**Q1**

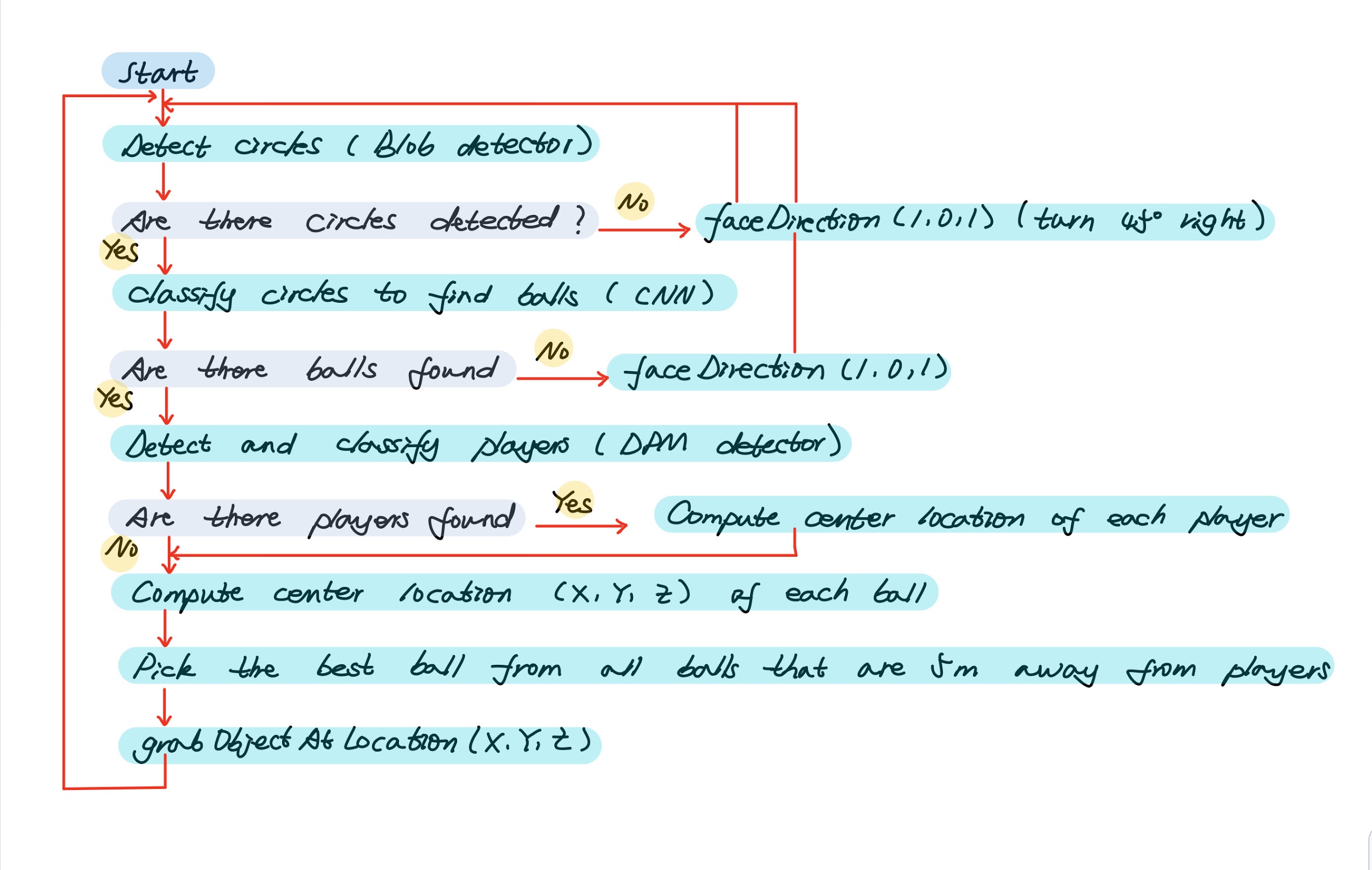
**a):**

**Brain storm**

* Input images shot by stereo cameras.
* Need to find a way to detect circles.
* Need a dataset of images of tennis balls and train a classifier to decide whether a patch contain tennis ball.
* Need a dataset of images of tennis players and train a classifier to decide whether a patch contain tennis player.
* Need to compute depth and 3D location.
* Need to analyze the optimal decision on where to proceed.
* Need to avoid player during matching.

**b):**

**Flow chart**

* For simplicity, I did not consider dropping off balls.
* Assume (X, Y, Z) is in camera’s coordinates.
* Since the robot should work during match. Then there might be balls to collect even if the robot has collected all balls already on the court. Then I decided that the robot will not stop unless someone manually turned it off.

**c):**

**Pseudo code**

% start

While not turned off

Get image\_left

Get image\_right

% return a list of patches of circles detected

Circles = blobDetector(image\_left)

If length(circles) == 0

faceDirection(1, 0, 1)

Continue

End if

% return a sublist of circles that are classified to be balls

Balls = CNN(circles)

If length(balls) == 0

faceDirection(1, 0, 1)

Continue

End if

% algorithm implemented in q2

% given patches, return a list of center location correspond to each patch

Ball\_locations = q2(balls)

%return a list of patches of players

Players = DPMPersonDetector(image\_left)

% compute location of players and exclude balls that are too close to some player

If length(players) > 0

Player\_locations = q2(players)

For player\_location in player\_locations

For ball\_location in ball\_locations

If norm(player\_location, ball\_location) < 5

ball\_locations.remove(ball\_location)

End if

End for

End for

End if

% pick the closest ball to collect

If length(ball\_locations) > 0

(X, Y, Z) = minNorm(ball\_locations)

grabObjectAtLocation(X, Y, Z)

End if

End while

**Q2**

**a):**

**Compute depth**

**Q2\_a.m:**

globals;

imnames = {'004945', '004964', '005002'};

for i = 1:length(imnames)

imname = imnames{i};

% compute depth

camera\_parameters = getData(imname, 'test', 'calib');

f = camera\_parameters.f;

T = camera\_parameters.baseline;

disparity\_struct = getData(imname, 'test', 'disp');

disparity = disparity\_struct.disparity;

depth = f\*T./(disparity);

% save depth

depth\_filename = fullfile(RESULTS\_DIR, strcat(imname, '\_depth.mat'));

save(depth\_filename, 'depth');

% plot depth

fig = figure('position', [100, 100,size(disparity,2)\*0.7, size(disparity,1)\*0.7]);

subplot('position', [0,0,1,1]);

imagesc(depth, [0,256]);

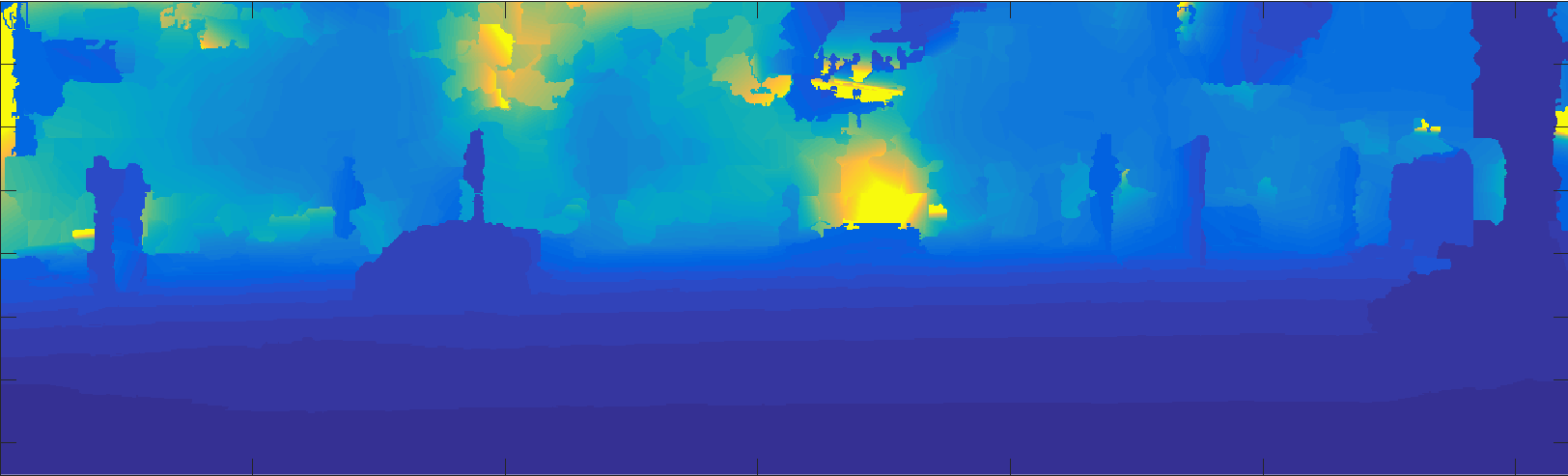
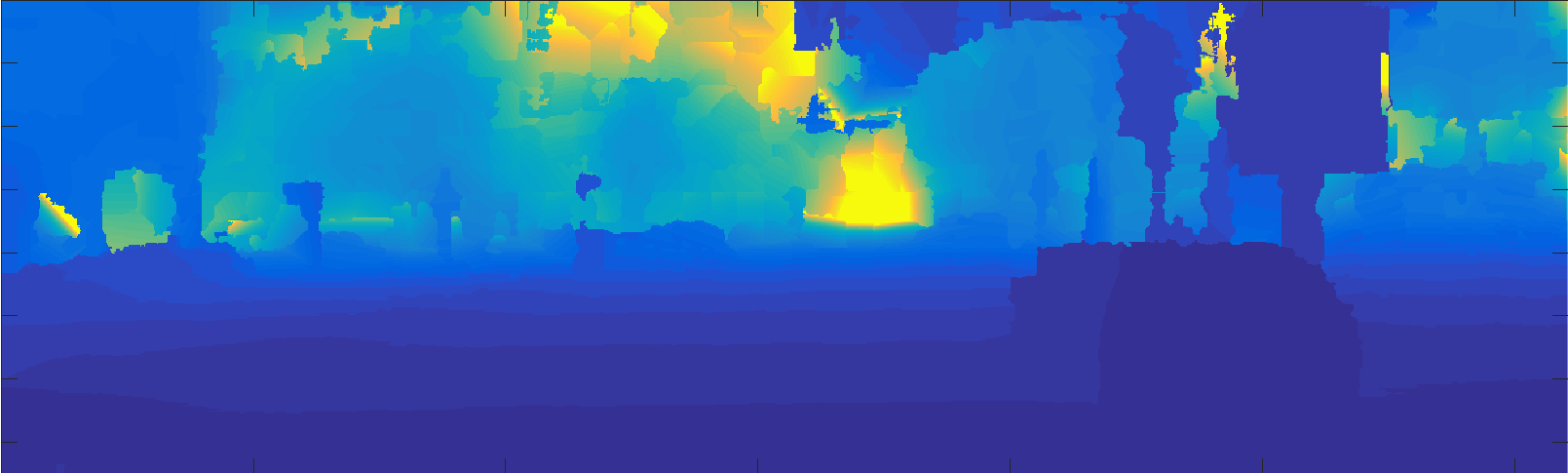
axis equal;

% save result

result\_name = fullfile('../results', strcat('q2\_a\_', imname, '.png'));

saveas(fig, result\_name);

End

**Results:**

**b):**

**Detect object**

**Detect\_object.m:**

function [ds, bs] = detectObject(detectorType, imname, model\_thresh, nms\_thresh)

data = getData([], [], detectorType);

model = data.model;

col = 'r';

imdata = getData(imname, 'test', 'left');

im = imdata.im;

f = 1.5;

imr = imresize(im,f); % if we resize, it works better for small objects

% detect objects

fprintf('running the detector, may take a few seconds...\n');

tic; % measure running time

[ds, bs] = imgdetect(imr, model, model\_thresh); % you may need to reduce the threshold if you want more detections

e = toc;

fprintf('finished! (took: %0.4f seconds)\n', e);

% non maximum suppression

top = nms(ds, nms\_thresh);

if model.type == model\_types.Grammar

bs = [ds(:,1:4) bs];

end

if ~isempty(ds)

% resize back

ds(:, 1:end-2) = ds(:, 1:end-2)/f;

bs(:, 1:end-2) = bs(:, 1:end-2)/f;

end;

% showboxesMy(im, reduceboxes(model, bs(top,:)), col);

fprintf('detections:\n');

ds = ds(top, :);

**Q2\_b.m:**

globals;

test\_fid = fopen(fullfile(TEST\_DIR, 'test.txt'));

imname = fgetl(test\_fid);

% detext car, cyclist, person for all test image and save into results

% save will overwrite the file if exist

while ischar(imname)

[car\_ds, car\_bs] = detectObject('detector-car', imname, -0.6, 0.1);

car\_ds\_filename = fullfile(RESULTS\_DIR, strcat(imname, '\_car\_ds.mat'));

car\_bs\_filename = fullfile(RESULTS\_DIR, strcat(imname, '\_car\_bs.mat'));

save(car\_ds\_filename, 'car\_ds');

save(car\_bs\_filename, 'car\_bs');

[cyclist\_ds, cyclist\_bs] = detectObject('detector-cyclist', imname, 0, 0.1);

cyclist\_ds\_filename = fullfile(RESULTS\_DIR, strcat(imname, '\_cyclist\_ds.mat'));

cyclist\_bs\_filename = fullfile(RESULTS\_DIR, strcat(imname, '\_cyclist\_bs.mat'));

save(cyclist\_ds\_filename, 'cyclist\_ds');

save(cyclist\_bs\_filename, 'cyclist\_bs');

[person\_ds, person\_bs] = detectObject('detector-person', imname, -0.6, 0.1);

person\_ds\_filename = fullfile(RESULTS\_DIR, strcat(imname, '\_person\_ds.mat'));

person\_bs\_filename = fullfile(RESULTS\_DIR, strcat(imname, '\_person\_bs.mat'));

save(person\_ds\_filename, 'person\_ds');

save(person\_bs\_filename, 'person\_bs');

imname = fgetl(test\_fid);

end

fclose(test\_fid);

**c):**

**Visualize detection**

**drawAndLabelBoxes.m:**

function drawAndLabelBoxes(ds, label, fig)

num\_rows = size(ds, 1);

if num\_rows > 0

for row = 1:num\_rows

x\_left = ds(row, 1); x\_right = ds(row, 3);

y\_top = ds(row, 2); y\_bottom = ds(row, 4);

lineX = [ x\_left, x\_right , x\_right , x\_left , x\_left ] ;

lineY = [ y\_bottom , y\_bottom, y\_top , y\_top , y\_bottom ] ;

figure(fig);

hold on;

switch label

case 'car'

line( lineX , lineY, 'Color', 'r', 'LineWidth', 3 ) ;

case 'person'

line( lineX , lineY, 'Color', 'b', 'LineWidth', 3 ) ;

case 'cyclist'

line( lineX , lineY, 'Color', 'c', 'LineWidth', 3 ) ;

end

text(x\_left, y\_top, label, 'Color', 'w', 'FontSize', 16, 'FontWeight', 'bold');

end

end

**Q2\_c.m:**

globals;

imnames = {'004945', '004964', '005002'};

for i = 1:length(imnames)

imname = imnames{i};

imdata = getData(imname, 'test', 'left');

im = imdata.im;

car\_ds\_data = getData(imname, 'test', 'car\_ds');

car\_ds = car\_ds\_data.car\_ds;

person\_ds\_data = getData(imname, 'test', 'person\_ds');

person\_ds = person\_ds\_data.person\_ds;

cyclist\_ds\_data = getData(imname, 'test', 'cyclist\_ds');

cyclist\_ds = cyclist\_ds\_data.cyclist\_ds;

fig = figure;

imshow(im);

drawAndLabelBoxes(car\_ds, 'car', fig);

drawAndLabelBoxes(person\_ds, 'person', fig);

drawAndLabelBoxes(cyclist\_ds, 'cyclist', fig);

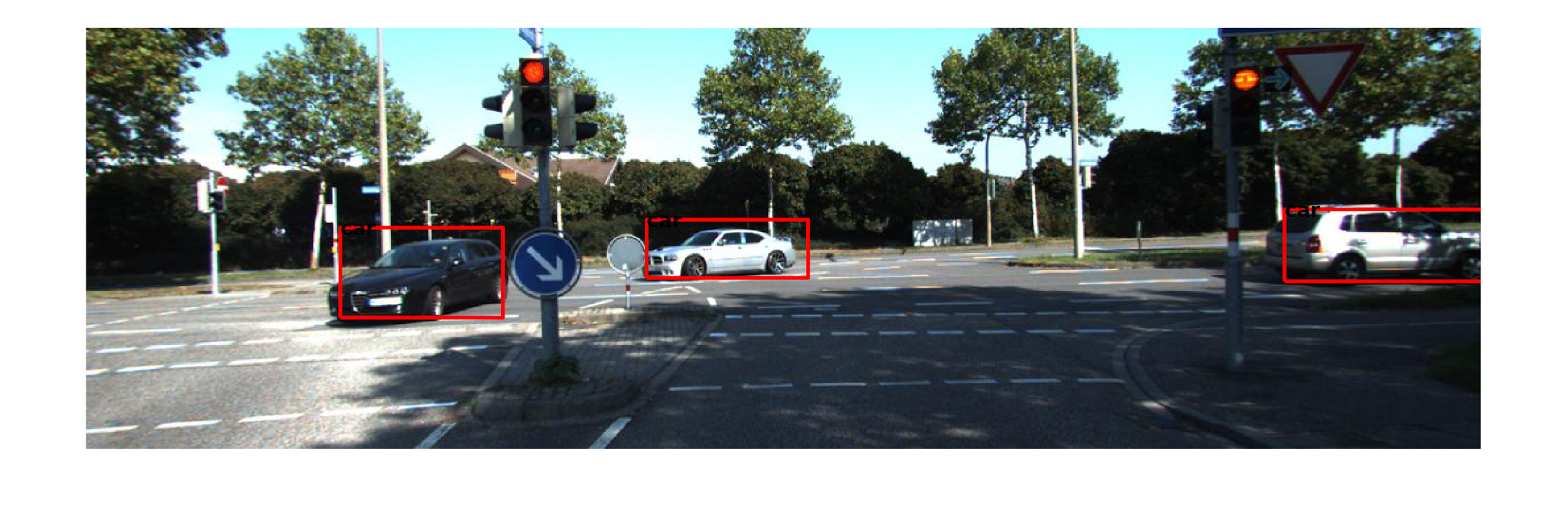
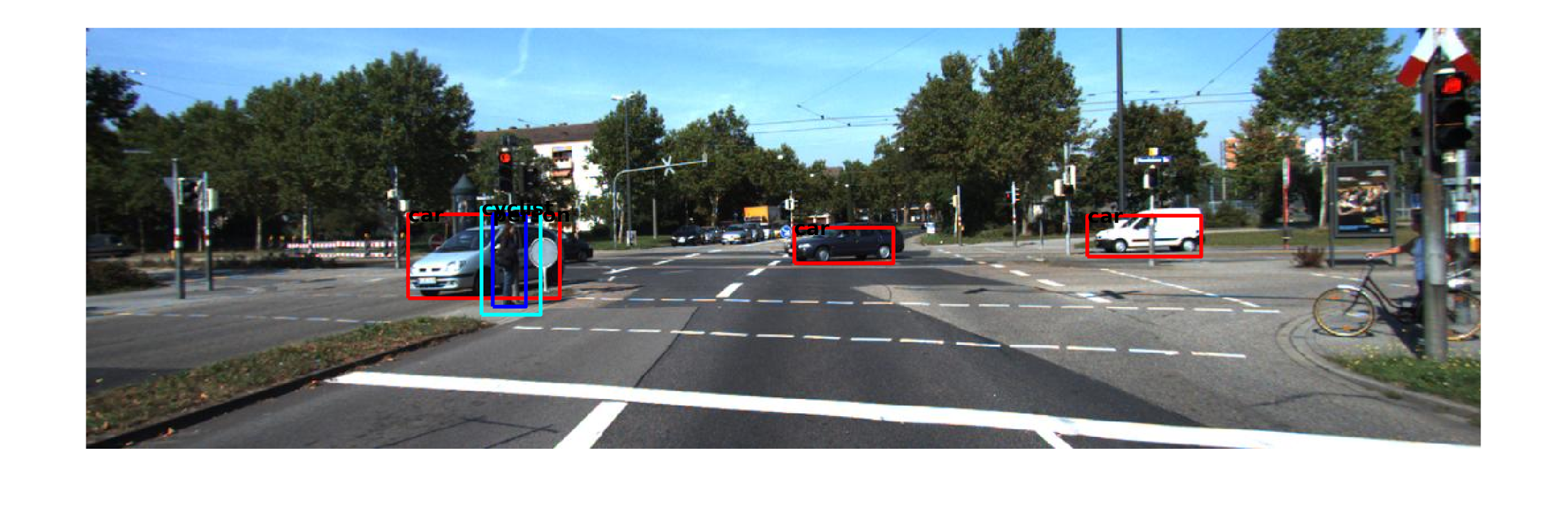
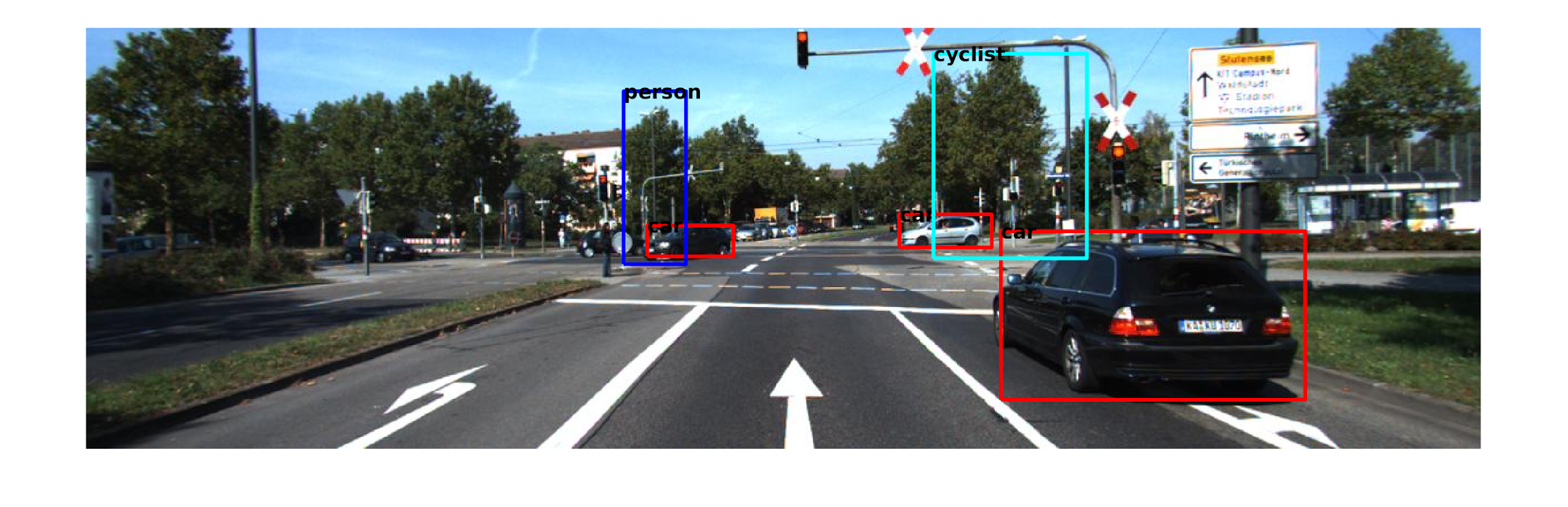
% save result

result\_name = fullfile('../results', strcat('q2\_b\_', imname, '.png'));

saveas(fig, result\_name);

End

**Results:**



**d):**

**Compute 3D location**

**ComputeCenterLocation.m:**

function ds = computeCenterLocation(ds, location)

% given ds and corresponding 3d locations

% compute center location for each detection and store them back into ds

% col 7:9 in ds are center locations

num\_detections = size(ds, 1);

num\_rows = size(location, 1);

num\_cols = size(location, 2);

if num\_detections > 0

if size(ds, 2) == 6

ds = [ds zeros(num\_detections, 3)];

end

for row = 1:num\_detections

x\_left = round(ds(row, 1)); x\_left = min(max(x\_left, 1), num\_cols);

x\_right = round(ds(row, 3)); x\_right = min(max(x\_right, 1), num\_cols);

y\_top = round(ds(row, 2)); y\_top = min(max(y\_top, 1), num\_rows);

y\_bottom = round(ds(row, 4)); y\_bottom = min(max(y\_bottom, 1), num\_rows);

detection\_location = location(y\_top:y\_bottom, x\_left:x\_right, :);

center\_location = reshape(mean(mean(detection\_location, 1), 2), [1 3]);

ds(row, 7:9) = center\_location;

end

else

ds = zeros(0, 9);

End

**Q2\_d.m:**

globals;

imnames = {'004945', '004964', '005002'};

for i = 1:length(imnames)

% get depth, K, car ds, person ds, cyclist ds

imname = imnames{i};

depth\_data = getData(imname, 'test', 'depth');

depth = depth\_data.depth;

calib\_data = getData(imname, 'test', 'calib');

K = calib\_data.K;

car\_ds\_data = getData(imname, 'test', 'car\_ds');

car\_ds = car\_ds\_data.car\_ds;

person\_ds\_data = getData(imname, 'test', 'person\_ds');

person\_ds = person\_ds\_data.person\_ds;

cyclist\_ds\_data = getData(imname, 'test', 'cyclist\_ds');

cyclist\_ds = cyclist\_ds\_data.cyclist\_ds;

num\_rows = size(depth, 1);

num\_cols = size(depth, 2);

% compute and save 3d location for each pixel in image with name imname

location = zeros(num\_rows, num\_cols, 3);

for row = 1:num\_rows

y = num\_rows+1-row;

for col = 1:num\_cols

x = col;

Z = depth(row, col);

result = K\[x; y; 1];

w = Z/result(3);

X = result(1)\*w;

Y = result(2)\*w;

location(row, col, :) = [X Y Z];

end

end

location\_filename = fullfile(RESULTS\_DIR, strcat(imname, '\_location.mat'));

save(location\_filename, 'location');

% compute and save center 3d location for each car detected

car\_ds = computeCenterLocation(car\_ds, location);

car\_ds\_filename = fullfile(RESULTS\_DIR, strcat(imname, '\_car\_ds.mat'));

save(car\_ds\_filename, 'car\_ds');

% compute and save center 3d location for each cyclist detected

cyclist\_ds = computeCenterLocation(cyclist\_ds, location);

cyclist\_ds\_filename = fullfile(RESULTS\_DIR, strcat(imname, '\_cyclist\_ds.mat'));

save(cyclist\_ds\_filename, 'cyclist\_ds');

% compute and save center 3d location for each person detected

person\_ds = computeCenterLocation(person\_ds, location);

person\_ds\_filename = fullfile(RESULTS\_DIR, strcat(imname, '\_person\_ds.mat'));

save(person\_ds\_filename, 'person\_ds');

End

**e):**

**Perform segmentation**

**Q2\_e.m:**

globals;

imnames = {'004945', '004964', '005002'};

for i = 1:length(imnames)

imname = imnames{i};

% get location, ds and (center location)

location\_data = getData(imname, 'test', 'location');

location = location\_data.location;

car\_ds\_data = getData(imname, 'test', 'car\_ds');

car\_ds = car\_ds\_data.car\_ds;

person\_ds\_data = getData(imname, 'test', 'person\_ds');

person\_ds = person\_ds\_data.person\_ds;

cyclist\_ds\_data = getData(imname, 'test', 'cyclist\_ds');

cyclist\_ds = cyclist\_ds\_data.cyclist\_ds;

% concatenate all ds together

all\_ds = [cyclist\_ds; car\_ds; person\_ds];

num\_rows = size(location, 1);

num\_cols = size(location, 2);

num\_detections = size(all\_ds, 1);

segmentation = zeros(num\_rows, num\_cols);

% find and label all pixel that is segmented

for j = 1:num\_detections

center\_location = all\_ds(j, 7:9);

x\_left = round(all\_ds(j, 1)); x\_left = min(max(x\_left, 1), num\_cols);

x\_right = round(all\_ds(j, 3)); x\_right = min(max(x\_right, 1), num\_cols);

y\_top = round(all\_ds(j, 2)); y\_top = min(max(y\_top, 1), num\_rows);

y\_bottom = round(all\_ds(j, 4)); y\_bottom = min(max(y\_bottom, 1), num\_rows);

for col = x\_left:x\_right

for row = y\_top:y\_bottom

if norm(reshape(location(row, col, :), [1 3])-center\_location) <= 15

segmentation(row, col) = j;

end

end

end

end

fig = figure;

imagesc(segmentation);

truesize(fig);

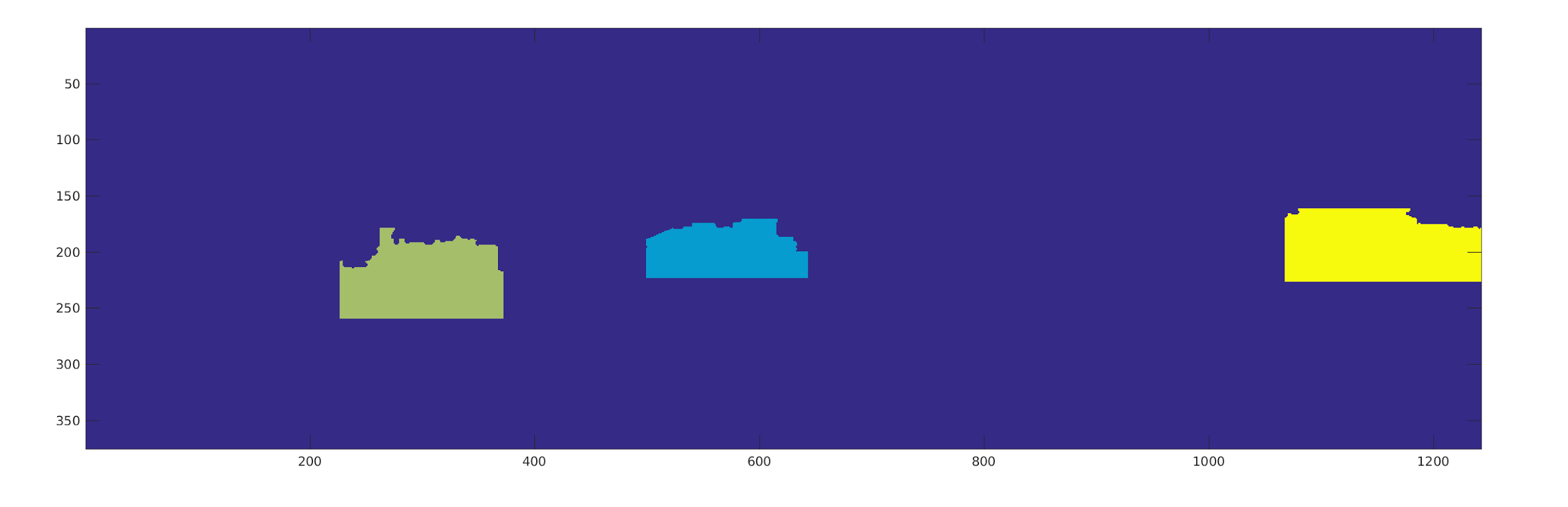
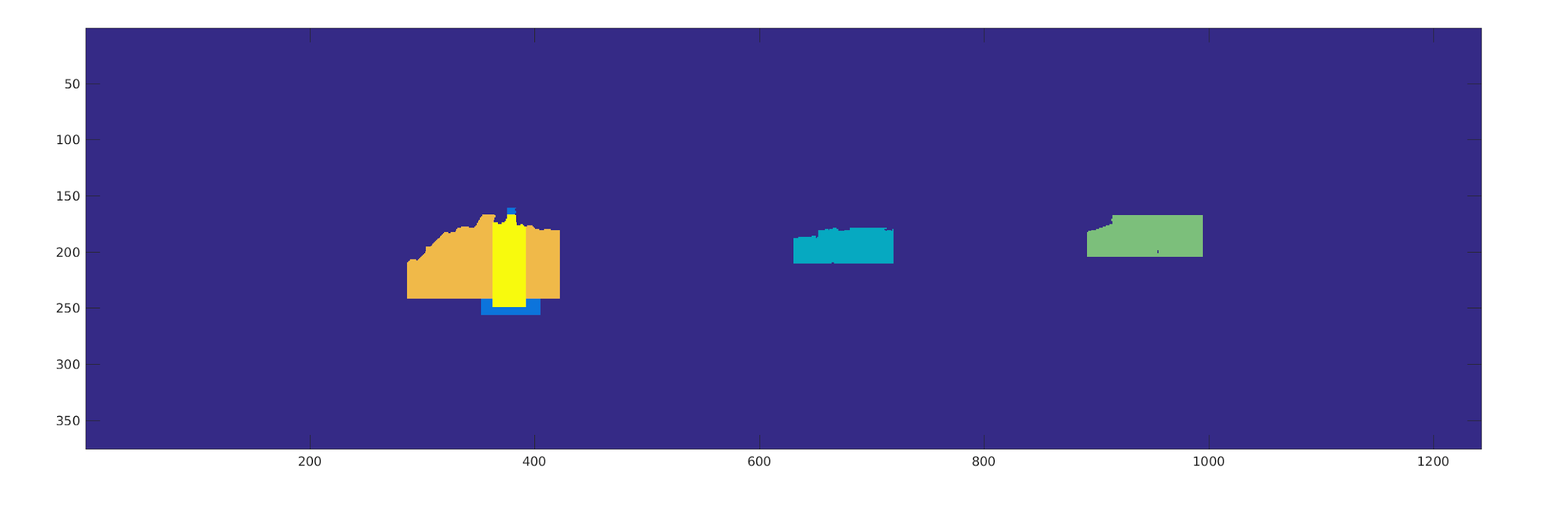
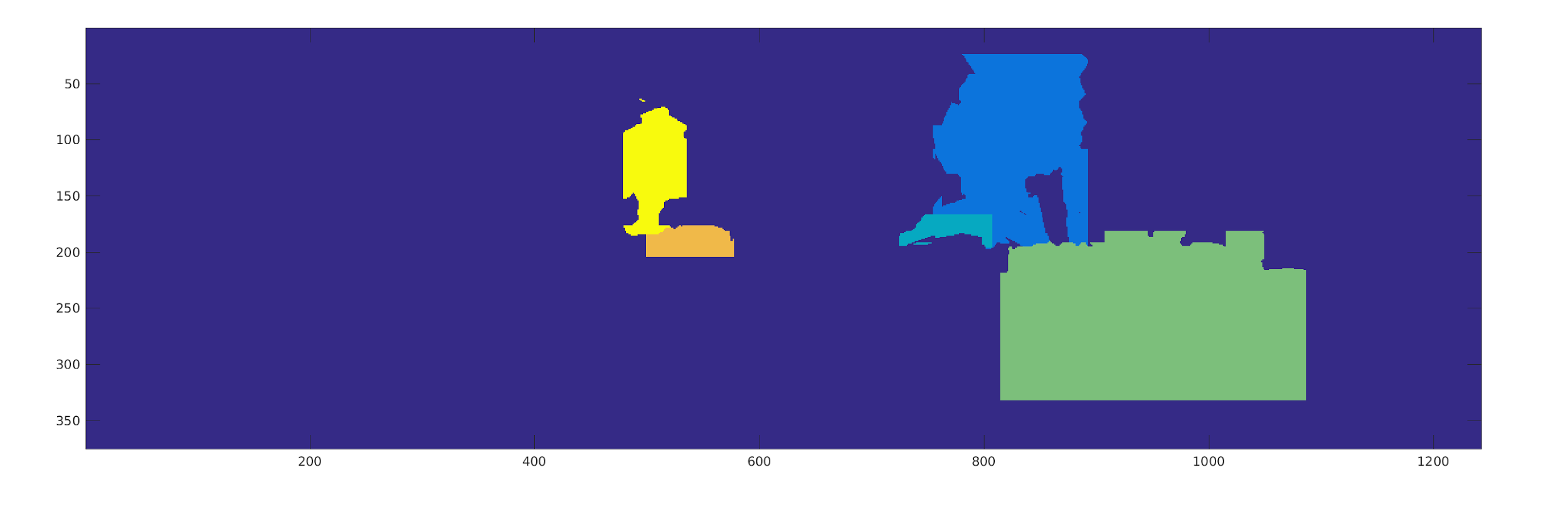
% save result

result\_name = fullfile('../results', strcat('q2\_e\_', imname, '.png'));

saveas(fig, result\_name);

End

**Results:**



**f):**

**Create textual description**

**Q2\_f.m:**

globals;

imnames = {'004945', '004964', '005002'};

for i = 1:length(imnames)

imname = imnames{i};

% get ds and (center location)

car\_ds\_data = getData(imname, 'test', 'car\_ds');

car\_ds = car\_ds\_data.car\_ds;

person\_ds\_data = getData(imname, 'test', 'person\_ds');

person\_ds = person\_ds\_data.person\_ds;

cyclist\_ds\_data = getData(imname, 'test', 'cyclist\_ds');

cyclist\_ds = cyclist\_ds\_data.cyclist\_ds;

num\_car = size(car\_ds, 1);

num\_person = size(person\_ds, 1);

num\_cyclist = size(cyclist\_ds, 1);

% concatenate ds together, label car as 1, person as 2, cyclist as 3

all\_ds = [car\_ds; person\_ds; cyclist\_ds];

all\_ds(1:num\_car, 10) = 1;

all\_ds(num\_car+1:num\_car+num\_person, 10) = 2;

all\_ds(num\_car+num\_person+1:num\_car+num\_person+num\_cyclist, 10) = 3;

num\_detections = size(all\_ds, 1);

% find closed object type and distance

closest\_distance = inf(1);

closest\_object = 'unknown';

left\_right = 'unknown';

for j = 1:num\_detections

distance = norm(all\_ds(j,7:9));

if distance < closest\_distance

closest\_distance = distance;

if all\_ds(j, 7) < 0

left\_right = 'left';

else

left\_right = 'right';

end

switch all\_ds(j, 10)

case 1

closest\_object = 'car';

case 2

closest\_object = 'person';

case 3

closest\_object = 'cyclist';

end

end

End

% print textual description

fprintf('for image %s: %d car, %d person, and %d cyclist are detected.\n',...

imname, num\_car, num\_person, num\_cyclist);

fprintf('There is a %s on your %s, which is %d meters away from you\n\n',...

closest\_object, left\_right, round(closest\_distance));

End

**Results:**

for image 004945: 3 car, 1 person, and 1 cyclist are detected.

There is a car on your right, which is 12 meters away from you

for image 004964: 3 car, 1 person, and 1 cyclist are detected.

There is a person on your left, which is 23 meters away from you

for image 005002: 3 car, 0 person, and 0 cyclist are detected.

There is a car on your left, which is 24 meters away from you