December 6, 2017

Vibhavi Peiris

Student Number: 1000597687

CSC420

Assignment 5

Note\*: **All Coding done on Matlab R2016a.**

Question 1 – Tracking:

* 1. Instead of using a list I decided to use a struct with the format seen below.

tracks = struct(previous\_frame,previous\_frame\_val,occurences,occurences\_val,dets,dets\_val,frames\_field,frames\_val);

I loop through the frames find the sim matrix (using the code provided)

Then I have a while loop that stops once there are only 0’s left in the sim matrix. Each iteration the loop find the max value in the sim matrix, returns its k and t value. Then checks if that current detection is in the tracks. I do this by getting an array containing the last added detection for each track. If the current detection is in the track I add it to that specific track(i). When adding it I increase the occurrences field by 1, add the new detection(next\_det) to the end of the detections list. I save the current frame value + 1, and I replace the prev\_det\_val with next\_det (this is the latest detection on this track). The I updated that track(i) with this struct. If the current detection was not found in tracks list then I create a new struct and add it to the track. With the fields (occurunces = 2, dets = [cur\_dets next\_dets] frame = current frame value+1, prev\_det = next\_dets).

Outside of the frame’s loop I use find, to find tracks that have occurrences > 2 and >5 for part 1.2

function track\_objects()

% this is a very simple tracking algo

% for more serious tracking, look-up the papers in the projects pdf

FRAME\_DIR = '../data/frames/';

DET\_DIR = '../data/detections/';

start\_frame = 62;

end\_frame = 71;

%tracks = zeros(0,13); %formart [dets\_cur dets\_next occurences]

%field names

previous\_frame = 'prev\_frame'; previous\_frame\_val = zeros(4,0); %had to transpose so ismember would work

occurences = 'occur'; occurences\_val = {0};

dets = 'dets'; dets\_val = zeros(0,6);

frames\_field = 'frames'; frames\_val = zeros(0,0); % the frames a detection occurs in

tracks = struct(previous\_frame,previous\_frame\_val,occurences,occurences\_val,dets,dets\_val,frames\_field,frames\_val);

for idx = start\_frame:end\_frame

im\_cur = imread(fullfile(FRAME\_DIR, sprintf('%06d.jpg', idx)));

data = load(fullfile(DET\_DIR, sprintf('%06d\_dets.mat', idx)));

dets\_cur = data.dets;

im\_next = imread(fullfile(FRAME\_DIR, sprintf('%06d.jpg', idx+1)));

data = load(fullfile(DET\_DIR, sprintf('%06d\_dets.mat', idx+1)));

dets\_next = data.dets;

% sim has as many rows as dets\_cur and as many columns as dets\_next

% sim(k,t) is similarity between detection k in frame i, and detection

% t in frame j

% sim(k,t)=0 means that k and t should probably not be the same track

sim = compute\_similarity(dets\_cur, dets\_next, im\_cur, im\_next);

%greedy approach

finding = true; %loop until no values in sim are > 0

while finding

[maxValue, idxs] = max(sim(:)); %find current max

[i,j] = ind2sub(size(sim),idxs); % get its i and j

if maxValue == 0

finding = false; %stop loop all sims>0 are found

else

%get previous dets locations from tracks

prev\_dets = [tracks.prev\_frame]'; % transpose so that ismember function could find row

%see if current det is in tracks by comparing with prev\_dets

trans\_dets\_cur = dets\_cur(i,1:4); % get current sims detection box

[Result,LocResult] = ismember(trans\_dets\_cur,prev\_dets,'rows');

if Result %if current det is in tracks, then add to that track

curr\_occurs = tracks(LocResult).occur + 1; %increase number of occurence for current track by 1

curr\_dets = tracks(LocResult).dets;

previous\_frame\_val = dets\_next(j,1:4)'; %set prev\_fram value to det\_next

dets\_val = [curr\_dets;dets\_next(j,1:4)]; %append the next det to the tracks dets list

frames\_val = [tracks(LocResult).frames idx+1]; %save the frames this track was seen in

tracks(LocResult) = struct(previous\_frame,previous\_frame\_val,occurences,curr\_occurs,dets,dets\_val,frames\_field,frames\_val);

%tracks(LocResult,:) = [dets\_cur(i,:) dets\_next(j,:) curr\_occurs+1]; %save the detections, occured twice (current & next)

else

%add current det and next det to a new track

track\_size = size(tracks,2);

curr\_occurs = 2;

previous\_frame\_val = dets\_next(j,1:4)';

dets\_val = [dets\_cur(i,1:4);dets\_next(j,1:4)];

frames\_val = [idx idx+1];

tracks(track\_size+1) = struct(previous\_frame,previous\_frame\_val,occurences,curr\_occurs,dets,dets\_val,frames\_field,frames\_val);

%tracks(track\_size+1) = [dets\_cur(i,:) dets\_next(j,:) 2]; %save the detections, occured twice (current & next)

end

sim(i,j)=0; %set the current sim to 0 so it wont be choosen again

end

end

end;

%only take tracks with occurences > 2

idx = find([tracks.occur]>2);

correct\_tracks = tracks(idx);

%visualize tracks with more than 5

idx = find([tracks.occur]>5);

visualize\_tracks = tracks(idx);

track\_size = size(visualize\_tracks,2);

colors = {'r','g','b', 'y', 'm', 'c', 'w','k'}; %set colors for modified showboxes function

%%loop through each frame and show detections

for idx = start\_frame:end\_frame

im\_cur = imread(fullfile(FRAME\_DIR, sprintf('%06d.jpg', idx)));

boxes = [];

frame\_colors = {};

%get the boxes (loop through each track)

for i = 1:track\_size

track\_dets = visualize\_tracks(i).dets; %get dets for all frams on current track

track\_frames = visualize\_tracks(i).frames;

location = find(track\_frames == idx);

if location %if current track has current frame

boxes = [boxes track\_dets(location,:)];

frame\_colors{size(frame\_colors,2)+1} = colors{i}; %set a color got current box

end

end

figure,showboxes(im\_cur, boxes, frame\_colors) %run modified showbox function

end

end

function sim = compute\_similarity(dets\_cur, dets\_next, im\_cur, im\_next)

n = size(dets\_cur, 1);

m = size(dets\_next, 1);

sim = zeros(n, m);

area\_cur = compute\_area(dets\_cur);

area\_next = compute\_area(dets\_next);

c\_cur = compute\_center(dets\_cur);

c\_next = compute\_center(dets\_next);

im\_cur = double(im\_cur);

im\_next = double(im\_next);

weights = [1,1,2];

for i = 1: n

% compare sizes of boxes

a = area\_cur(i) \* ones(m, 1);

sim(i, :) = sim(i, :) + weights(1) \* (min(area\_next, a) ./ max(area\_next, a))';

% penalize distance (would be good to look-up flow, but it's slow to

% compute for images of this size)

sim(i, :) = sim(i, :) + weights(2) \* exp((-0.5\*sum((repmat(c\_cur(i, :), [size(c\_next, 1), 1]) - c\_next).^2, 2)) / 5^2)';

% compute similarity of patches

box = round(dets\_cur(i, 1:4));

box(1:2) = max([1,1],box(1:2));

box(3:4) = [min(box(3),size(im\_cur, 2)), min(box(4),size(im\_cur, 1))];

im\_i = im\_cur(box(2):box(4),box(1):box(3), :);

im\_i = im\_i / norm(im\_i(:));

for j = 1 : m

d = norm(c\_cur(i, :) - c\_next(j, :));

if d>60 % distance between boxes too big

sim(i,j) = 0;

continue;

end;

box = round(dets\_next(j, 1:4));

box(1:2) = max([1,1],box(1:2));

box(3:4) = [min(box(3),size(im\_cur, 2)), min(box(4),size(im\_cur, 1))];

im\_j = im\_next(box(2):box(4),box(1):box(3), :);

im\_j = double(imresize(uint8(im\_j), [size(im\_i, 1), size(im\_i, 2)]));

im\_j = im\_j / norm(im\_j(:));

c = sum(im\_i(:) .\* im\_j(:));

sim(i,j) = sim(i,j) + weights(3) \* c;

end;

end;

end

function area = compute\_area(dets)

area = (dets(:, 3) - dets(:, 1) + 1).\* (dets(:, 4) - dets(:, 2) + 1);

end

function c = compute\_center(dets)

c = 0.5 \* (dets(:, [1:2]) + dets(:, [3:4]));

end

* 1. The code for this is at the bottom of the code from part A. I also changed the input of showBoxes(im, boxes, colors,out). So that I can send in the colors relative to the boxes. Then in the showboxes function’s main loop I set c = color{i}. the code is below.

function showboxes(im, boxes, colors,out) %changed this added colors input

if nargin > 3

% different settings for producing pdfs

print = true;

%wwidth = 2.25;

%cwidth = 1.25;

cwidth = 1.4;

wwidth = cwidth + 1.1;

imsz = size(im);

% resize so that the image is 300 pixels per inch

% and 1.2 inches tall

scale = 1.2 / (imsz(1)/300);

im = imresize(im, scale, 'method', 'cubic');

%f = fspecial('gaussian', [3 3], 0.5);

%im = imfilter(im, f);

boxes = (boxes-1)\*scale+1;

else

print = false;

cwidth = 2;

end

image(im);

if print

truesize(gcf);

end

axis image;

axis off;

set(gcf, 'Color', 'white');

if ~isempty(boxes)

numfilters = floor(size(boxes, 2)/4);

if print

for i = 1:numfilters

x1 = boxes(:,1+(i-1)\*4);

y1 = boxes(:,2+(i-1)\*4);

x2 = boxes(:,3+(i-1)\*4);

y2 = boxes(:,4+(i-1)\*4);

% remove unused filters

del = find(((x1 == 0) .\* (x2 == 0) .\* (y1 == 0) .\* (y2 == 0)) == 1);

x1(del) = [];

x2(del) = [];

y1(del) = [];

y2(del) = [];

if i == 1

w = wwidth;

else

w = wwidth;

end

line([x1 x1 x2 x2 x1]', [y1 y2 y2 y1 y1]', 'color', c, 'linewidth', w);

end

end

% draw the boxes with the detection window on top (reverse order)

if 1

for i = numfilters:-1:1

%for i = 1:1

x1 = boxes(:,1+(i-1)\*4);

y1 = boxes(:,2+(i-1)\*4);

x2 = boxes(:,3+(i-1)\*4);

y2 = boxes(:,4+(i-1)\*4);

% remove unused filters

del = find(((x1 == 0) .\* (x2 == 0) .\* (y1 == 0) .\* (y2 == 0)) == 1);

x1(del) = [];

x2(del) = [];

y1(del) = [];

y2(del) = [];

if i == 1

c = 'r'; %[160/255 0 0];

s = '-';

% elseif i == 13+1 || i == 14+1

% c = 'c';

% s = '--';

else

c = 'b';

s = '-';

end

c = colors{i}; %ADDED THIS LINE set color to be realtive to current box

line([x1 x1 x2 x2 x1]', [y1 y2 y2 y1 y1]', 'color', c, 'linewidth', cwidth, 'linestyle', s);

end end; end

% save to pdf

if print

% requires export\_fig from http://www.mathworks.com/matlabcentral/fileexchange/23629-exportfig

export\_fig([out]);

end

* 1. You could try using a sliding window to find detections from current window in next frame, and keep ones that are above a certain similarity threshold. You could also try checking similarity at different level using a gaussian pyramid.
  2. First option would be if there is more than one camera and you have the internal and external camera matrix information. You can compute the 3d word coordinates for the center of the 3d bounding box for each player. See how much that center moves through each frame. This will give you a distance per frame, which you can convert to distance per second depending on how long a frame is. If you have only one camera. You could have the user select the 4 corners of the field, and perform homography to get a better representation of distances (like in a2 with the shoe) then you can just find the difference between each center point of detection box between frames, to get distance travelled.

1. Question 2 – Deep Learning
   1. Part 2.1 find derivatives

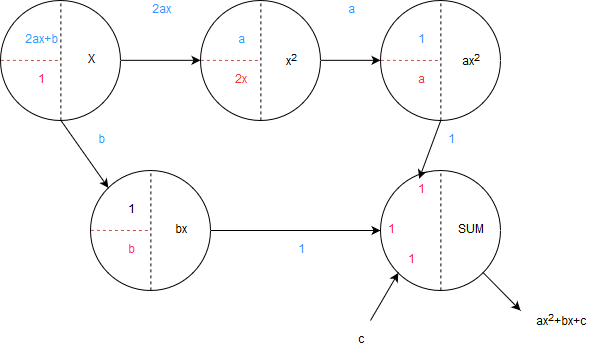


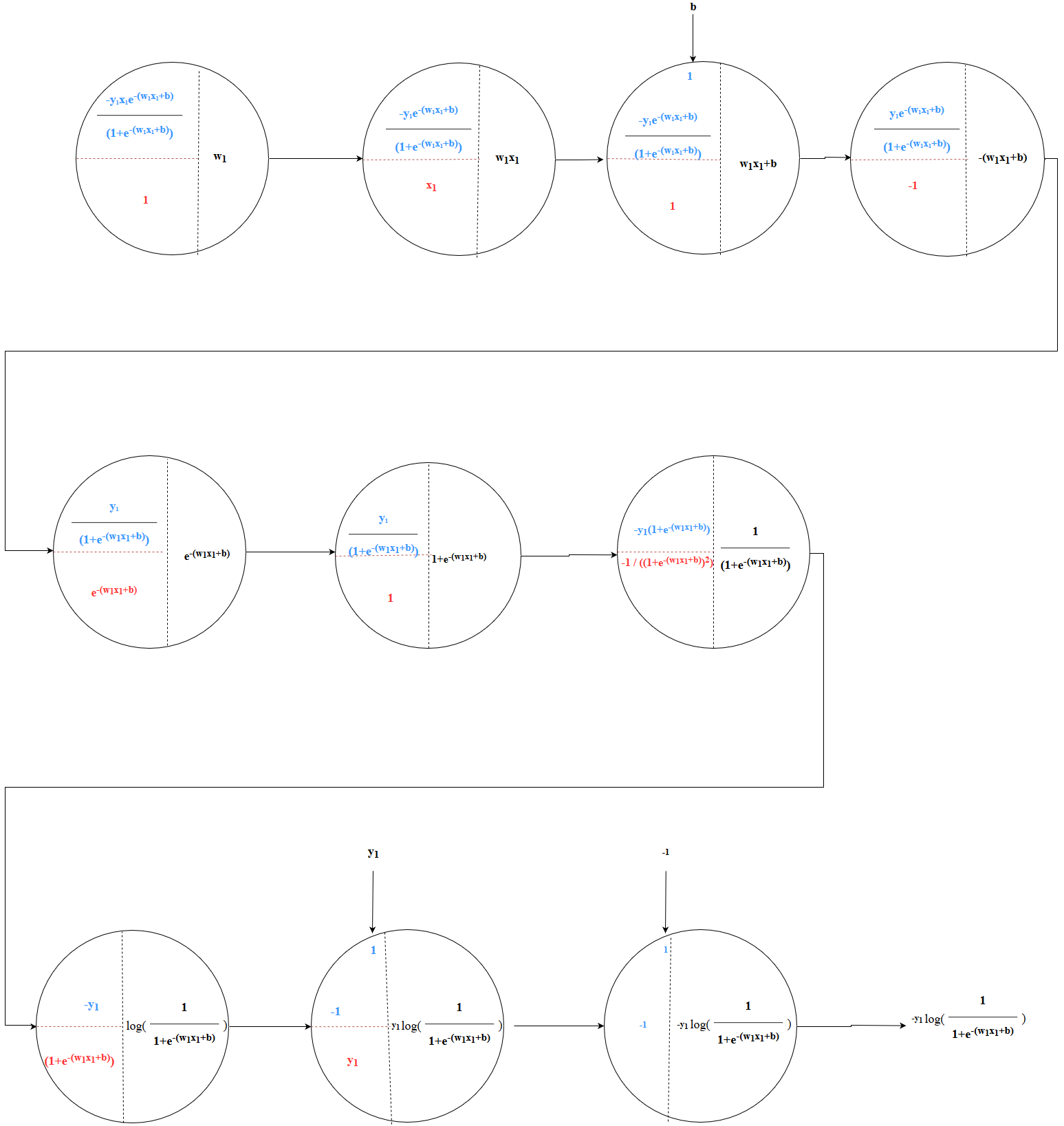






* 1. Question 2.2



* 1. Question 2.3 & 2.4