Alexandru Cojocaru s1575638

Mihail Dunaev s1567045

**Advanced Vision Practical 1**

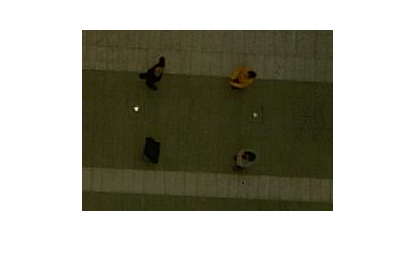
**Report**

The problem of detection and tracking people has been solved in several stages applied on each of the frames:

* Background subtraction
* Application of mathematical morphology to eliminate noise
* Identify resulting groups representing the persons
* If one identified area contains 2 persons separate it using Principal Component Analysis (PCA)
* For the tracking task, we used the following metrics:
  + RGB histogram
  + Distance between the group and the position of each dancer in the previous frame
  + Position prediction using the trajectory from the previous 2 frames

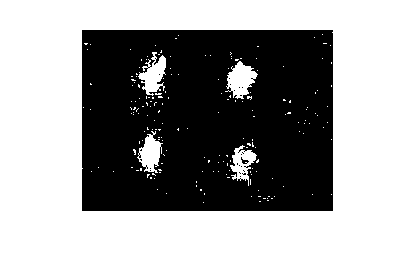
**Background Subtraction**

The first step for detection is represented by the background subtraction.We have used this method because the camera is stationary and the illumination is constant. As there is an object (the laptop) which becomes part of the background and a fifth person, which we should also neglect, we define a section of the photo, which will become our area of interest. From this point, we will work only within this section.



Dance Area

Now we subtract the value of Red, Green and Blue of the pixels in our current frame from the values of the background image. In this way, we construct a binary image where white pixels are associated with subtraction values higher than a threshold (we used a value of 10), indicating a new object in the frame.



**Noise removal**

To eliminate the noise we use mathematical morphology. First, we erode the binary image obtained, gradually eliminating the boundaries of the detected areas. But, while we decrease the size of the detected area, in this way we also enlarge the areas of holes inside them. To solve this, we use dilation to fill the holes inside our objects. In the end, we select our final objects as the ones which have regions of 4-connectedness.

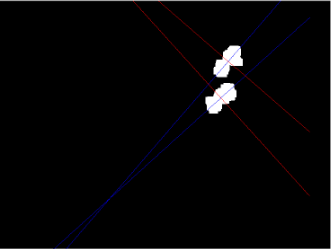
 

Step 1: Erosion Step 2: Dilation

**Detection at collisions**

The remaining problem we need to solve for the detection stage was that in the case when 2 persons were interacting we detected one big area consisting of the 2 merged regions.

We knew when these “collisions” occurred, by counting the number of regions after de noise removal process. In this case we needed to separate the region with the biggest area into 2 separate areas. We have considered many methods, but the one which gave the best results was the Principal Component Analysis. This implies that we find the vector which has the largest variance within the points in the region. Then, we use the vector perpendicular on this which goes through the mean of these points, to divide into 2 regions.

**Results of detection**

Overall, the detection stage obtained really good results as compared to the ground truth.

Using a maximum Euclidian distance of 10 from any ground truth center we correctly detected:

**774** people, resulting in a **92,14%** of correct detections. In the remaining cases, the detection was not far away from the truth center (within a 20 distance from the truth) and it appeared in the situations when the person wearing white clothes was situated inside the white area of the background.

The mean distance between the ground truth and estimated centers within the 10 pixels**: 5.59 pixels**

**False detected = 0**

 ****

 ****

**Tracking**

The tracking process refers to the task of correctly predicting the pairing between a detected region and the corresponding person. To accomplish this, we combined three different metrics:

To begin with, we used the rbg histogram to obtain a distribution of colors. We considered the RGB color space to calculate the discretization of the colors in the image into a number of bins by counting the number of image pixels in each bin. We compute the rbg histogram of the detected regions and compare it to the histogram of the persons in the first frame. By comparing the 2 histograms we obtain a solution for the problem of recognizing a person of unknown position. On the other hand, it focuses only on the proportion of colors, so we might obtain identical histograms for different objects which share the same colors. To compare the 2 histograms we used the Bhattacharyya distance. This method measures the similarity of 2 discrete distributions, thus, obtaining a relative closeness of the 2 color distributions.

We also considered other methods to measure the correlation between 2 histograms, but based on the results obtained we used the Bhattacharyya distance.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| bhattacharyya | | | | |
| player | p1 | p2 | p3 | p4 |
| p1 | 3.94E-008 | 0.3671 | 0.1956 | 0.3539 |
| p2 | x | 0 | 0.335 | 0.2914 |
| p3 | x | x | 1.49E-008 | 0.2965 |
| p4 | x | x | x | 1.49E-008 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| chisquare | | | | |
| player | p1 | p2 | p3 | p4 |
| p1 | 0 | 0.3938 | 0.1408 | 0.3508 |
| p2 | x | 0 | 0.3171 | 0.243 |
| p3 | x | x | 0 | 0.2341 |
| p4 | x | x | x | 0 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| euclidian | | | | |
| player | p1 | p2 | p3 | p4 |
| p1 | 0 | 0.0777 | 0.0513 | 0.0671 |
| p2 | x | 0 | 0.0586 | 0.0565 |
| p3 | x | x | 0 | 0.0467 |
| p4 | x | x | x | 0 |

But because as mentioned the histogram evaluation only considers the color content, the tracking will fail in most cases when trying to distinguish between the two persons wearing black clothes. Another example when this method does not work properly is due to the fact that when considering the detected area of a person it also contains some small areas of the background, so for instance when a person is situated around the yellow part of the background will get a good matching with the person wearing yellow clothes, which is a false result.

This is why we needed to consider additional metrics.

We compute the **distance between a detected person and the positions of each persons in the previous frame**. We can do this, because at each frame, the entire tracking process leads to a result consisting of the positions of each identified labeled person. These results are stored in a vector, so that at frame i, we compute the distance between the detected region and the correctly identified position of every dancer at frame i – 1.

This resulting distance is normalized () and to use it as a probability of matching a region with a label of a person we use P = 1 - .

Using this distance gives us a good indication that the region closest to the position of dancer j in the previous frame should be matched with dancer j. However, this is not the case when there are larger gaps between frames. In this situation, a dancer A might change his position from to a distant position , while other dancer B would make such a move that he would get closer to . In this way, the distance metric would indicate that because the dancer B new position is closer to dancer A previous position, giving a erroneous pairing.

Furthermore, when the dancers interact their positions are close to each other and interchangeable, which in many cases will lead to tracking errors.

To solve these problems we used a **prediction** of the future position of a dancer.

For every dancer, we look at the previous two points on his trajectory. To estimate the future point of this dancer, we fit a polynomial of degree 1 using the two points. In this way we obtain the equation of the line which goes through the previous 2 positions. So, we assume that in the next frame the dancer will maintain this trajectory.

Therefore, we compute the trajectory equation for each of the 4 dancers. For each lines we measure the distance from the centers of every detected region to the line. The closer a center is to that line, the higher probability that it corresponds to the “owner” of that trajectory.

So, the probability is determined by first normalizing the distance and subtracting it from 1 (1 - ).

At this point, we can finally combine these three types of probabilities.

But, because the predictions about the future position give us a more accurate estimation we give it a bigger importance:

⋅

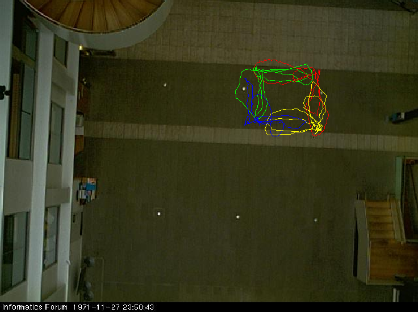
We considered different degrees of the polynomial which defines the trajectories and different degrees of the importance of the predicted probability and we obtained the best results for a 1-degree polynomial and using the square of the prediction probability.

We also need to mention that when computing the equation of the line going through the previous 2 points we might obtain one of the coefficients infinite, when the x coordinates of the 2 points are equal. In this rare cases we do not use the prediction anymore and the decision is based only on histograms and distance.

**Results of tracking**

The results of the tracking process show that for every dancer we produce a single connected trajectory and all the pairings are correct.

We will now show the trajectories of all dancers obtained by our system, compared to the ground truth trajectories (overlaid on the background image).

*Obtained trajectories Ground truth trajectories*

Overall, the prediction and tracking work really well, as seen from the success rates, 94% for prediction and 100% for tracking.

**Code:**

% AV Assignment 1

height = 480; % vertical

width = 640; % horizontal

depth = 3; % RGB

% background image

bck = imread('DATA1/bgframe.jpg');

% positions of dancers in each frame

load('DATA1/positions1.mat');

first\_frame = true;

num\_frames = 210;

count\_frames = 0;

num\_dancers = 4;

unknown\_dancer = 5;

dmax = 100;

area\_threshold = 1500;

bd\_threshold = 0.25;

my\_positions = zeros(4, num\_frames, 2);

n = 2;

% different color for each dancer

colors = ['r' 'y' 'g' 'b' 'w'];

% image difference threshold

eps = 10;

% dance area

da\_sw = 320;

da\_dw = 250;

da\_sh = 60;

da\_dh = 180;

dance\_bck = bck(da\_sh:(da\_sh+da\_dh), da\_sw:(da\_sw + da\_dw), :);

% read all the images

for i = 1:num\_frames

fn = sprintf('DATA1/frame%d.jpg',i+109);

img = imread(fn);

% focus only on the dance area XXX

dance\_img = img(da\_sh:(da\_sh+da\_dh), da\_sw:(da\_sw + da\_dw), :);

% compute the diff, take only values greater than eps

diff\_img = imabsdiff(dance\_img, dance\_bck);

gray\_img = (diff\_img(:,:,1) > eps) | (diff\_img(:,:,2) > eps) | ...

(diff\_img(:,:,3) > eps);

% morph - get rid of the noise

gray\_img = bwmorph(gray\_img,'erode',2);

gray\_img = bwmorph(gray\_img,'dilate',6);

% label each area (4 connectivity)

labeled = bwlabel(gray\_img,4);

% get stats for each area; stats = {Area, Centroid, BoundingBox}

stats = regionprops(labeled,'basic');

% sort stats after area; stats(is)

[~,is] = sort([stats.Area],'descend');

% blob images

imshow(img);

hold on;

num\_blobs = min(num\_dancers,size(stats,1));

num\_combined = num\_dancers - num\_blobs;

pos = zeros(num\_dancers, 3);

probs = zeros(num\_dancers, num\_dancers);

bds = zeros(num\_dancers, num\_dancers);

blob\_index = 1;

% look at all the blobs

for j = 1:num\_blobs

% do some precomputation on blob

% blob center (height, width)

ch = stats(is(j)).Centroid(2);

cw = stats(is(j)).Centroid(1);

% blob leftmost position

csh = stats(is(j)).BoundingBox(2) - 0.5;

csw = stats(is(j)).BoundingBox(1) - 0.5;

% blob size

cimgh = stats(is(j)).BoundingBox(4);

cimgw = stats(is(j)).BoundingBox(3);

% count how many pixels in blob - this is actually area

num\_elems = stats(is(j)).Area;

% check if multiple dancers

if num\_combined > 0

num\_combined = num\_combined - 1;

% get xs & ys

k = 1;

xs = zeros(num\_elems,1);

ys = zeros(num\_elems,1);

for h = csh:(csh+cimgh)

for w = csw:(csw+cimgw)

if gray\_img(h,w) > 0

xs(k) = w;

ys(k) = h;

k = k + 1;

end

end

end

% perform PCA, code taken from bytefish

X = [xs ys];

miu = mean(X);

Xm = bsxfun(@minus, X, miu);

C = cov(Xm);

[V, D] = eig(C);

% sort eigen

[D, i2] = sort(diag(D),'descend');

V = V(:,i2);

% compute second line

a = 0;

b = 0;

if V(1,2) ~= 0

a = V(2,2) / V(1,2);

b = miu(2) - (a \* miu(1));

end

% separate dancers after second line; d1 = up, d2 = down;

xs\_d1 = [];

ys\_d1 = [];

xs\_d2 = [];

ys\_d2 = [];

for k = 1:num\_elems

if V(1,2) == 0

if xs(k) < miu(1)

xs\_d2 = [xs\_d2 xs(k)];

ys\_d2 = [ys\_d2 ys(k)];

else

xs\_d1 = [xs\_d1 xs(k)];

ys\_d1 = [ys\_d1 ys(k)];

end

else

line\_val = xs(k) \* a + b;

if ys(k) < line\_val

xs\_d2 = [xs\_d2 xs(k)];

ys\_d2 = [ys\_d2 ys(k)];

else

xs\_d1 = [xs\_d1 xs(k)];

ys\_d1 = [ys\_d1 ys(k)];

end

end

end

xm\_d1 = mean(xs\_d1);

ym\_d1 = mean(ys\_d1);

pos(blob\_index,:) = [(xm\_d1+da\_sw) (ym\_d1+da\_sh) size(xs\_d1,2)];

% compute histogram and bhattacharyya distance for first blob

% code take from av repo

num\_elems = size(xs\_d1,2);

histr = zeros(1, num\_elems);

histg = zeros(1, num\_elems);

histb = zeros(1, num\_elems);

% bins

edges = zeros(256,1);

for k = 1 : 256;

edges(k) = k-1;

end

for k = 1:num\_elems

h = ys\_d1(k);

w = xs\_d1(k);

histr(k) = dance\_img(h,w,1);

histg(k) = dance\_img(h,w,2);

histb(k) = dance\_img(h,w,3);

end

% one normalised vector

histr = histc(histr,edges)';

histr = histr / num\_elems;

histg = histc(histg,edges)';

histg = histg / num\_elems;

histb = histc(histb,edges)';

histb = histb / num\_elems;

histv = [histr' histg' histb'];

histv = histv / 3;

% save bd

for k = 1:num\_dancers

bd = bhattacharyya(histv, hists{k});

bds(blob\_index,k) = 1 - bd;

end

% do the same for other blob

blob\_index = blob\_index + 1;

xm\_d2 = mean(xs\_d2);

ym\_d2 = mean(ys\_d2);

pos(blob\_index,:) = [(xm\_d2+da\_sw) (ym\_d2+da\_sh) size(xs\_d2,2)];

num\_elems = size(xs\_d2,2);

histr = zeros(1, num\_elems);

histg = zeros(1, num\_elems);

histb = zeros(1, num\_elems);

for k = 1:num\_elems

h = ys\_d2(k);

w = xs\_d2(k);

histr(k) = dance\_img(h,w,1);

histg(k) = dance\_img(h,w,2);

histb(k) = dance\_img(h,w,3);

end

% one normalised vector

histr = histc(histr,edges)';

histr = histr / num\_elems;

histg = histc(histg,edges)';

histg = histg / num\_elems;

histb = histc(histb,edges)';

histb = histb / num\_elems;

histv = [histr' histg' histb'];

histv = histv / 3;

% save bd

for k = 1:num\_dancers

bd = bhattacharyya(histv, hists{k});

bds(blob\_index,k) = 1 - bd;

end

else

% save position

pos(blob\_index,:) = [(cw+da\_sw) (ch+da\_sh) stats(is(j)).Area];

% compute hist & save bd

histr = zeros(1, num\_elems);

histg = zeros(1, num\_elems);

histb = zeros(1, num\_elems);

% bins

edges = zeros(256,1);

for k = 1 : 256;

edges(k) = k-1;

end

k = 1;

for h = csh:(csh+cimgh)

for w = csw:(csw+cimgw)

if gray\_img(h,w) > 0

histr(k) = dance\_img(h,w,1);

histg(k) = dance\_img(h,w,2);

histb(k) = dance\_img(h,w,3);

k = k + 1;

end

end

end

% one normalised vector

histr = histc(histr,edges)';

histr = histr / num\_elems;

histg = histc(histg,edges)';

histg = histg / num\_elems;

histb = histc(histb,edges)';

histb = histb / num\_elems;

histv = [histr' histg' histb'];

histv = histv / 3;

% save hist if first frame

if first\_frame == true

dancer\_index = unknown\_dancer;

if cw > (450 - da\_sw)

if ch > (150 - da\_sh)

dancer\_index = 2;

else

dancer\_index = 1;

end

else

if ch > (150 - da\_sh)

dancer\_index = 4;

else

dancer\_index = 3;

end

end

hists{dancer\_index} = histv;

probs(blob\_index,dancer\_index) = 1;

else

for k = 1:num\_dancers

bd = bhattacharyya(histv, hists{k});

bds(blob\_index,k) = 1 - bd;

end

end

end

blob\_index = blob\_index + 1;

end

if first\_frame == false

% add rgb hist prob

probs = bds;

% add everything else

for j = 1:num\_dancers

% add pos prob; j = blob, k = dancer's prev position

max\_ppoz = 0;

for k = 1:num\_dancers

dw = pos(j,1) - my\_positions(k,i-1,1);

dh = pos(j,2) - my\_positions(k,i-1,2);

dpoz = sqrt(dw\*dw + dh\*dh);

ppoz = 1 - (dpoz/dmax);

probs(j,k) = probs(j,k) \* ppoz;

end

% add pred prob, j = dancer, k = blob

if i > 2

xs = [];

ys = [];

rk = 1;

for k = 1:2

px = my\_positions(j,i - 3 + k, 1);

py = my\_positions(j,i - 3 + k, 2);

% check for duplicated xs

duplicated = false;

duplicated2 = false;

for k2 = 1:size(xs,2)

if px == xs(k2)

duplicated = true;

break;

end

end

if duplicated == true

continue;

end;

xs(rk) = px;

ys(rk) = py;

rk = rk + 1;

end

% interpolate - using polynomial

if size(xs,2) == n

coefs = polyfit(xs, ys, n-1);

% sanity checks

nan\_flag = false;

for k = 1:size(coefs,2)

if isinf(coefs(k))

nan\_flag = true;

break;

end

if isnan(coefs(k))

nan\_flag = true;

break;

end

end

if nan\_flag == false

% compute distances, first find out max

errs = zeros(1,num\_dancers);

max\_err = -1;

for k = 1:num\_dancers

cx = pos(k,1);

cy = pos(k,2);

err = abs(polyval(coefs,cx) - cy)/sqrt(coefs(1)\*coefs(1) + 1);

errs(k) = err;

if err > max\_err

max\_err = err;

end

end

for k = 1:num\_dancers

probs(k,j) = probs(k,j) \* power(1 - (errs(k)/(max\_err+0.5)),2);

end

end

end

end

end

end

% compute most probable dancer

dancers = zeros(1,num\_dancers);

for j = 1:num\_dancers

maxp = -1;

ki = 0;

for k = 1:num\_dancers

if probs(k,j) > maxp

dancers(k) = j;

maxp = probs(k,j);

ki = k;

end

end

for k = 1:num\_dancers

probs(ki,k) = -1;

end

end

% save dancers

for j = 1:num\_dancers

my\_positions(dancers(j), i, :) = [pos(j,1) pos(j,2)];

end

% draw circles & centers

for j = 1:num\_dancers

radius = sqrt(pos(j,3)/pi);

rectangle('Position', [pos(j,1) - radius, pos(j,2) - radius, ...

2\*radius, 2\*radius],'EdgeColor',colors(dancers(j)), ...

'Curvature',[1,1]);

plot(pos(j,1),pos(j,2),colors(dancers(j)));

end

if first\_frame == true

first\_frame = false;

end

% simulate 9 frames/s = 0.11 s

pause(0.11);

hold off;

count\_frames = count\_frames + 1;

end

% compute difference between ground truth and detected positions

num\_detected = 0;

mean\_distance = 0;

for i = 1:num\_frames

for dancer = 1:num\_dancers

dx = abs(positions(dancer,i,1) - my\_positions(dancer,i,1));

dy = abs(positions(dancer,i,2) - my\_positions(dancer,i,2));

d = sqrt(dx\*dx + dy\*dy);

if d <= 10

num\_detected = num\_detected + 1;

mean\_distance = mean\_distance + d;

end

end

end

% display num detected, percentage and mean

mean\_distance = mean\_distance / num\_detected;

disp(num\_detected);

disp((num\_detected \* 100) / (num\_frames \* 4));

disp(mean\_distance);

% draw computed trajectories on background

hold off;

imshow(bck);

hold on;

for dancer = 1:num\_dancers

plot(my\_positions(dancer,:,1),my\_positions(dancer,:,2),colors(dancer));

end

k = waitforbuttonpress;

% draw ground truth trajectories

for dancer = 1:num\_dancers

plot(positions(dancer,:,1),positions(dancer,:,2),colors(dancer));

end