# CSC420 A2 Report

Tianshu Zhu
1002111225

# Q1

# a):

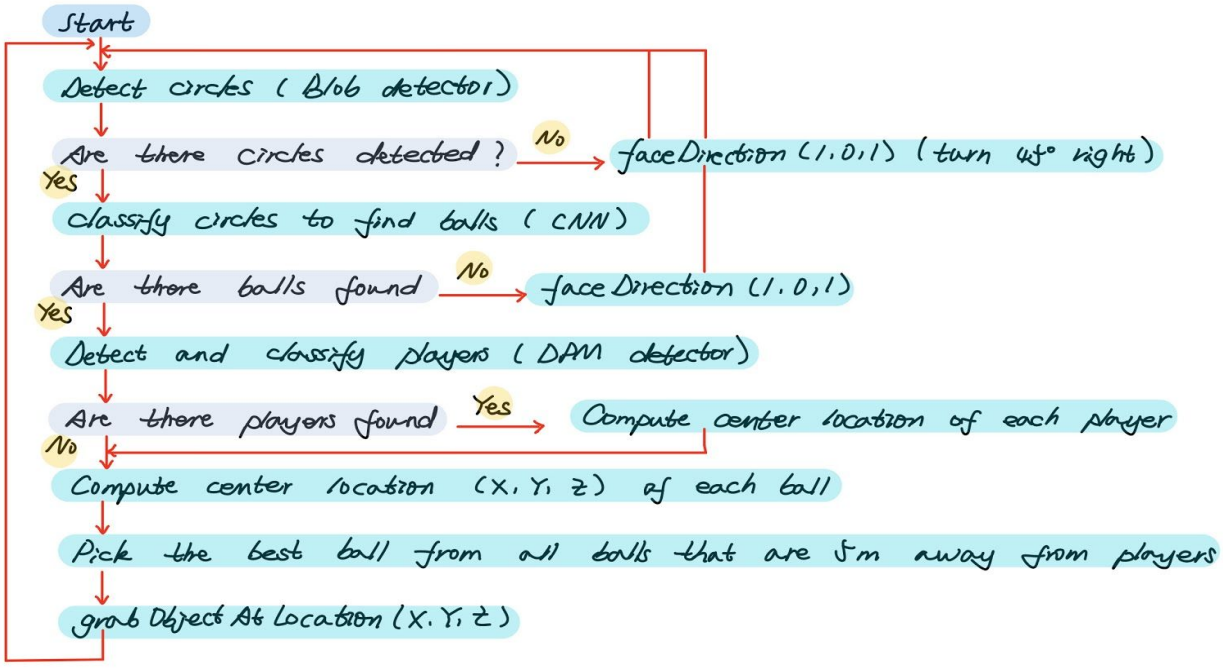
## Brain storm

- 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

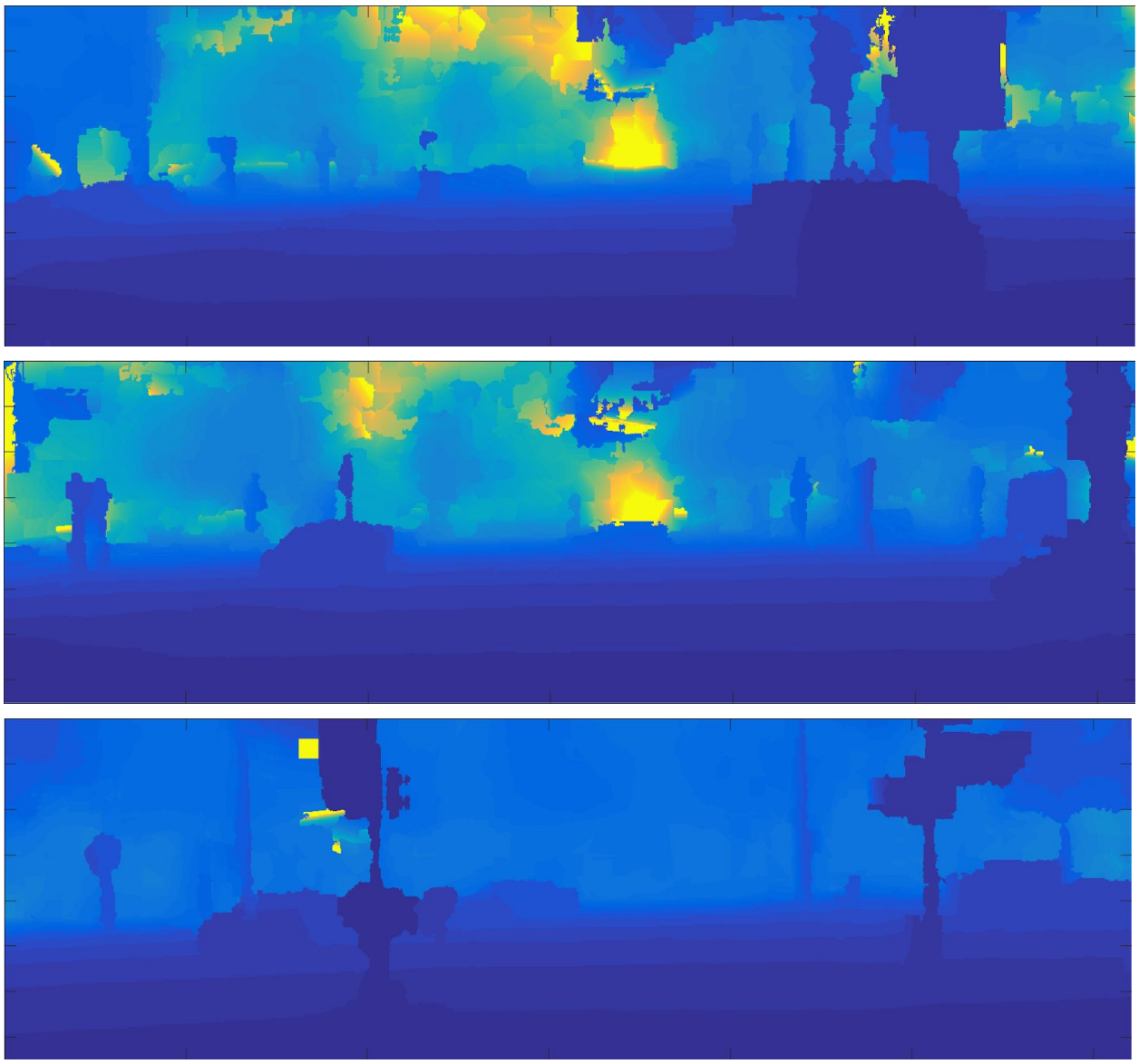
# a):

### 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:

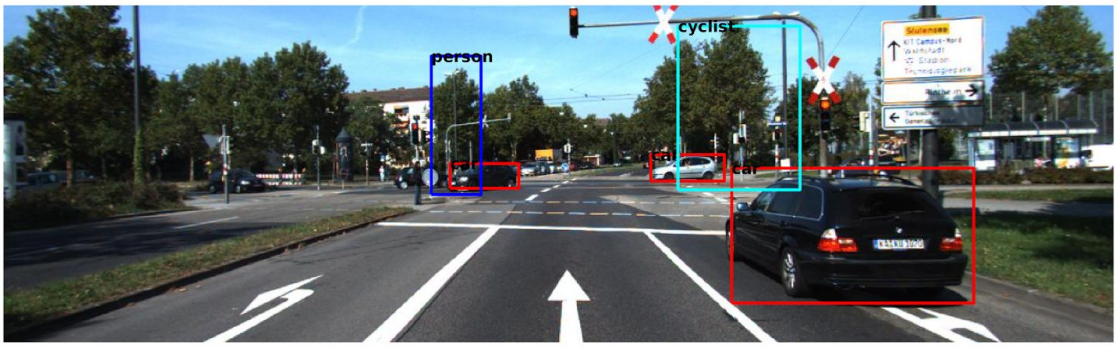
```
function drawAndLabelBoxes(ds, label, fig)
num_rows = size(ds, 1);
if num_rows > 0
	for row = 1:num_rows
		x_left = ds(row, 1); x_right = ds(row, 3);
		y_top = ds(row, 2); y_bottom = ds(row, 4);
		lineX = [ x_left, x_right , x_right , x_left , x_left ] ;
		lineY = [ y_bottom , y_bottom, y_top , y_top , 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:

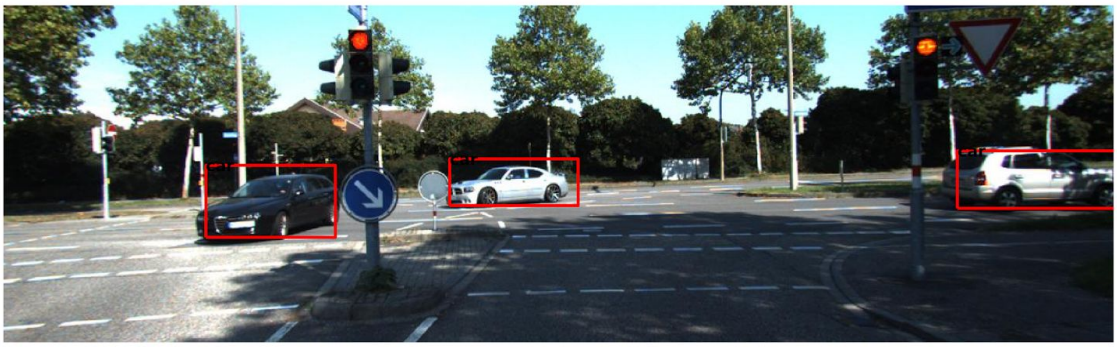
```
globals;
imnames = {'004945', '004964', '005002'};
for i = 1:length(imnames)
	imname = imnames{i};
	imdata = getData(imname, 'test', 'left');
	im = imdata.im;
	car_ds_data = getData(imname, 'test', 'car_ds');
	car_ds = car_ds_data.car_ds;
	person_ds_data = getData(imname, 'test', 'person_ds');
	person_ds = 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):

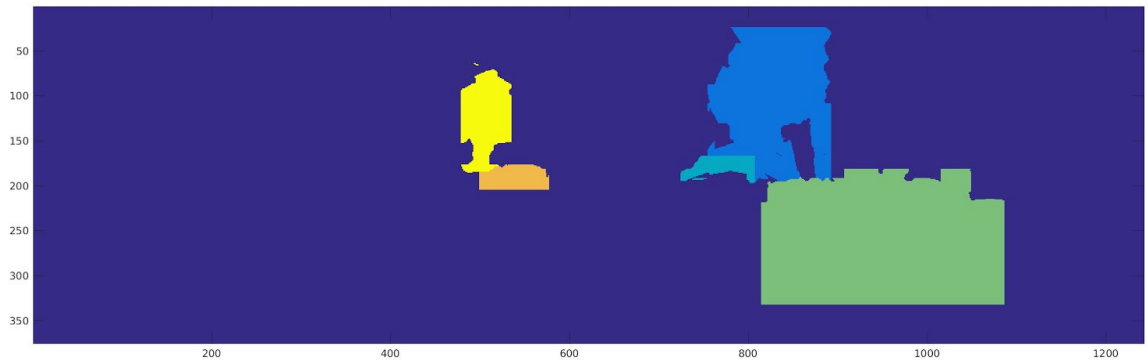
## Perform segmentation

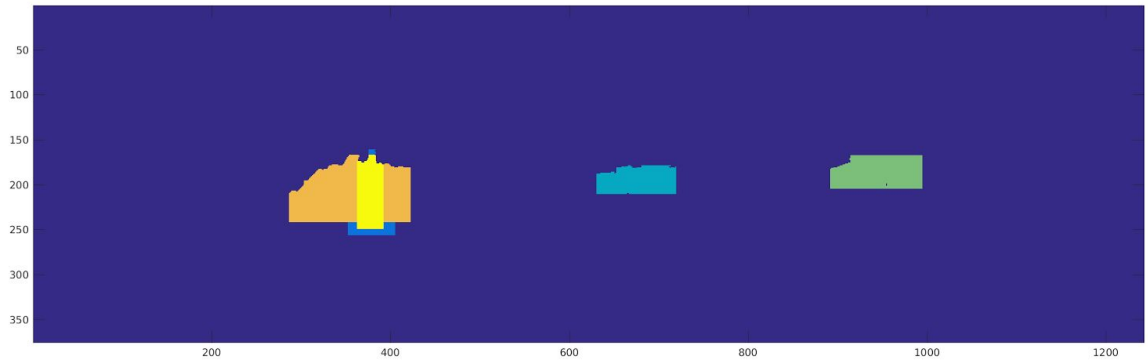
## Q2_e.m:

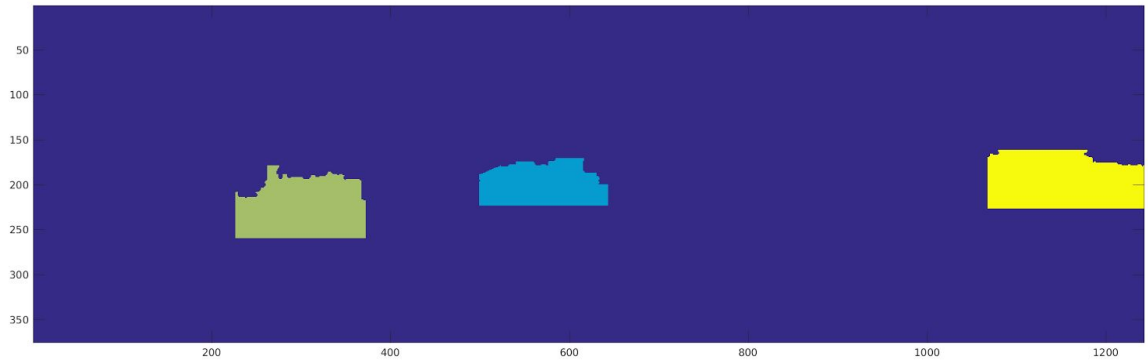
```
globals;
imnames = {'004945', '004964', '005002'};
for i = 1:length(imnames)
	imname = imnames{i};
	% get location, ds and (center location)
	location_data = getData(imname, 'test', 'location');
	location = location_data.location;
	car_ds_data = getData(imname, 'test', 'car_ds');
	car_ds = car_ds_data.car_ds;
	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