Q1a.)

Training data: tennis court data and it needs to know where it is, so it may needs some landmark too.

Training data: tennis ball data, such as color, size, etc...

2. Navigate itself to tennis ball location.

1. Detect tennis ball

3. Player and obstacle detection and avoidance.

4. Real time localization

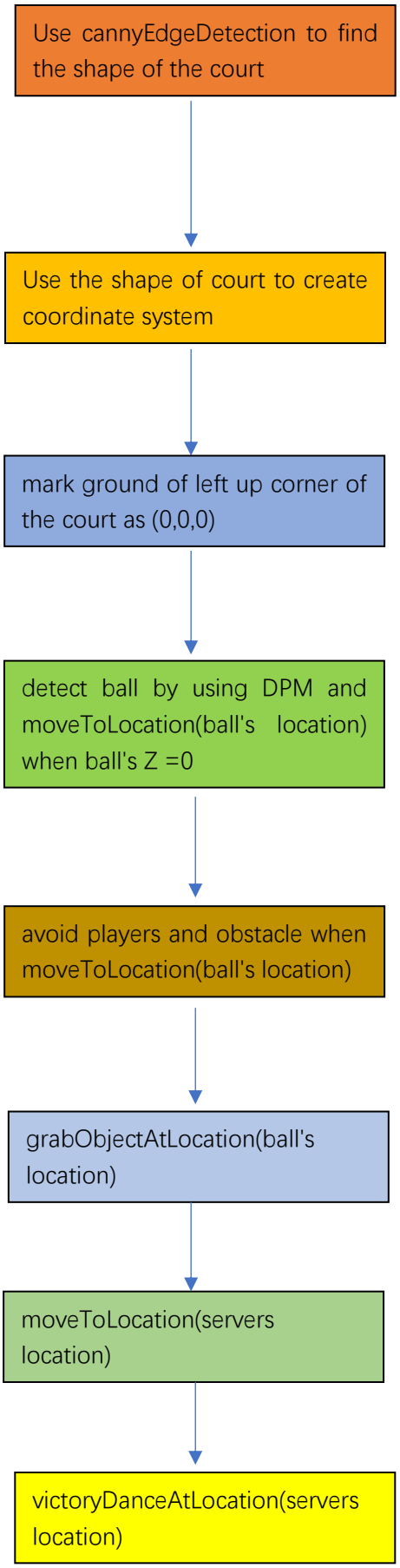
5. Delivery tennis balls back to servers.

Training data: player and obstacle's data so that it can recognize this is obstacles.

Training data: landmark, tennis court data

Training data: landmark, tennis court data, servers location data.

Q1b.):

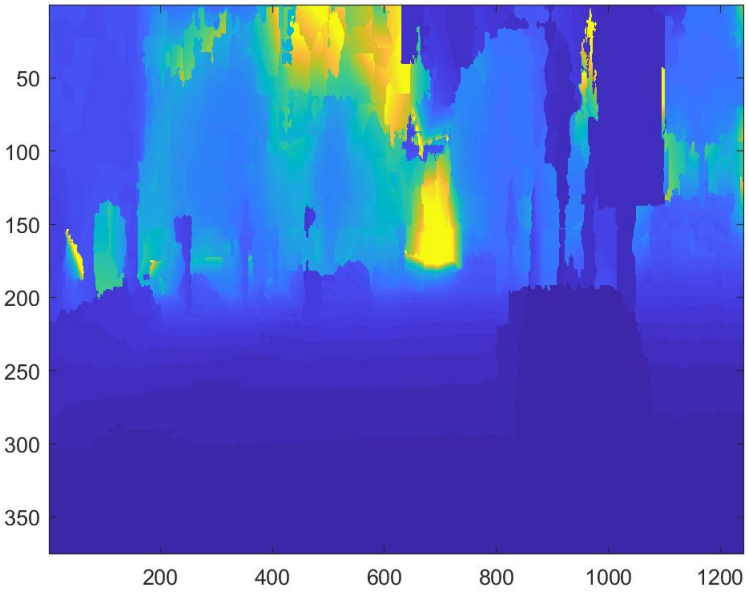

Use cannyEdgeDetection to find the shape of the court
Use the shape of court to create coordinate system
mark ground of left up corner of the court as (0,0,0)
detect ball by using DPM and moveToLocation(ball's location) when ball's Z =0
avoid players and obstacle when moveToLocation(ball's location)
grabObjectAtLocation(ball's location)
moveToLocation(servers location)
victoryDanceAtLocation(servers location)

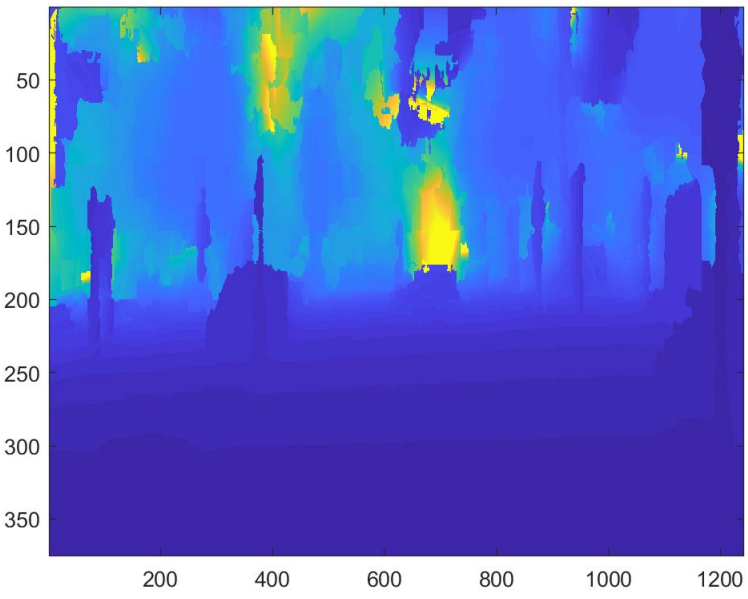
Q1c.):

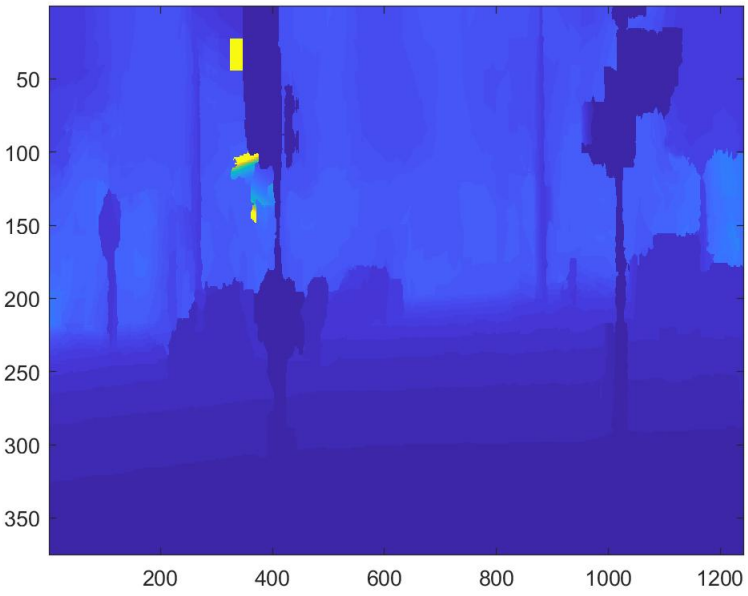
```
Boundary_map = cannyEdgeDetection(tennis_court_map);
Save_to_robot(Boundary_map);        # Use the shape of court to create coordinate system
While(1):
    Read(image_frame);
    [X,Y,Z] = DPM(tennis_ball);
    If Z == 0:
        moveToLocation(X,Y,Z)    #move to ball location when ball reach ground
        If obstacle_Detect() == true:
            Avoid();
        If faceDirection() == [X,Y,Z]:      #arrived ball's location
            grabObjectAtLocation(ball's location);
            moveToLocation(servers location);
```

Question 2

a.)


50
100
150
200
250
300
350
200
400
600
800
1000
1200

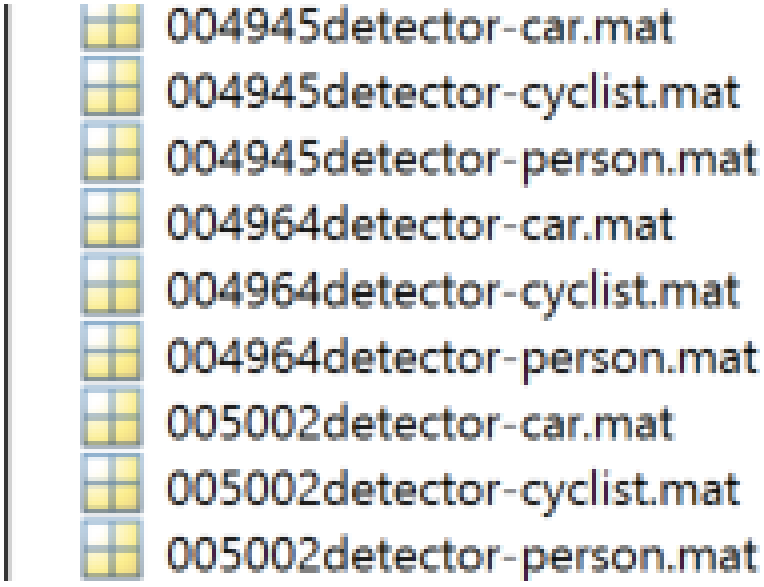

50
100
150
200
250
300
350
200
400
600
800
1000
1200



For Detail, Please visit Appendix.

b.)

I store every DS for every type for every image under code folder as mat file. In each mat file, it contains both bs and ds. just like the following figure:



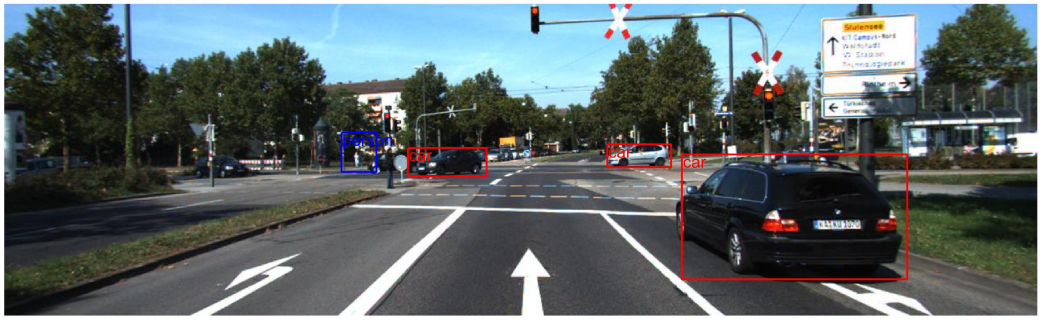
For Detail, Please visit Appendix.

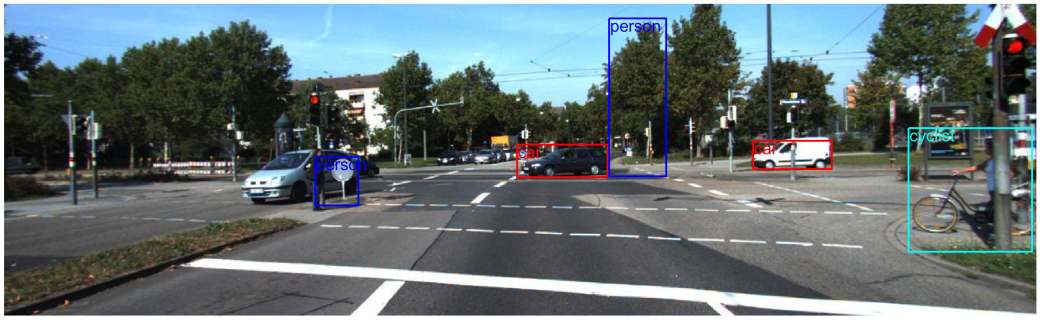
c.)

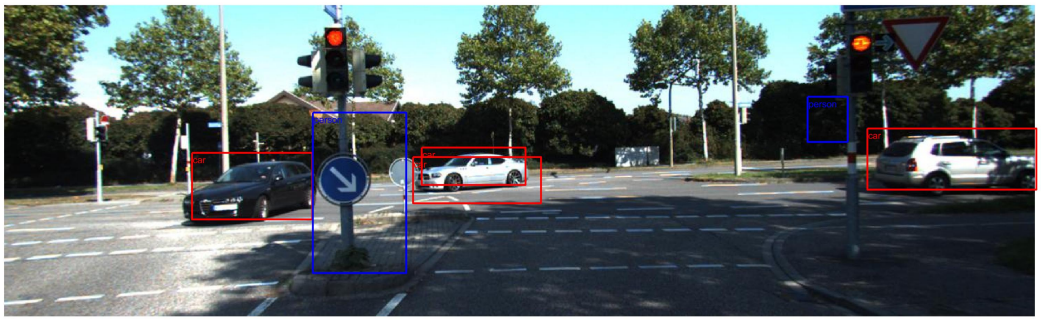
threshold for car: 0

threshold for person: -0.55

threshold for cyclist: -0.5





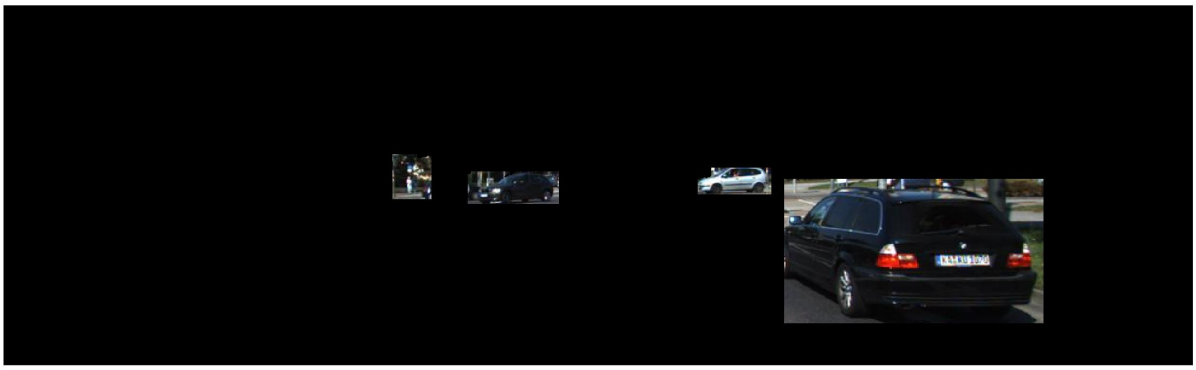


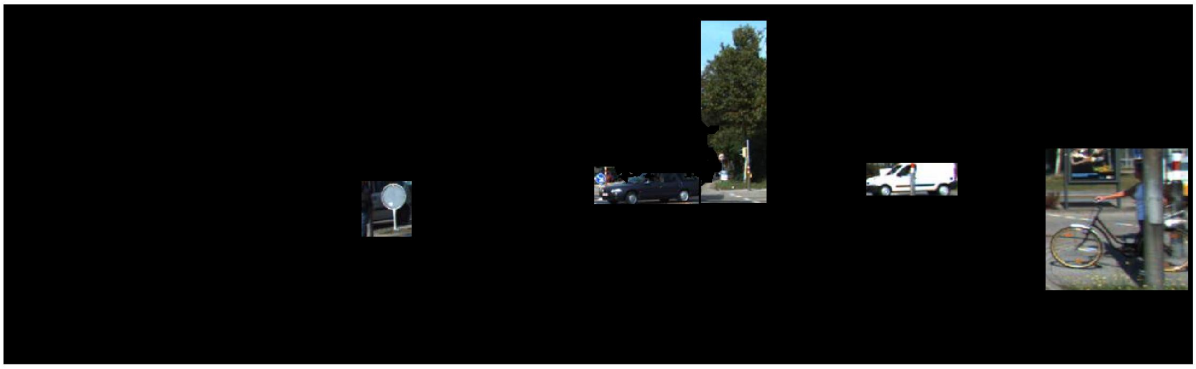
For Detail, Please visit Appendix.

d.)
I will find all pixels inside each bounding box, and then compute these pixel's world coordinate, and then average them. The average value of X,Y,Z will be the 3D location (centre of mass) of each detected object.

For Detail, Please visit Appendix.

e.)





f.)

In 004945:

There is a car to you right with distances 5.712012e+01

There is a car to you right with distances 1.150438e+01

There is a car to you left with distances 4.654811e+01

There is a car to you left with distances 7.571202e+01

In 004964:

There is a car to you right with distances 8.281848e+01

There is a car to you right with distances 4.007096e+01

There is a car to you right with distances 8.276886e+01

There is a car to you left with distances 1.915666e+01

There is a car to you right with distances 1.928514e+01

In 005002:

There is a car to you left with distances 2.929314e+01

There is a car to you left with distances 2.419383e+01

There is a car to you left with distances 2.476850e+01

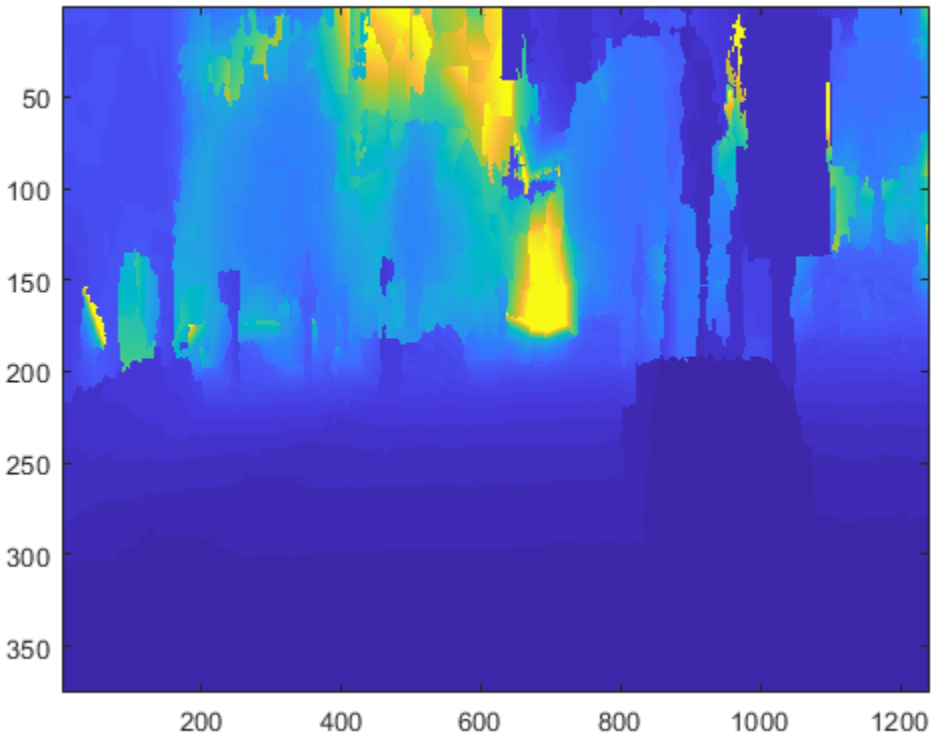
There is a car to you right with distances 3.397670e+01

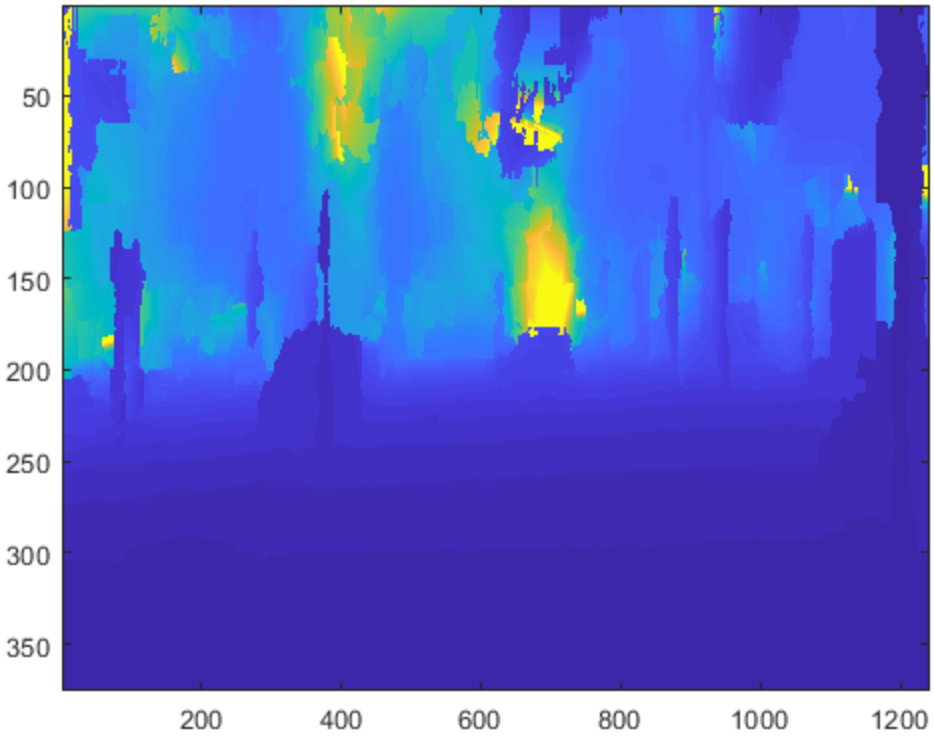
There is a car to you right with distances 5.260385e+01

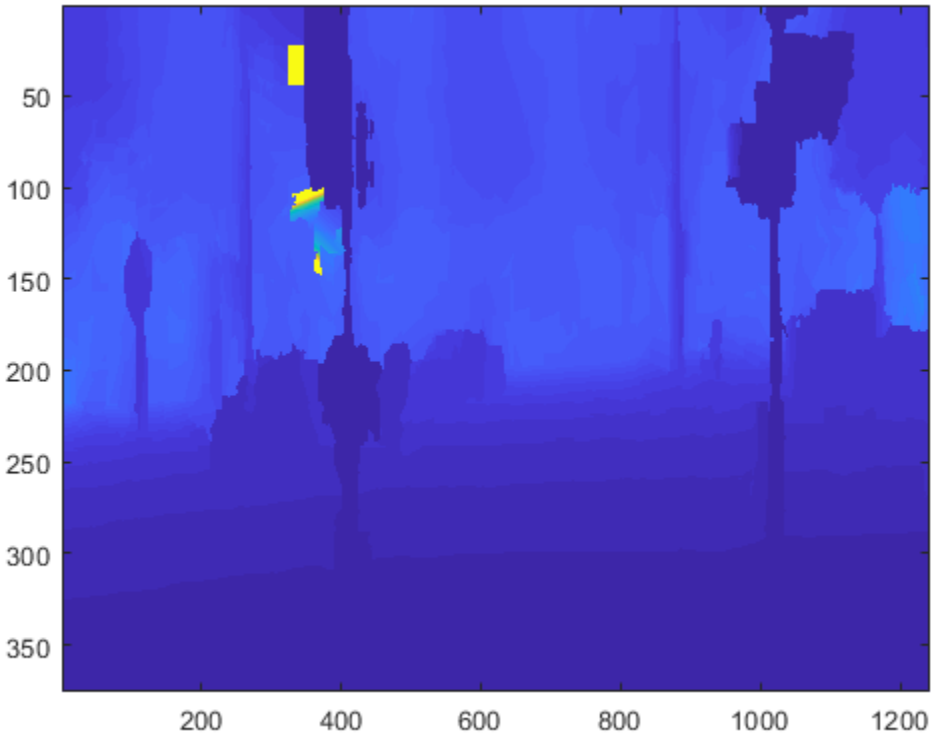
There is a car to you left with distances 2.207100e+01

Q2a

```
data = getData([], 'test','list');
ids = data.ids(1:3);
for i= 1:3
 calib = getData(ids{i}, 'test', 'calib');
 disp = getData(ids{i}, 'test', 'disp');
 disparity = disp.disparity;
 numerator = calib.f*calib.baseline;
 depth = numerator./disparity;
 %incase depth larger than 255
 depth(depth>255)=255;
 figure;imagesc(depth);
end
```




50
100
150
200
250
300
350
200
400
600
800
1000
1200


50
100
150
200
250
300
350
200
400
600
800
1000
1200

Published with MATLAB® R2017b

Q2b

```
addpath('dpm') ;
addpath('devkit') ;

col = 'r';
imdata = getData([], 'test','list');
ids = imdata.ids(1:3);
for i = 1:3
    DS = [];
    f = 1.5;
    image = getData(ids{i}, 'test', 'left');
    im = image.im;
    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;
    %[ds, bs] = imgdetect(imr, model, model.thresh); % you may need to
 reduce the threshold if you want more detections
    detect_list = {'detector-car','detector-person','detector-
cyclist'};
    thereshold = {0,-0.55,-0.5};
    for detect_label  = 1:3
        data = getData([], [], detect_list{detect_label});
        model = data.model;
        [ds, bs] = imgdetect(imr,model, thereshold{detect_label});
        e = toc;
        fprintf('finished! (took: %0.4f seconds)\n', e);
        name = strcat(ids{i},detect_list{detect_label});
        if ~isempty(ds)
            % resize back
            ds(:, 1:end-2) = ds(:, 1:end-2)/f;
            bs(:, 1:end-2) = bs(:, 1:end-2)/f;
            top = nms(ds, 0.5);
            ds = ds(top,:);
            bs = bs(top,:);
        end
        save(name,'ds','bs');
    end
end
```

```
running the detector, may take a few seconds...
finished! (took: 65.4113 seconds)
finished! (took: 138.1273 seconds)
finished! (took: 182.0020 seconds)
running the detector, may take a few seconds...
finished! (took: 64.2202 seconds)
finished! (took: 137.0133 seconds)
finished! (took: 181.5092 seconds)
running the detector, may take a few seconds...
finished! (took: 67.0462 seconds)
```

```
finished! (took: 139.7745 seconds)
finished! (took: 183.5678 seconds)
```

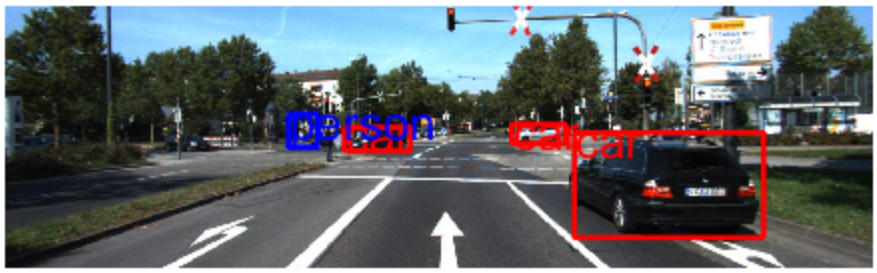
Published with MATLAB® R2017b

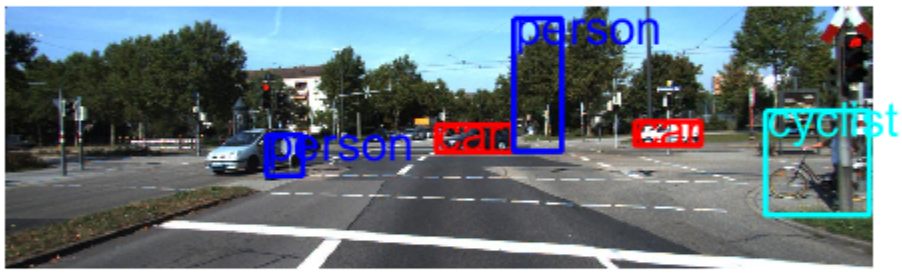
Q2c

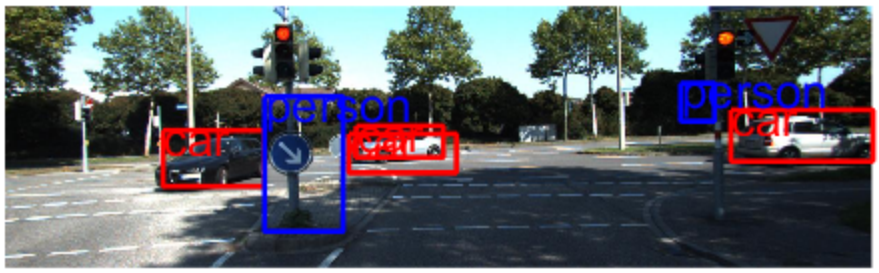
```
data = getData([], 'test','list');
ids = data.ids(1:3);
car_model = getData([], [], 'detector-car');
car_model = car_model.model;
person_model = getData([], [], 'detector-person');
person_model = person_model.model;
cyclist_model = getData([], [], 'detector-cyclist');
cyclist_model = cyclist_model.model;
for i = 1:3
    imdata = getData(ids{i}, 'test', 'left');
    im = imdata.im;
    car_ds = imdata.car.ds;
    person_ds = imdata.person.ds;
    cyclist_ds = imdata.cyclist.ds;
    car_bs = imdata.car.bs;
    person_bs = imdata.person.bs;
    cyclist_bs = imdata.cyclist.bs;
    figure;axis ij; hold on
    imagesc(im);
    if ~isempty(car_ds)
        showboxesMy(im, reduceboxes(car_model, car_bs), 'red');
        text(car_ds(:,1)+1,
 car_ds(:,2)+8,'car','Color','red','FontSize',18);
    end

   if ~isempty(person_ds)
        showboxesMy(im, reduceboxes(person_model, person_bs), 'blue');
        text(person_ds(:,1)+1,
 person_ds(:,2)+8,'person','Color','blue','FontSize',18);
   end
   if ~isempty(cyclist_ds)
        showboxesMy(im, reduceboxes(cyclist_model,
 cyclist_bs), 'cyan');
        text(cyclist_ds(:,1)+1,
 cyclist_ds(:,2)+8,'cyclist','Color','cyan','FontSize',18);
   end
   hold off;

end
```


person
car
car


person
person
car
cyclist



Published with MATLAB® R2017b

Q2d

DESCRIPTIVE TEXT

```
data = getData([], 'test','list');
ids = data.ids(1:3);

for i = 1:3
    %get image
    imdata = getData(ids{i}, 'test', 'left');
    im = imdata.im;

    %get car detections
    car_ds = imdata.car.ds;
    %if this is first time run Q2d, we need add 3 cols for avg x,y,z
    if size(car_ds,2) < 9
        car_ds = [car_ds zeros([size(car_ds,1),3])];
    end
    person_ds = imdata.person.ds;
    if size(person_ds,2) < 9
       person_ds = [person_ds zeros([size(person_ds,1),3])];
    end
    cyclist_ds = imdata.cyclist.ds;
    if size(cyclist_ds,2) < 9
       cyclist_ds = [cyclist_ds zeros([size(cyclist_ds,1),3])];
    end




    %create a white image for record pixels in boundary boxs
    inbox =0 * ones(size(im,1),size(im,2), 3, 'uint8');
    car_bs = imdata.car.bs;
    person_bs = imdata.person.bs;
    cyclist_bs = imdata.cyclist.bs;



    calib = getData(ids{i}, 'test', 'calib');
    disp = getData(ids{i}, 'test', 'disp');
    disparity = disp.disparity;
    f = calib.f;
    numerator = f*calib.baseline;
    depth = numerator./disparity;
    pleft = calib.P_left;
    [K, R, t]= KRt_from_P(pleft);
    Px = K(1,3);
    Py = K(2,3);



    for car = 1:size(car_ds,1)
```

```
        num = 0;
        avg_world_x = 0;
        avg_world_z = 0;
        avg_world_y = 0;
        for y = 1:size(inbox,1)
            for x = 1:size(inbox,2)
               % because t = Cw
               %compute the world coordinate for every pixel inside
the
               %boundary box
                if
and(and(x>car_ds(car,1),x<car_ds(car,3)),and(y>car_ds(car,2),y<car_ds(car,4)))
                    inbox(y,x,:) = im(y,x,:);
                    num = num + 1;
                    camera_Coor_Z = depth(y,x);
                    camera_Coor_X = (camera_Coor_Z.*(x - Px))./f;
                    camera_Coor_Y = (camera_Coor_Z.*(y - Py))./f;
                    camera_Coor = [camera_Coor_X camera_Coor_Y
camera_Coor_Z];
                    world_Coor = (camera_Coor - t)/R;
                    avg_world_x = avg_world_x + world_Coor(1,1);
                    avg_world_y = avg_world_y + world_Coor(1,2);
                    avg_world_z = avg_world_z + world_Coor(1,3);
                end
            end
        end
        avg_world_x = avg_world_x/num;
        avg_world_y = avg_world_y/num;
        avg_world_z = avg_world_z/num;
        car_ds(car,9) = avg_world_z;
        car_ds(car,7) = avg_world_x;
        car_ds(car,8) = avg_world_y;
    end
    ds = car_ds;
    bs = car_bs;
    save(imdata.car_name,'ds','bs');



    %same thing for person
    for person = 1:size(person_ds,1)
        num = 0;
        avg_world_x = 0;
        avg_world_z = 0;
        avg_world_y = 0;
        for y = 1:size(inbox,1)
            for x = 1:size(inbox,2)
                if
and(and(x>person_ds(person,1),x<person_ds(person,3)),and(y>person_ds(person,2),y<
                    inbox(y,x,:) = im(y,x,:);
                    num = num + 1;
                    camera_Coor_Z = depth(y,x);
                    camera_Coor_X = (camera_Coor_Z.*(x - Px))./f;
```

```
                    camera_Coor_Y = (camera_Coor_Z.*(y - Py))./f;
                    camera_Coor = [camera_Coor_X camera_Coor_Y
camera_Coor_Z];
                    world_Coor = (camera_Coor - t)/R;
                    avg_world_x = avg_world_x + world_Coor(1,1);
                    avg_world_y = avg_world_y + world_Coor(1,2);
                    avg_world_z = avg_world_z + world_Coor(1,3);
                end
            end
        end
        avg_world_x = avg_world_x/num;
        avg_world_y = avg_world_y/num;
        avg_world_z = avg_world_z/num;
        person_ds(person,9) = avg_world_z;
        person_ds(person,7) = avg_world_x;
        person_ds(person,8) = avg_world_y;
    end
    ds = person_ds;
    bs = person_bs;
    save(imdata.person_name,'ds','bs');



    %same thing for cyclist
    for cyclist = 1:size(cyclist_ds,1)
        num = 0;
        avg_world_x = 0;
        avg_world_z = 0;
        avg_world_y = 0;
        for y = 1:size(inbox,1)
            for x = 1:size(inbox,2)
                if
and(and(x>cyclist_ds(cyclist,1),x<cyclist_ds(cyclist,3)),and(y>cyclist_ds(cyclist
                    inbox(y,x,:) = im(y,x,:);
                    num = num + 1;
                    camera_Coor_Z = depth(y,x);
                    camera_Coor_X = (camera_Coor_Z.*(x - Px))./f;
                    camera_Coor_Y = (camera_Coor_Z.*(y - Py))./f;
                    camera_Coor = [camera_Coor_X camera_Coor_Y
camera_Coor_Z];
                    world_Coor = (camera_Coor - t)/R;
                    avg_world_x = avg_world_x + world_Coor(1,1);
                    avg_world_y = avg_world_y + world_Coor(1,2);
                    avg_world_z = avg_world_z + world_Coor(1,3);
                end
            end
        end
        avg_world_x = avg_world_x/num;
        avg_world_y = avg_world_y/num;
        avg_world_z = avg_world_z/num;
        cyclist_ds(cyclist,9) = avg_world_z;
        cyclist_ds(cyclist,7) = avg_world_x;
        cyclist_ds(cyclist,8) = avg_world_y;
```

```
    end
    ds = cyclist_ds;
    bs = cyclist_bs;
    save(imdata.cyclist_name,'ds','bs');
    %save inbox as mat file
    inbox_name = strcat(ids{i},'_inbox');
    save(inbox_name,'inbox');
end
```

Published with MATLAB® R2017b

Q2e

DESCRIPTIVE TEXT

```
data = getData([], 'test','list');
ids = data.ids(1:3);

for i = 1:3
    imdata = getData(ids{i}, 'test', 'left');
    im = imdata.im;

    %add three detection's ds into one ds for easier comparsion
    car_ds = imdata.car.ds;
    person_ds = imdata.person.ds;
    cyclist_ds = imdata.cyclist.ds;
    ds = [];
    if ~isempty(car_ds)
        ds =  [ds;car_ds(:,[7 8 9])];
    end
    if ~isempty(person_ds)
        ds = [ds;person_ds(:,[7 8 9])];
    end
    if ~isempty(cyclist_ds)
        ds = [ds;cyclist_ds(:,[7 8 9])];
    end
     calib = getData(ids{i}, 'test', 'calib');
     disp = getData(ids{i}, 'test', 'disp');
     disparity = disp.disparity;
     f = calib.f;
     fT = f*calib.baseline;
     depth = fT./disparity;
     pleft = calib.P_left;
     [K, R, t]= KRt_from_P(pleft);
     Px = K(1,3);
     Py = K(2,3);


    % inbox store all the pixel inside boundary box
    inbox = imdata.inbox.inbox;
    if inbox == 0
        inbox = im;
    end

    %loop all the pixels inside boundary box
    for y = 1:size(inbox(:,:,1),1)
        for x = 1:size(inbox(:,:,1),2)
            if and(and(inbox(y,x,1) ~= 0,inbox(y,x,2)
 ~=0),inbox(y,x,3) ~= 0)
                    camera_Coor_Z = depth(y,x);
                    camera_Coor_X = (camera_Coor_Z.*(x - Px))./f;
                    camera_Coor_Y = (camera_Coor_Z.*(y - Py))./f;
```

```
                    camera_Coor = [camera_Coor_X camera_Coor_Y
 camera_Coor_Z];
                    world_Coor = (camera_Coor - t)/R;
                    %use euclidean distance
                     if min(pdist2(world_Coor,ds)) > 15
                         inbox(y,x,:) = 0;
                     end
                end
            end
        end

    figure;imshow(inbox);


end
```

Published with MATLAB® R2017b

Q2f

```
data = getData([], 'test','list');
ids = data.ids(1:3);

for i = 1:3
   fprintf('In %s:\n',ids{i});
   imdata = getData(ids{i}, 'test', 'left');
   car_ds = imdata.car.ds;
   person_ds = imdata.person.ds;
   cyclist_ds = imdata.cyclist.ds;
   if ~isempty(car_ds)
       for car_idx = 1:size(car_ds,1)
           fprintf('There is a car ');
           if car_ds(car_idx,7) < 0
                fprintf('to you left ');
           else
               fprintf('to you right ');
           end
           distance = norm(car_ds(car_idx,[7 8 9]));
           fprintf('with distances %d\n',distance);
       end

   end
   if ~isempty(person_ds)
       for person_idx = 1:size(person_ds,1)
           fprintf('There is a car ');
           if person_ds(person_idx,7) < 0
                fprintf('to you left ');
           else
               fprintf('to you right ');
           end
           distance = norm(person_ds(person_idx,[7 8 9]));
           fprintf('with distances %d\n',distance);
       end
   end
   if ~isempty(cyclist_ds)
       for cyclist_idx = 1:size(cyclist_ds,1)
           fprintf('There is a car ');
           if cyclist_ds(cyclist_idx,7) < 0
                fprintf('to you left ');
           else
               fprintf('to you right ');
           end
           distance = norm(cyclist_ds(cyclist_idx,[7 8 9]));
           fprintf('with distances %d\n',distance);
       end
   end
 end
```

```
In 004945:
There is a car to you right with distances 5.712012e+01
```

```
There is a car to you right with distances 1.150438e+01
There is a car to you left with distances 4.654811e+01
There is a car to you left with distances 7.571202e+01
In 004964:
There is a car to you right with distances 8.281848e+01
There is a car to you right with distances 4.007096e+01
There is a car to you right with distances 8.276886e+01
There is a car to you left with distances 1.915666e+01
There is a car to you right with distances 1.928514e+01
In 005002:
There is a car to you left with distances 2.929314e+01
There is a car to you left with distances 2.419383e+01
There is a car to you left with distances 2.476850e+01
There is a car to you right with distances 3.397670e+01
There is a car to you right with distances 5.260385e+01
There is a car to you left with distances 2.207100e+01
```

Published with MATLAB® R2017b