until frame 3000? If not, which frame does it lose track? By losing track, we mean that the Intersection over Union (remember Homework 3?) is less than 0.5. You do not need to write code for this. Just try to estimate this visually. Please use ≤ 2 sentences to answer when it fails, and the reason you think why it fails. Show the frame where you think the tracker loses track.

No.

It fails at frame number 2719. The reason is that rectangle exceeds frame boundary.

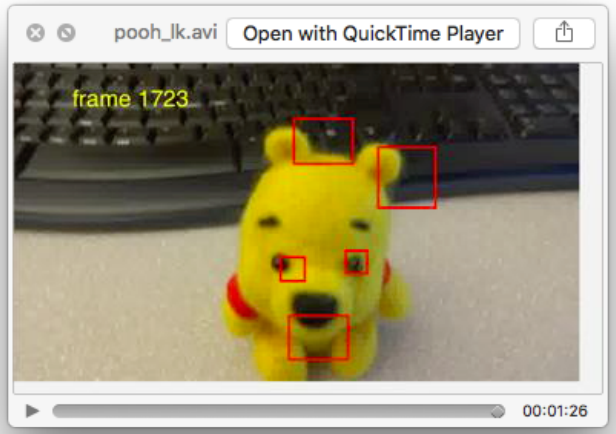

Figure 1: **My tracker got lost on frame number 2719**, the frame count in the picture starts from frame 992, so the lost track frame is $992 + 1727 = 2719$

Question 2.1.3 Extra Credit (10 pts)

Try modifying the LK tracker such that it can track Winnie the Pooh for more frames. Any techniques or modifications are welcomed.

Please be sure to describe what you did or tried to do. Make it easy for the marker to give you some extra marks here :)

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

Question 2.2.1 Training step 1: perturbation (10 pts)

Be sure to include your matlab code using the commented `genPerturbedConfigurations` below ...

```
function perturbedConfigurations = ...
    genPerturbedConfigurations(singleFrameAnnotation, meanShape, n, ...
    scalesToPerturb)
%
% singleFrameAnnotation 5-by-2 ground truth
% meanShape 5-by-2
% n number of perturbation
% perturbedConfigurations 4-by-(n*5)

```

```
ann = reshape(singleFrameAnnotation(2:end), [2, 5]).';
% init meanShape at center and scale to fit annotation
scale_MS_Ann = findscale(ann, meanShape); % N-by-2 -> 1
meanShape    = meanShape./scale_MS_Ann; % scaling
diff_center  = mean(ann, 1) - mean(meanShape, 1);
meanShape    = meanShape + repmat(diff_center, [size(meanShape,1),1]); % ...
    move to center

% perturb
perturbed_MS = cell(n,1);
scaled_MS    = cell(n,1);
tranl_MS     = cell(n,1);
scale_perturbedMS_Ann = cell(n,1);
for i = 1:n
    %
    [scaled_MS{i}, scale_perturbedMS_Ann{i}] = scaleMeanShape(meanShape, ...
        scalesToPerturb);
    %!!!!!! scale 1st or tranl 1st !!!!
    %     perturbed_MS{i} = scaleMeanShape(tranl_MS{i}, scalesToPerturb); ...
        % scale first or translation?
%     scale_perturbedMS_Ann{i} = findscale(ann, scaled_MS{i});
    perturbed_MS{i} = translMeanShape(scaled_MS{i}); % scale first or ...
        translation?

end

% tune SIFT scale
TA_scale = [7 4 4 10 10]; % TA's scale
sift_scale = cell(n,1);
for i = 1:n
    % !!! product or divide !!!
    sift_scale{i} = TA_scale ./ (scale_MS_Ann * ...
        scale_perturbedMS_Ann{i}); % scale TA_scale
end
sift_scale_mat   = cell2mat(sift_scale.'); % 1-by-(n*5)
perturbed_MS_mat = (cell2mat(perturbed_MS)).'; % 2-by-(n*5)
perturbedConfigurations = [perturbed_MS_mat; sift_scale_mat; ...
    zeros(size(sift_scale_mat))];
end

function [scaled_MS, scale] = scaleMeanShape(meanShape, scalesToPerturb)
% perturb meanShape by random scaling
% ----------------------------------
% record center and scale with center in case of large scale
center = mean(meanShape, 1);

nScale = numel(scalesToPerturb);
scale  = scalesToPerturb(randperm(nScale, 1)); % randomly select one scale
meanShape = meanShape./scale;

center_after_scale = mean(meanShape, 1);
diff_center  = center_after_scale - center;
scaled_MS    = meanShape + repmat(diff_center, [size(meanShape,1),1]); %
end

function tranl_MS = translMeanShape(meanShape)
% perturb meanShape by random translation
% ---------------------------------------
% !!!! translation pixels !!!!
tranl    = 5.*rand(1, size(meanShape,2)); % + or -
```

```
tranl_MS = meanShape + repmat(tranl, [size(meanShape,1), 1]);
end
```

Question 2.2.2 Training steps 2&3: prepare D and F (10 pts)

Be sure to include your matlab code using the commented `genDisplacementMatrix.m` below ...

```
function D = genDisplacementMatrix(annotation, perturbedCfg)
% generate displacement matrix between perturbed configurations and ground
% truth
% annotation: ground truth 5-by-2
% perturbedCfg: 4-by-(n*5)
% D: n*10
n    = size(perturbedCfg, 2)./5;
ann  = annotation(2:end);
pCfg = reshape(perturbedCfg(1:2,:), [10, n]).';
D    = repmat(ann, [n, 1]) - pCfg;
end
```

Be sure to include your matlab code using the commented `genFeatureMatrix.m` below ...

```
function F = genFeatureMatrix(annotation, perturbedCfg, nPertCfg, poohpath)
% generate feature matrix
%
% << annotation  : ground truth 1-by-11
% << perturbedCfg: perturbations 4-by-(5nPertCfg)
% << nPertCfg    : number of perturbations
% << poohpath    : train path
%
% >> F: feature matrix nPertCfg-by-640
%
ann = annotation;
pCfg = reshape(perturbedCfg, [4, 5, nPertCfg]);

draw = 0;

I = imread(fullfile(poohpath,'training',sprintf('image-%04d.jpg', ann(1))));
if draw
    imshow(I); hold on;
    % Draw ground truth locations
    now_ann = reshape(ann(2:end), 2, 5)'; plot(now_ann(:, 1), now_ann(:, ...
        2), 'r+', 'MarkerSize', 15, 'LineWidth', 3);
end
F = zeros(nPertCfg, 5*128);
for i = 1:nPertCfg
    % Extract SIFT from I according to perturbations
    d = siftwrapper(I, pCfg(:,:,i));
    if draw
        % Draw SIFT descriptors
        h3 = vl_plotsiftdescriptor(d, pCfg(:,:,i)); set(h3,'color','g');
        pause(0.5);
    end
%     for ii = 1:5
%         d(:,ii) = d(:,ii)./sum(d(:,ii));
%     end
    % !!! normalize !!!
    F(i,:) = d(:).';
```

```
%     F(i,:) = d(:)./sum(d(:));
end

end
```

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

Be sure to include your matlab code using the commented `learnMappingAndUpdateConfigurations.m` below ...

```
function [W, perturbedCfg_out, distPertToAnn] = ...
    learnMappingAndUpdateConfigurations(F, D, perturbedCfg, nPertCfg, ...
    nTrain, annotations)
% learn mapping D = FW by solving least square prob min||FW-D||_F^2 and
% update perturbed configurations using predicted displacement on features
%
W = learnLS(F, D);
D_update = F*W; % 1000-by-10
D_update = reshape(D_update', [2,5*nPertCfg,nTrain]);
distPertToAnn = 0;
% update configurations with new displacement
for i = 1:nTrain
    perturbedCfg_out{i}(1:2,:) = perturbedCfg{i}(1:2,:) + D_update(:,:,i);
    perturbedCfg_out{i}(3:4,:) = perturbedCfg{i}(3:4,:);
    distPertToAnn = distPertToAnn + sum( pdist2( ...
        reshape(perturbedCfg_out{i}(1:2,:), [10,nPertCfg]).', ... % 100-by-10
                    annotations(i,2:end) ) ); % 1-by10
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.

Question 2.2.5 Integration (15 pts)

Be sure to include your matlab code using the commented `SDMtrain.m` below ...

```
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
%

% init path & variables
poohpath = 'data/pooh';
```

```
debug = 0;

ann = annotations;
nTrain = size(ann,1);
% - perturb variables
nPertCfg        = 100; % number of perturbed configurations
% !!!! scales !!!!!
scalesToPerturb = [0.8, 1, 1.2]; % [0.8, 1.0, 1.2]
perturbedCfg    = cell(nTrain,1);

DD = cell(nTrain,1); % displacement
FF = cell(nTrain,1); % feature

notconvergence = 5;
W = cell(notconvergence,1);
ww= 1;
while notconvergence % repeat 5 times
    fprintf('Training iteration %d\n', ww);
    for i = 1:nTrain
        if notconvergence == 5
            perturbedCfg{i} = genPerturbedConfigurations(ann(i,:), ...
                mean_shape, nPertCfg, scalesToPerturb); % 4-by-(n*5)
            DD{i} = genDisplacementMatrix(ann(i,:), perturbedCfg{i}); % ...
                n-by-10
            FF{i} = genFeatureMatrix(ann(i,:), perturbedCfg{i}, nPertCfg, ...
                poohpath);
        else
            DD{i} = genDisplacementMatrix(ann(i,:), perturbedCfg{i}); % ...
                n-by-10
            FF{i} = genFeatureMatrix(ann(i,:), perturbedCfg{i}, nPertCfg, ...
                poohpath);
        end
    end
    D = cell2mat(DD); % mn-by-10
    F = cell2mat(FF); % mn-by-640
    [W{ww}, perturbedCfg, distPertToAnn] = ...
        learnMappingAndUpdateConfigurations(F, D, perturbedCfg, nPertCfg, ...
        nTrain, annotations);
    fprintf('Get mapping matrix. Distance between updated configurations ...
        and ground truth is\n');
    distPertToAnn
    ww = ww+1;
    notconvergence = notconvergence-1;
end
models.map = W;
end
```

Question 2.2.6 Implementation (20 pts)

Be sure to include your matlab code using the commented `SDMtrain.m` below ...

```
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
%

% init path & variables
poohpath = 'data/pooh';
debug = 0;

ann = annotations;
nTrain = size(ann,1);
% - perturb variables
nPertCfg        = 100; % number of perturbed configurations
% !!!! scales !!!!!
scalesToPerturb = [0.8, 1, 1.2]; % [0.8, 1.0, 1.2]
perturbedCfg    = cell(nTrain,1);

DD = cell(nTrain,1); % displacement
FF = cell(nTrain,1); % feature

notconvergence = 5;
W = cell(notconvergence,1);
ww= 1;
while notconvergence % repeat 5 times
    fprintf('Training iteration %d\n', ww);
    for i = 1:nTrain
        if notconvergence == 5
            perturbedCfg{i} = genPerturbedConfigurations(ann(i,:), ...
                mean_shape, nPertCfg, scalesToPerturb); % 4-by-(n*5)
            DD{i} = genDisplacementMatrix(ann(i,:), perturbedCfg{i}); % ...
                n-by-10
            FF{i} = genFeatureMatrix(ann(i,:), perturbedCfg{i}, nPertCfg, ...
                poohpath);
        else
            DD{i} = genDisplacementMatrix(ann(i,:), perturbedCfg{i}); % ...
                n-by-10
            FF{i} = genFeatureMatrix(ann(i,:), perturbedCfg{i}, nPertCfg, ...
                poohpath);
        end
    end
    D = cell2mat(DD); % mn-by-10
    F = cell2mat(FF); % mn-by-640
    [W{ww}, perturbedCfg, distPertToAnn] = ...
        learnMappingAndUpdateConfigurations(F, D, perturbedCfg, nPertCfg, ...
        nTrain, annotations);
    fprintf('Get mapping matrix. Distance between updated configurations ...
        and ground truth is\n');
    distPertToAnn
    ww = ww+1;
    notconvergence = notconvergence-1;
end
models.map = W;
end
```

NOTE: The TAs have provided a master script `runTrackPooh_SDM.m` which runs `SDMtrain.m` and `SDMtrack.m`. Please make sure your tracker runs without errors when it is called from `runTrackPoohSDM.m`. This will be used for grading. Do not modify `runTrackPooh_SDM.m`.

Question 2.2.7 Discussions (5 pts) You tried two different trackers on the same sequence. Which one performs better, and why? List two advantages and disadvantages each for the LK and SDM trackers respectively on this part-based tracking task.

SDM tracker works better than LK tracker.

For SDM tracker:

I advantages:

(1) the convergence rate is quadratic

(2) guaranteed to converge provided that the initial estimate is sufficiently close to the minimum

II disadvantages:

(1) need properly tune parameters to train in different scenarios

(2) lose track for rapid movement

For LK tracker:

I advantages:

(1) fast

(2) accurate time derivatives

II disadvantages:

(1) errors on boundaries

(2) not robust to changes of illumination, rapid movement

Question 2.2.8 Extra credits (3+3+3 pts) Write which challenges if any you have overcome here and include the path to your code for fixing this.

Challenge overcome:

(1) over 1410: done.

(2) over 2232: track for the first few frames: done by add large scales for training.

(3) over 2452: failed.

Question 2.2.9 Extra credits (10 pts) Write about your modifications for dealing with rotation and include a video (and tell us the path here so we find it!). Be sure to let us know the path to your modified code so we can run it too.