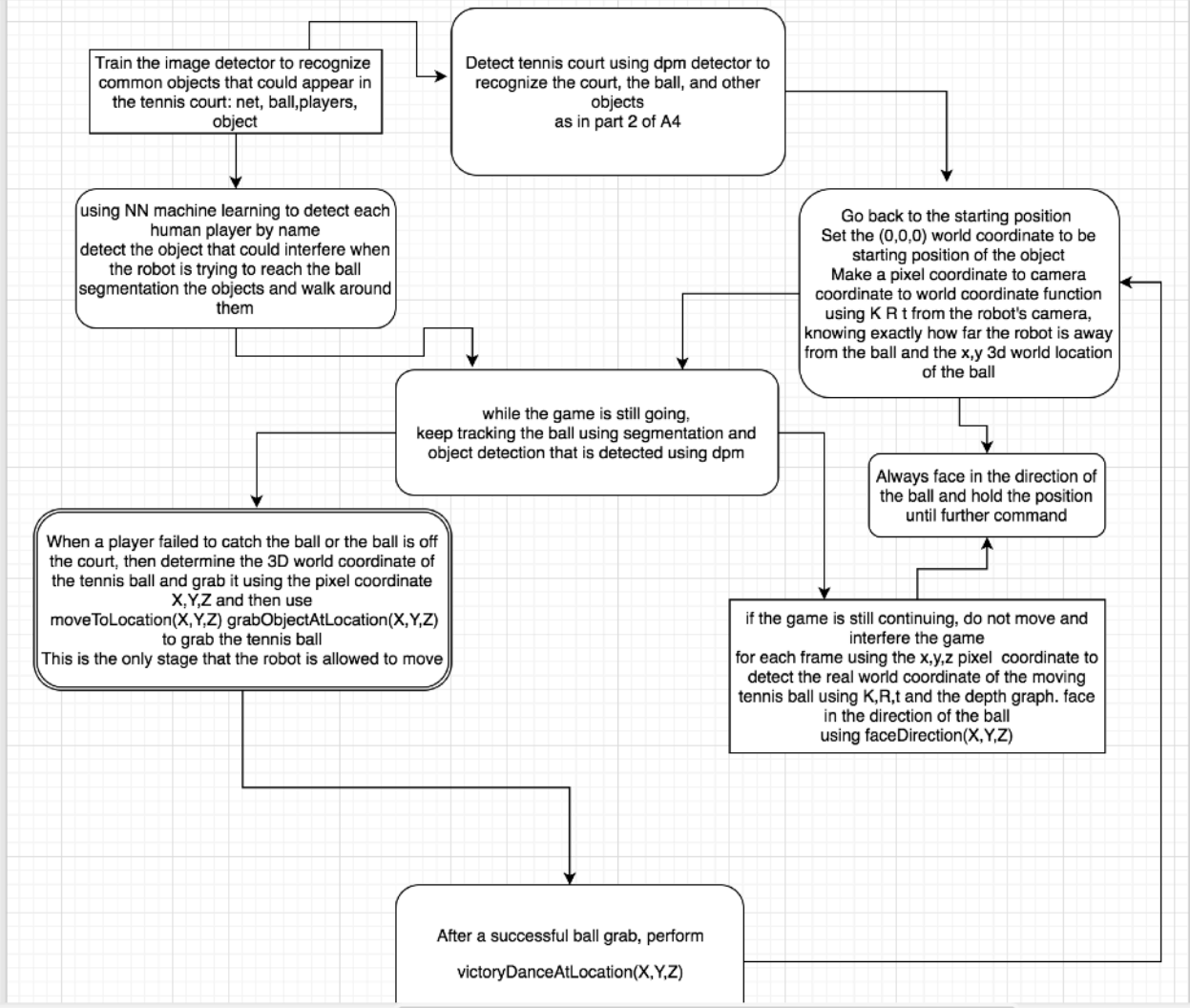
1.
faceDirection(X,Y,Z)
moveToLocation(X,Y,Z)
grabObjectAtLocation(X,Y,Z)
victoryDanceAtLocation(X,Y,Z)
K,[R|t]

a)
Encounter:
Tennis court
Tennis player
Tennis ball
The net
The judges
Objects along the field, such as water, towels, bags
Other ballboy robots

Tranning data:
Images of Tennis ball
Images of the tennis court
Images of human players, judges and etc
Images of other ballboy robots
Images of different objects that could appear along the side of the court

Challenges:
Detecting the tennis ball's projectile
Decide on which robot to use on catching the ball
Accurately catch and toss the tennis ball
Recognize, identify the players
Avoid objects along the tennis court

b)



c)

Train the dpm detector with images of players, court, tennis ball, net, judges chair, common objects

Court_Matrix = dpm.detect(“tennis-court”);

While the robot is turned on:

for each frame:

set the robot’s position to be (0,0,0)

d = getData(frame image,'test','disp');

calib = getData(frame image,'test','calib');

f = calib.f;

T = calib.baseline;

z = (f * T) ./ d.disparity;

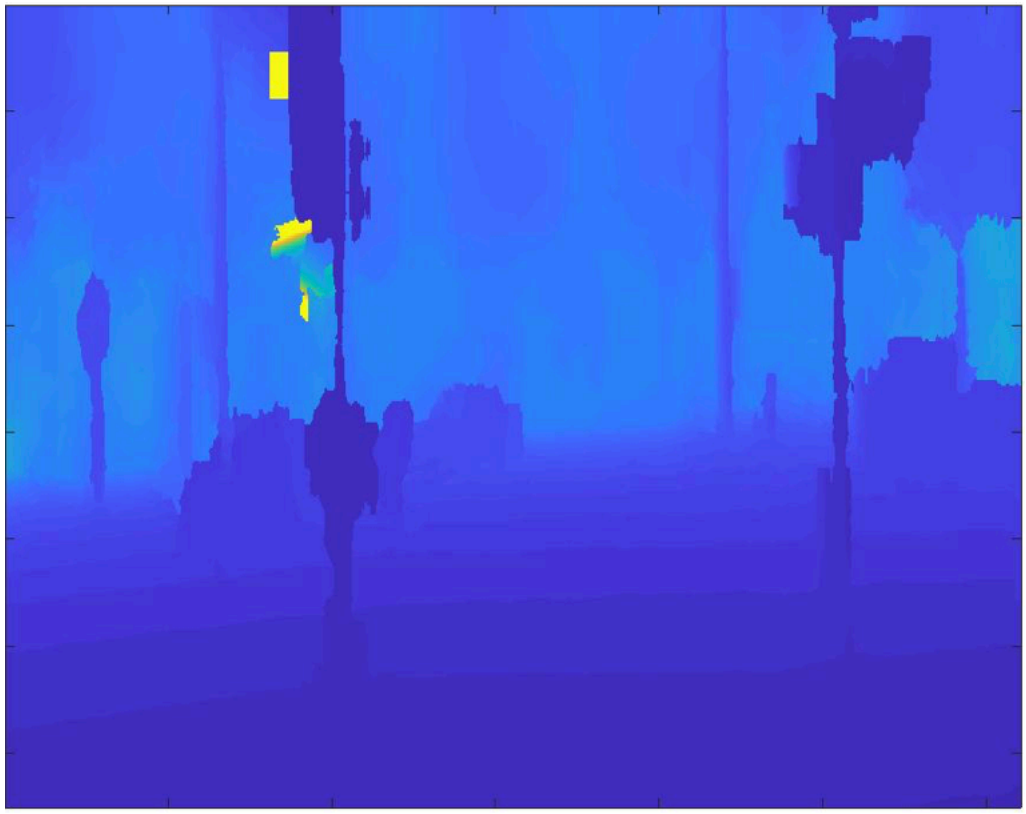
```
/% z is our depth graph
        [pX,pY,pZ] = dpm.getPixelCoordinate.detect(“ball”);
        [X,Y,Z] = pixel_to_world_coordinate(pX,pY,pZ);
/% using KR T from the camera
        faceDirection(X,Y,Z) following the ball
        M = DetectObjects(0,0,0,X,Y,Z);
        /% determine the irrelevant objects in-between 0 0 0 to X Y Z
        M = segmentation(M);
        /% segmentation our matrix M to get a graph of these objects
        avoidObjects(Court_Matrix);
        if game fail to continue:
           while !grabObjectAtLocation(X,Y,Z):
                avoidObjects(M);/% avoid all the objects in-between
                moveToLocation(X,Y,Z);
                grabObjectAtLocation(X,Y,Z);
                VictoryDanceAtLocation(X,Y,Z);
                moveToLocation(0,0,0);
                game continues;
```

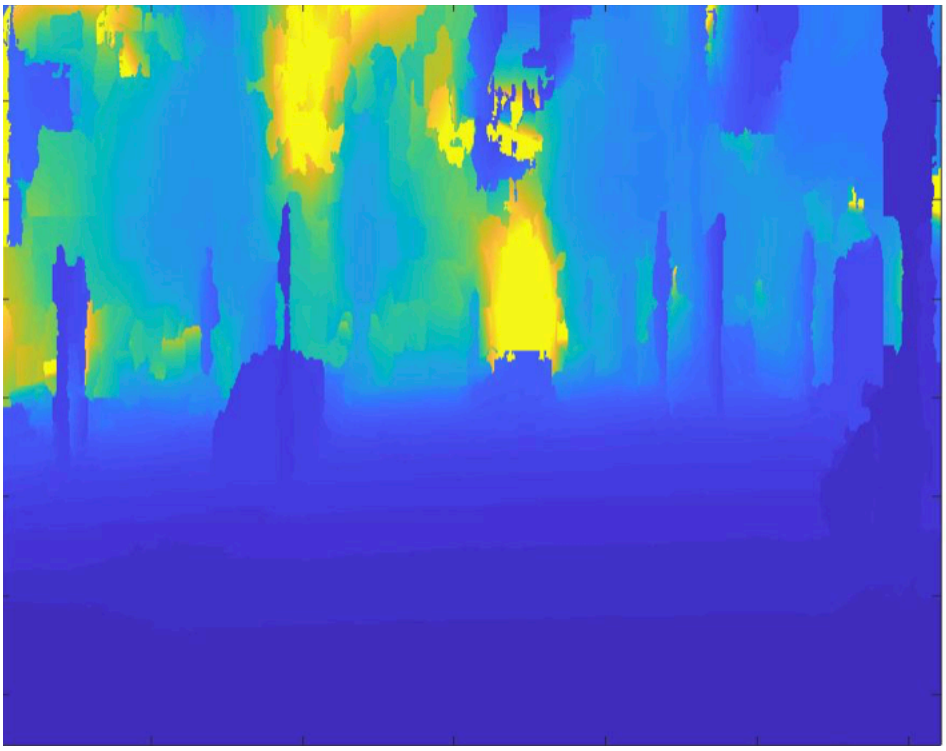
2.

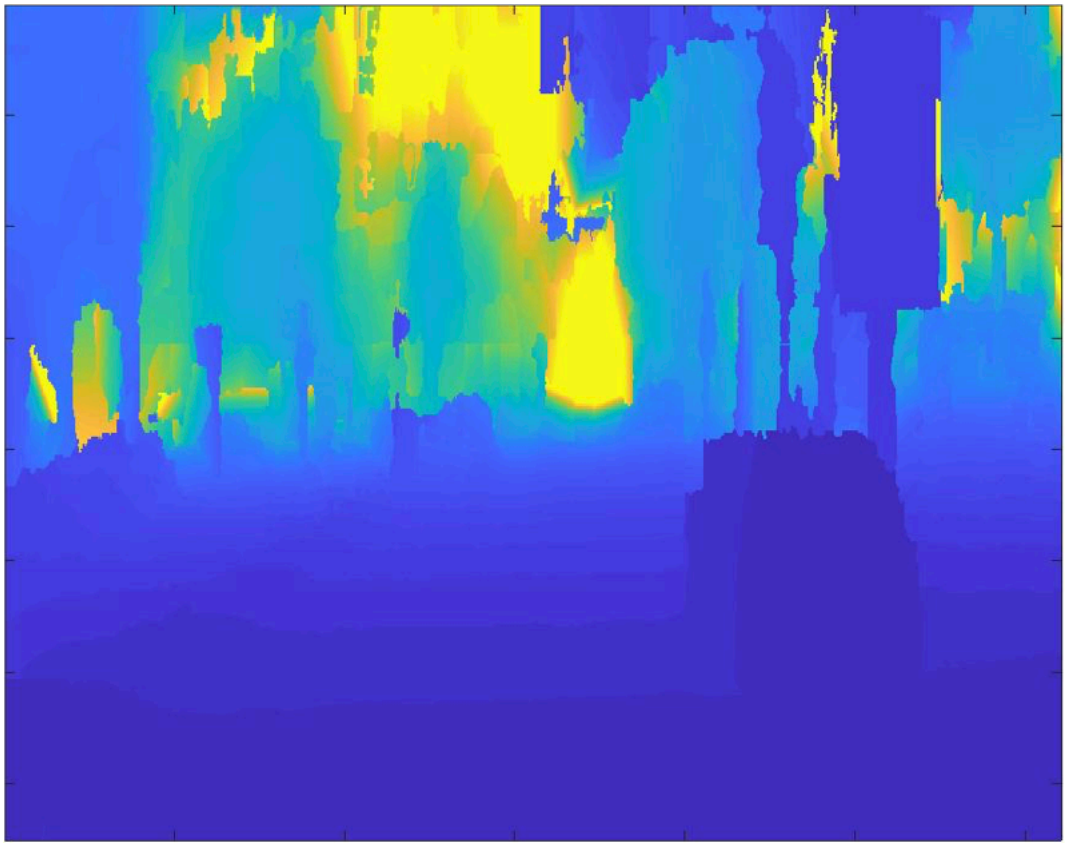
a)

```
function depth
data = getData([],'test','list');
ids = data.ids(1:3);
   for index = 1:3
      disp(1)
      d = getData(ids{index},'test','disp');
      calib = getData(ids{index},'test','calib');
      f = calib.f;
      T = calib.baseline;
      z = (f * T) ./ d.disparity;
      figure;
      imagesc(z, [0,150]);
      hold on;

   end
```

end







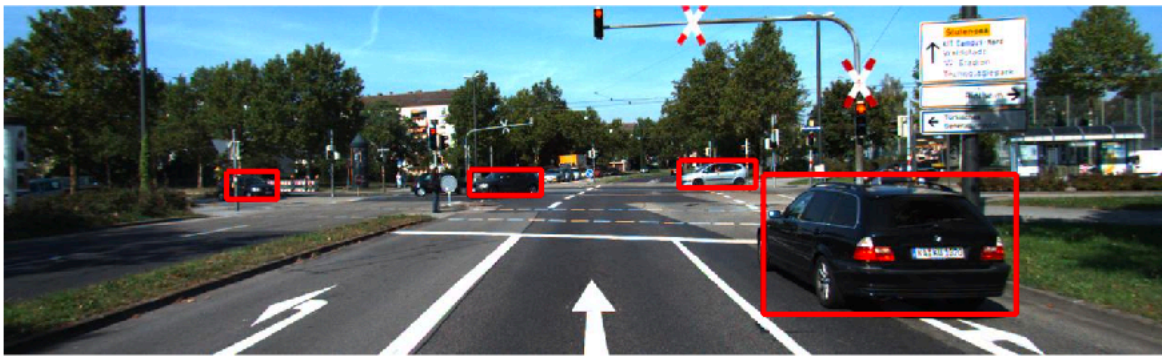
b)
***After couple of trails, detector-cyclist has extremely bad performance on detecting cyclists thus, I choose to use detector-bicycle as mentioned by the instructor in piazza
***detector-person has limited performance in some cases as well, as in 004945 it's hard to detect the person beside the traffic sign

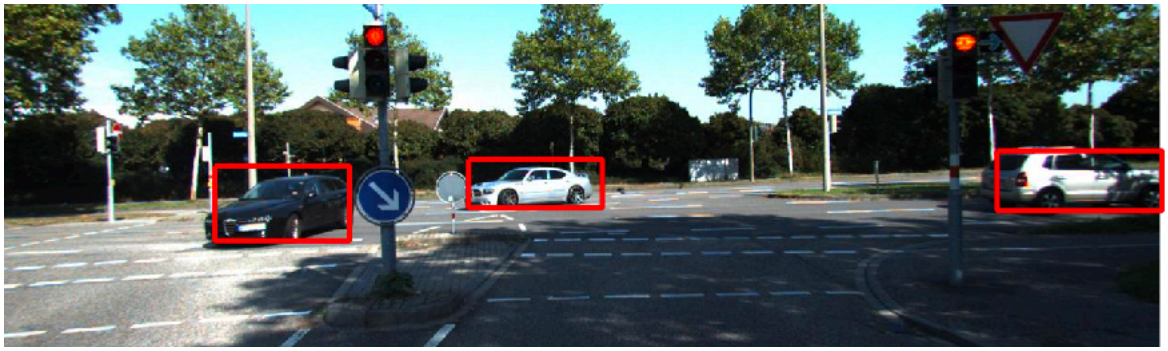
CAR DETECTION
Modifying demo_car.m

```
data = getData([], [], 'detector-car');
model = data.model;
col = 'r';
data = getData([],'test','list');
ids = data.ids(1:3);
for index = 1:3
imdata = getData(ids{index}, '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;
[ds, bs] = imgdetect(imr, model, model.thresh + 0.55); % you may need to reduce the threshold if
you want more detections
e = toc;
fprintf('finished! (took: %0.4f seconds)\n', e);
nms_thresh = 0.3;
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;
    figure;
    showboxesMy(im, reduceboxes(model, bs(top,:)), col);
    hold on;
end;
fprintf('detections:\n');
ds = ds(top, :);

save(char(strcat('./test/results/',ids{index},'_car_detection.txt')),'ds','-ascii');
end
```

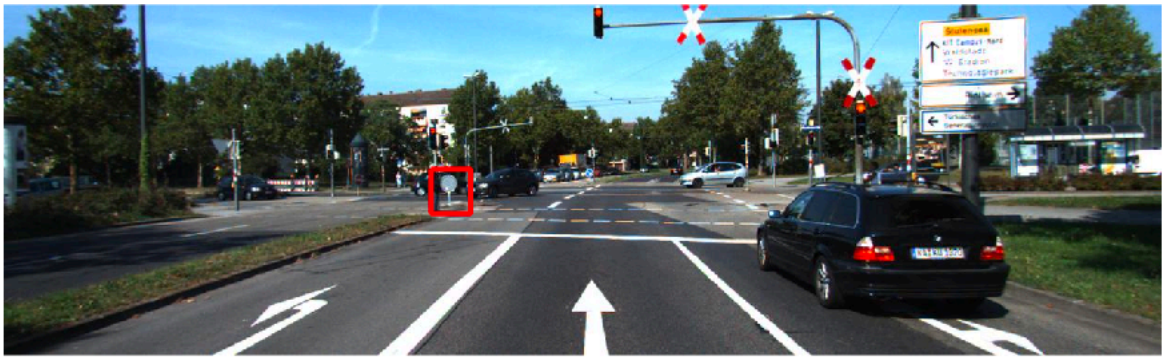
OUTPUT





PERSON DETECTION
Modifying demo_car.m

```
data = getData([], [], 'detector-person');
model = data.model;
col = 'r';
data = getData([],'test','list');
ids = data.ids(1:3);
for index = 1:3
imdata = getData(ids{index}, 'test', 'left');
im = imdata.im;
f = 3;
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 + 1.32); % you may need to reduce the threshold if
you want more detections
e = toc;
fprintf('finished! (took: %0.4f seconds)\n', e);
nms_thresh = 0.5;
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;
    figure;
    showboxesMy(im, reduceboxes(model, bs(top,:)), col);
```



```
    hold on;
end;
fprintf('detections:\n');
ds = ds(top, :);

save(char(strcat('./test/results/',ids{index},'_person_detection.txt')),'ds','-ascii');
end
```

OUTPUT

CYCLIST DETECTION

Modifying demo_car.m

```
data = getData([], [], 'detector-bicycle');
model = data.model;
col = 'r';
data = getData([],'test','list');
ids = data.ids(1:3);
for index = 1:3
imdata = getData(ids{index}, '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;
[ds, bs] = imgdetect(imr, model, model.thresh + 0.12); % you may need to reduce the threshold if
you want more detections
e = toc;
fprintf('finished! (took: %0.4f seconds)\n', e);
nms_thresh = 0.5;
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;
    figure;
    showboxesMy(im, reduceboxes(model, bs(top,:)), col);
    hold on;
end;
fprintf('detections:\n');
ds = ds(top, :);

save(char(strcat('./test/results/',ids{index},'_cyclist_detection.txt')),'ds','-ascii');
end
```

OUTPUT

c)

FOR ALL THE 3 PICTURES

showboxesMy1.m

```
function showboxesMy1(im, boxes,col,out)
% Draw bounding boxes on top of an image.
%   showboxes(im, boxes, out)
%
%   If out is given, a pdf of the image is generated (requires export_fig).
if nargin < 3
    col = 'r';
end;
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
    % if printing, increase the contrast around the boxes
    % by printing a white box under each color box
    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

%       if i ==  13+1 || i == 14+1
%         c = 'k';
%         w = cwidth + 0.5;
%       else
        c = 'w';
%       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 = col; %[160/255 0 0];
      s = '-';
%    elseif i ==  13+1 || i == 14+1
%      c = 'c';
%      s = '--';
    else
      c = 'b';
      s = '-';
    end
    if (col == 'r')
        object = 'Car';
    end
    if (col == 'b')
      object = 'Person';
    end
     if (col == 'c')
      object = 'Cyclist';
    end

    line([x1 x1 x2 x2 x1]', [y1 y2 y2 y1 y1]', 'color', c, 'linewidth', cwidth, 'linestyle', s);
    text(x1, y1,object,'Color','GREEN','FontSize',12);
  end
  end;
  end

% save to pdf
if print
   % requires export_fig from http://www.mathworks.com/matlabcentral/fileexchange/23629-
exportfig
export_fig([out]);
end
```

LABEL DETECTION 004964

Modifying demo_car1.m

```
data = getData([], [], 'detector-car');
model = data.model;
% col = 'r';
```

```
imdata = getData('004964', 'test', 'left');
im = imdata.im;
f = 1.5;
imr = imresize(im,f); % if we resize, it works better for small objects

% start code

% end code

% detect objects
fprintf('running the detector, may take a few seconds...\n');
tic;
[ds1, bs1] = imgdetect(imr, model, model.thresh + 0.55); % you may need to reduce the
threshold if you want more detections
e = toc;
fprintf('finished! (took: %0.4f seconds)\n', e);
nms_thresh = 0.3;
top = nms(ds1, nms_thresh);
if model.type == model_types.Grammar
  bs1 = [ds1(:,1:4) bs1];
end
image(im);
if ~isempty(ds1)
    % resize back
    ds1(:, 1:end-2) = ds1(:, 1:end-2)/f;
    bs1(:, 1:end-2) = bs1(:, 1:end-2)/f;
    showboxesMy1(im,reduceboxes(model, bs1(top,:)) , 'r');
end;
% figure;
% showboxesMy(im, reduceboxes(model, bs(top,:)), 'r');
ds1 = ds1(top, :);
% hold on;
fprintf('detections:\n');
% ds1 = ds1(top, :);
% disp(ds1);

data = getData([], [], 'detector-person');
model = data.model;
% col = 'r';
im = imdata.im;
f = 3;
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;
[ds2, bs2] = imgdetect(imr, model, model.thresh + 1.32); % you may need to reduce the
threshold if you want more detections
e = toc;
fprintf('finished! (took: %0.4f seconds)\n', e);
nms_thresh = 0.5;
top = nms(ds2, nms_thresh);
if model.type == model_types.Grammar
  bs2 = [ds2(:,1:4) bs2];
end
if ~isempty(ds2)
    % resize back
    ds2(:, 1:end-2) = ds2(:, 1:end-2)/f;
    bs2(:, 1:end-2) = bs2(:, 1:end-2)/f;
    showboxesMy1(im,reduceboxes(model, bs2(top,:)), 'b');
end;

fprintf('detections:\n');
ds2 = ds2(top, :);


data = getData([], [], 'detector-bicycle');
model = data.model;
% col = 'r';
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;
[ds3, bs3] = imgdetect(imr, model, model.thresh + 0.12); % you may need to reduce the
threshold if you want more detections
e = toc;
fprintf('finished! (took: %0.4f seconds)\n', e);
nms_thresh = 0.5;
top = nms(ds3, nms_thresh);
if model.type == model_types.Grammar
  bs3 = [ds3(:,1:4) bs3];
end
```

```
if ~isempty(ds3)
    % resize back
    ds3(:, 1:end-2) = ds3(:, 1:end-2)/f;
    bs3(:, 1:end-2) = bs3(:, 1:end-2)/f;
    showboxesMy1(im,reduceboxes(model, bs3(top,:)), 'c');
end;

fprintf('detections:\n');
ds3 = ds3(top, :);

% save( './test/results/005002_detection.txt','ds','-ascii');
```

OUTPUT:

LABEL DETECTION 004945

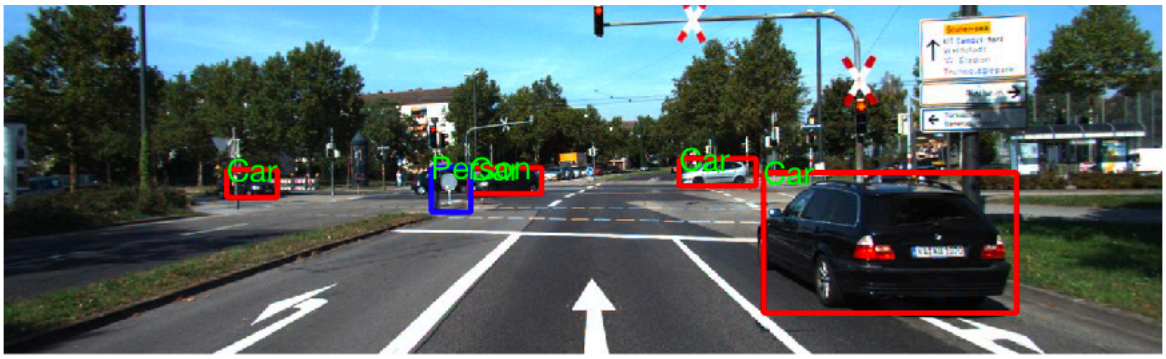
```
Modifying demo_car1.m
data = getData([], [], 'detector-car');
model = data.model;
% col = 'r';

imdata = getData('004945', 'test', 'left');
im = imdata.im;
f = 1.5;
imr = imresize(im,f); % if we resize, it works better for small objects

% start code
```

```
% end code

% detect objects
fprintf('running the detector, may take a few seconds...\n');
tic;
[ds1, bs1] = imgdetect(imr, model, model.thresh + 0.55); % you may need to reduce the
threshold if you want more detections
e = toc;
fprintf('finished! (took: %0.4f seconds)\n', e);
nms_thresh = 0.3;
top = nms(ds1, nms_thresh);
if model.type == model_types.Grammar
  bs1 = [ds1(:,1:4) bs1];
end
image(im);
if ~isempty(ds1)
    % resize back
    ds1(:, 1:end-2) = ds1(:, 1:end-2)/f;
    bs1(:, 1:end-2) = bs1(:, 1:end-2)/f;
    showboxesMy1(im,reduceboxes(model, bs1(top,:)) , 'r');
end;
% figure;
% showboxesMy(im, reduceboxes(model, bs(top,:)), 'r');
ds1 = ds1(top, :);
% hold on;
fprintf('detections:\n');
% ds1 = ds1(top, :);
% disp(ds1);

data = getData([], [], 'detector-person');
model = data.model;
% col = 'r';
im = imdata.im;
f = 3;
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;
[ds2, bs2] = imgdetect(imr, model, model.thresh + 1.32); % you may need to reduce the
threshold if you want more detections
e = toc;
fprintf('finished! (took: %0.4f seconds)\n', e);
```

```
nms_thresh = 0.5;
top = nms(ds2, nms_thresh);
if model.type == model_types.Grammar
  bs2 = [ds2(:,1:4) bs2];
end
if ~isempty(ds2)
    % resize back
    ds2(:, 1:end-2) = ds2(:, 1:end-2)/f;
    bs2(:, 1:end-2) = bs2(:, 1:end-2)/f;
    showboxesMy1(im,reduceboxes(model, bs2(top,:)), 'b');
end;

fprintf('detections:\n');
ds2 = ds2(top, :);


data = getData([], [], 'detector-bicycle');
model = data.model;
% col = 'r';
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;
[ds3, bs3] = imgdetect(imr, model, model.thresh + 0.12); % you may need to reduce the
threshold if you want more detections
e = toc;
fprintf('finished! (took: %0.4f seconds)\n', e);
nms_thresh = 0.5;
top = nms(ds3, nms_thresh);
if model.type == model_types.Grammar
  bs3 = [ds3(:,1:4) bs3];
end
if ~isempty(ds3)
    % resize back
    ds3(:, 1:end-2) = ds3(:, 1:end-2)/f;
    bs3(:, 1:end-2) = bs3(:, 1:end-2)/f;
    showboxesMy1(im,reduceboxes(model, bs3(top,:)), 'c');
end;

fprintf('detections:\n');
```

```
ds3 = ds3(top, :);

% save( './test/results/005002_detection.txt','ds','-ascii');
```

OUTPUT



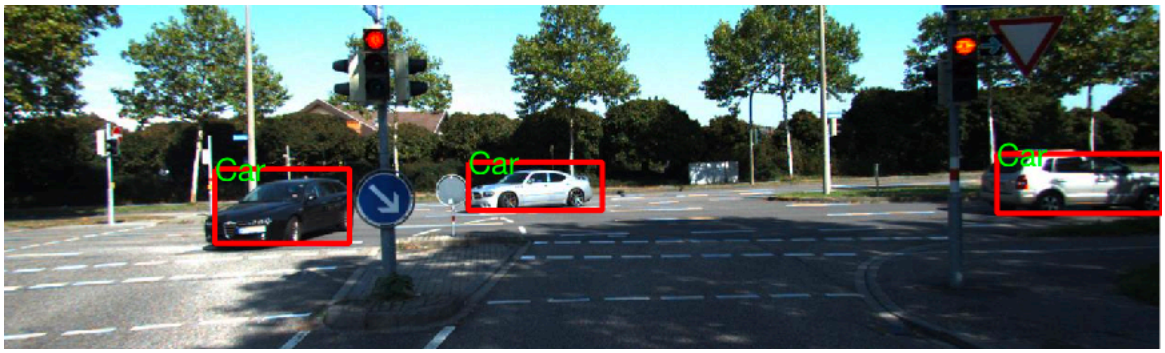
LABEL DETECTION 005002

```
data = getData([], [], 'detector-car');
model = data.model;
% col = 'r';

imdata = getData('005002', 'test', 'left');
im = imdata.im;
f = 1.5;
imr = imresize(im,f); % if we resize, it works better for small objects

% start code

% end code

% detect objects
fprintf('running the detector, may take a few seconds...\n');
tic;
[ds1, bs1] = imgdetect(imr, model, model.thresh + 0.55); % you may need to reduce the
threshold if you want more detections
```

```
e = toc;
fprintf('finished! (took: %0.4f seconds)\n', e);
nms_thresh = 0.3;
top = nms(ds1, nms_thresh);
if model.type == model_types.Grammar
  bs1 = [ds1(:,1:4) bs1];
end
image(im);
if ~isempty(ds1)
    % resize back
    ds1(:, 1:end-2) = ds1(:, 1:end-2)/f;
    bs1(:, 1:end-2) = bs1(:, 1:end-2)/f;
    showboxesMy1(im,reduceboxes(model, bs1(top,:)) , 'r');
end;
% figure;
% showboxesMy(im, reduceboxes(model, bs(top,:)), 'r');
ds1 = ds1(top, :);
% hold on;
fprintf('detections:\n');
% ds1 = ds1(top, :);
% disp(ds1);

data = getData([], [], 'detector-person');
model = data.model;
% col = 'r';
im = imdata.im;
f = 3;
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;
[ds2, bs2] = imgdetect(imr, model, model.thresh + 1.32); % you may need to reduce the
threshold if you want more detections
e = toc;
fprintf('finished! (took: %0.4f seconds)\n', e);
nms_thresh = 0.5;
top = nms(ds2, nms_thresh);
if model.type == model_types.Grammar
  bs2 = [ds2(:,1:4) bs2];
end
if ~isempty(ds2)
    % resize back
```

```
   ds2(:, 1:end-2) = ds2(:, 1:end-2)/f;
   bs2(:, 1:end-2) = bs2(:, 1:end-2)/f;
   showboxesMy1(im,reduceboxes(model, bs2(top,:)), 'b');
end;

fprintf('detections:\n');
ds2 = ds2(top, :);


data = getData([], [], 'detector-bicycle');
model = data.model;
% col = 'r';
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;
[ds3, bs3] = imgdetect(imr, model, model.thresh + 0.12); % you may need to reduce the
threshold if you want more detections
e = toc;
fprintf('finished! (took: %0.4f seconds)\n', e);
nms_thresh = 0.5;
top = nms(ds3, nms_thresh);
if model.type == model_types.Grammar
  bs3 = [ds3(:,1:4) bs3];
end
if ~isempty(ds3)
   % resize back
   ds3(:, 1:end-2) = ds3(:, 1:end-2)/f;
   bs3(:, 1:end-2) = bs3(:, 1:end-2)/f;
   showboxesMy1(im,reduceboxes(model, bs3(top,:)), 'c');
end;

fprintf('detections:\n');
ds3 = ds3(top, :);

% save( './test/results/005002_detection.txt','ds','-ascii');
```

OUPUT



d)

Modifying getData.m

```
function data = getData(imname, imset, whatdata)

% example to run: data = getData('000120', 'train', 'left');
% to load a detector, e.g.: data = getData([],[],'detector-car');

if nargin < 2
    fprintf('run with: data = getData(imname, imset, whatdata);\n');
    fprintf('where:\n');
    fprintf('   imset: "train" or "test"\n');
    fprintf('   whatdata: "list", "left", "right", "calib", "gt-left", "gt-right", "disp"\n')
    fprintf('   "left-plot" and "right-plot" and "disp-plot" will plot the data\n');
    fprintf('   "detector-car" will load a car detector, "detector-person", "detector-cyclist"
similarly\n');
    fprintf('   "superpixels" (if you ran spsstereo code for all images, you also got superpixels)\n');
    fprintf('if the function doesn"t work, please check if globals.m is correctly set\n');
end;

globals;
data = [];

switch whatdata
    case {'car_coor_detect','person_coor_detect','cyclist_coor_detect'}
        cls = strrep(whatdata, '_coor_detect', '');
        files = dir(fullfile('./test/results/', sprintf('%s_%s_detection.txt',imname ,cls)));
        if isempty(files)
            fprintf('file doesn"t exist!\n');
        else
```

```
        data = load(fullfile('./test/results/', files(1).name));
    end;
case {'list'}
    fid = fopen(fullfile(DATA_DIR, imset, [imset '.txt']), 'r+');
    ids = textscan(fid, '%s');
    ids = ids{1};
    fclose(fid);
    data.ids = ids;
case {'left', 'left-plot', 'right', 'right-plot'}
    leftright = strrep(whatdata, '-plot', '');
    imfile = fullfile(DATA_DIR, imset, leftright, sprintf('%s.jpg', imname));
    im = imread(imfile);
    data.im = im;
    if strcmp(whatdata, sprintf('%s-plot', leftright))
        figure('position', [100,100,size(im,2)*0.7,size(im,1)*0.7]);
        subplot('position', [0,0,1,1]);
        imshow(im);
    end;
case {'disp', 'disp-plot'}
    dispdir = fullfile(DATA_DIR, imset, 'results');
    dispfile = fullfile(dispdir, sprintf('%s_left_disparity.png', imname));
    if ~exist(dispfile, 'file')
        fprintf('you haven''t computed disparity yet...\n');
    else
        disparity = imread(dispfile);
        disparity = double(disparity)/256;
    end;
    data.disparity = disparity;
    if strcmp(whatdata, 'disp-plot')
        figure('position', [100,100,size(disparity,2)*0.7,size(disparity,1)*0.7]);
        subplot('position', [0,0,1,1]);
        imagesc(disparity);
        axis equal;
    end;
case 'calib'
    % read internal params
    calib_dir = fullfile(DATA_DIR, imset, 'calib');
    [~, ~, calib] = loadCalibration(fullfile(calib_dir, sprintf('%s.txt', imname)));
    [Kl,~,tl] = KRt_from_P(calib.P_rect{3});  % left camera
    [~,~,tr] = KRt_from_P(calib.P_rect{4});  % right camera
    f = Kl(1,1);
    baseline = abs(tr(1)-tl(1));   % distance between cams
    data.f = f;
```

```
    data.baseline = baseline;
    data.K = Kl;
    data.P_left = calib.P_rect{3};
    data.P_right = calib.P_rect{4};
  case {'gt-left', 'gt-right', 'gt-left-plot', 'gt-right-plot', 'gt-left-plot-3d', 'gt-right-plot-3d'}
    leftright = strrep(strrep(whatdata, 'gt-', ''), '-plot', '');
    leftright = strrep(leftright, '-3d', '');
    label_dir = fullfile(DATA_DIR, imset, sprintf('gt_%s', leftright));
    img_idx = str2num(imname);
    objects = readLabels(label_dir,img_idx);
    data.objects = objects;
    calibdata = getData(imname, imset, 'calib');
    P = calibdata.(sprintf('P_%s', leftright));
    for i = 1 : length(objects)
      [corners_2D,~,corners_3D] = computeBox3D(objects(i),P);
      orientation3D = computeOrientation3D(objects(i),P);
      objects(i).corners3D = corners_3D;
      objects(i).corners2D = corners_2D;
      objects(i).orientation3D = orientation3D;
    end;
    if strcmp(leftright, 'left')
      labels_file = fullfile(label_dir, [imname '.png']);
      seg = double(imread(labels_file));
      data.gt_seg = seg;
    end;
    which = strrep(whatdata, sprintf('gt-%s-plot', leftright), '');

    if strcmp(whatdata, sprintf('gt-%s-plot%s', leftright, which))
      imgdata = getData(imname, imset, leftright);
      seg = [];
      if isfield(data, 'gt_seg')
         seg = data.gt_seg;
      end;
      if isempty(which)
        plotGT(imgdata.im, objects, seg, '2d')
      else
        plotGT(imgdata.im, objects, seg, '3d')
      end;
      data.im = imgdata.im;
    end;
  case {'detector-car', 'detector-person', 'detector-pedestrian', 'detector-cyclist','detector-bicycle'}
    cls = strrep(whatdata, 'detector-', '');
    files = dir(fullfile(DETECTOR_DIR, sprintf('%s_final*.mat', cls)));
```

```
        if isempty(files)
            fprintf('file doesn''t exist!\n');
        else
            data = load(fullfile(DETECTOR_DIR, files(1).name));
        end;
    case {'superpixels'}
        dispdir = fullfile(DATA_DIR, imset, 'results');
        spfile = fullfile(dispdir, sprintf('%s_segment.png', imname));
        spim = [];
        if ~exist(spfile, 'file')
            fprintf('you haven''t ran spsstereo code yet...\n');
        else
            spim = imread(spfile);
            spim = double(spim);
        end;
        data.spim = spim;
    otherwise
        disp('unknown data type, try again');

end;
```

find_coordinates.m

```
function find_coordinates
data = getData([],'test','list');
ids = data.ids(1:3);
for index = 1:3
%     disp(1);
    d = getData(ids{index},'test','disp');
    calib = getData(ids{index},'test','calib');
    f = calib.f;
    [K, R, t] = KRt_from_P(calib.P_left);
    T = calib.baseline;
    camera_z = (f * T) ./ d.disparity;
    world_x = zeros(size(camera_z));
    world_y = zeros(size(camera_z));
    world_z = zeros(size(camera_z));
        R_t = [1 0 0 0.0598;0 1 0 -0.0004; 0 0 1 0.0027; 0 0 0 1];
%     adding t to R since R is just 1 0 0;0 1 0; 0 0 1;
%  t is [ 0.0598
%    -0.0004
%     0.0027
```

```
%       1 ]
    [x, y] = meshgrid(0:size(camera_z,2)-1, 0:size(camera_z,1)-1);
    camera_x = (camera_z/f).*(x - K(1,3));
    camera_y = (camera_z/f).*(y - K(2,3));
    for i_ = 1: size(camera_x,1)
        for j_ = 1: size(camera_x,2)
            A = inv(R_t)* [camera_x(i_,j_);camera_y(i_,j_);camera_z(i_,j_);1];
            world_x(i_,j_) = A(1,1);
            world_y(i_,j_) = A(2,1);
            world_z(i_,j_) = A(3,1);
        end
    end
    ans = [];
    car_ds = getData(ids{index}, [], 'car_coor_detect');
    for w = 1:size(car_ds, 1)
        x1 = car_ds(w,1);
        y1 = car_ds(w,2);
        x2 = car_ds(w,3);
        y2 = car_ds(w,4);

        x_ = (x1 + x2)/2;
        y_ = (y1 + y2)/2;
        x_ = round(x_);
        y_ = round(y_);

%         sum = 0;
%         c = 0;
%         if round(x2) > size(z,1)
%             x2 = size(z,1);
% %         disp(round(x1));
% %         disp(round(x2));
%         for i = round(x1):round(x2)
%             for j = round(y1):round(y2)
% % %                 disp(i);
% % %                 disp(j);
%                 sum = sum + z(j,i);
%                 c = c + 1.0;
%             end
%         end


        ans = [ans;world_x(y_,x_) world_y(y_,x_) world_z(y_,x_)];
```

```
    end
    save(char(strcat('./test/results/',ids{index},'_3dcar_coordinates.txt')),'ans','-ascii');

    ans = [];
    person_ds = getData(ids{index}, [], 'person_coor_detect');
    for w = 1:size(person_ds, 1)
        x1 = person_ds(w,1);
        y1 = person_ds(w,2);
        x2 = person_ds(w,3);
        y2 = person_ds(w,4);

        x_ = (x1 + x2)/2;
        y_ = (y1 + y2)/2;
        x_ = round(x_);
        y_ = round(y_);

        ans = [ans;world_x(y_,x_) world_y(y_,x_) world_z(y_,x_)];

    end
    save(char(strcat('./test/results/',ids{index},'_3dperson_coordinates.txt')),'ans','-ascii');

    ans = [];
    cyclist_ds = getData(ids{index}, [], 'cyclist_coor_detect');
    for w = 1:size(cyclist_ds, 1)
        x1 = cyclist_ds(w,1);
        y1 = cyclist_ds(w,2);
        x2 = cyclist_ds(w,3);
        y2 = cyclist_ds(w,4);

        x_ = (x1 + x2)/2;
        y_ = (y1 + y2)/2;
        x_ = round(x_);
        y_ = round(y_);

        ans = [ans;world_x(y_,x_) world_y(y_,x_) world_z(y_,x_)];

    end
    save(char(strcat('./test/results/',ids{index},'_3dcyclist_coordinates.txt')),'ans','-ascii');
end
end
```

e)
function segmentation

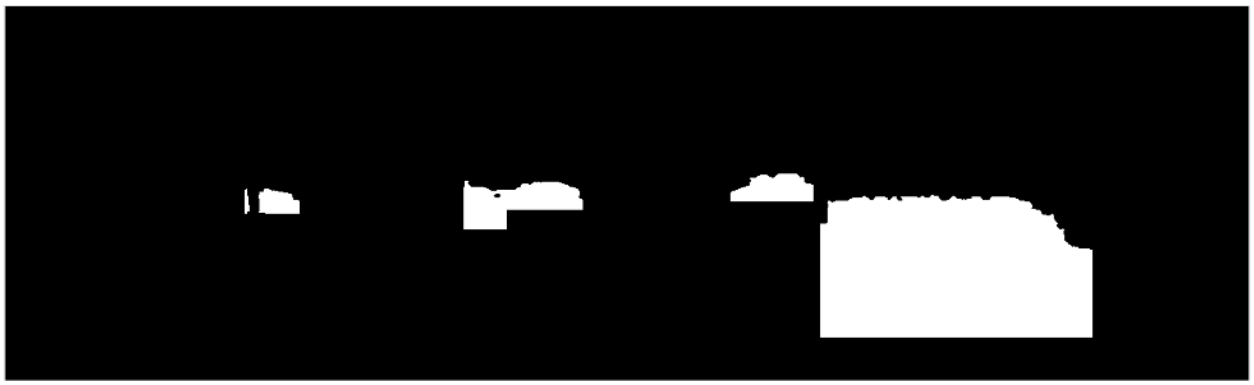
```
data = getData([],'test','list');
ids = data.ids(1:3);
for index = 1:3
%     disp(1);
    d = getData(ids{index},'test','disp');
    calib = getData(ids{index},'test','calib');
    f = calib.f;
    T = calib.baseline;
    z = (f * T) ./ d.disparity;
    camera_z = z;
    [K, R, t] = KRt_from_P(calib.P_left);
    car_ds = getData(ids{index}, [], 'car_coor_detect');
    person_ds = getData(ids{index}, [], 'person_coor_detect');
    cyclist_ds = getData(ids{index}, [], 'cyclist_coor_detect');
    car_3dlocation = getData(ids{index}, [], 'car_3dlocation_coor');
    person_3dlocation = getData(ids{index}, [], 'person_3dlocation_coor');
    cyclist_3dlocation = getData(ids{index}, [], 'cyclist_3dlocation_coor');
%     disp(car_ds);
        R_t = [1 0 0 0.0598;0 1 0 -0.0004; 0 0 1 0.0027; 0 0 0 1];
%     adding t to R since R is just 1 0 0;0 1 0; 0 0 1;
%  t is [ 0.0598
%    -0.0004
%     0.0027
%       1 ]
    [x, y] = meshgrid(0:size(camera_z,2)-1, 0:size(camera_z,1)-1);
    camera_x = (camera_z/f).*(x - K(1,3));
    camera_y = (camera_z/f).*(y - K(2,3));
    world_x = zeros(size(camera_z));
    world_y = zeros(size(camera_z));
    world_z = zeros(size(camera_z));
    for i_ = 1: size(camera_x,1)
        for j_ = 1: size(camera_x,2)
            A = inv(R_t)* [camera_x(i_,j_);camera_y(i_,j_);camera_z(i_,j_);1];
            world_x(i_,j_) = A(1,1);
            world_y(i_,j_) = A(2,1);
            world_z(i_,j_) = A(3,1);
        end
    end
    M = zeros(375,1242);
    for w = 1:size(car_ds, 1)
%         disp(1);
        x1 = car_ds(w,1);
        y1 = car_ds(w,2);
```

```
        x2 = car_ds(w,3);
        y2 = car_ds(w,4);

        x_ = car_3dlocation(w,1);
        y_ = car_3dlocation(w,2);
        z_ = car_3dlocation(w,3);
        if round(x2) > size(z,2)
            x2 = size(z,2);
        end
        if round(y2) > size(z,1)
            y2 = size(z,1);
        end
%       sum = 0;
%       c = 0;
%       if round(x2) > size(z,1)
%           x2 = size(z,1);
%       disp(round(x1));
%       disp(round(x2));
       for i = round(y1):round(y2)
%           disp(1);
           for j = round(x1):round(x2)
               if pdistance(world_x(i,j),world_y(i,j),world_z(i,j),x_,y_,z_) < 9
                   M(i,j) = 1;
               end
           end
       end

    for w = 1:size(person_ds, 1)
%       disp(1);
        x1 = person_ds(w,1);
        y1 = person_ds(w,2);
        x2 = person_ds(w,3);
        y2 = person_ds(w,4);

        x_ = person_3dlocation(w,1);
        y_ = person_3dlocation(w,2);
        z_ = person_3dlocation(w,3);
        if round(x2) > size(z,2)
            x2 = size(z,2);
        end
        if round(y2) > size(z,1)
            y2 = size(z,1);
        end
```

```
%       sum = 0;
%       c = 0;
%       if round(x2) > size(z,1)
%          x2 = size(z,1);
%       disp(round(x1));
%       disp(round(x2));
      for i = round(y1):round(y2)
%           disp(1);
         for j = round(x1):round(x2)
            if pdistance(world_x(i,j),world_y(i,j),world_z(i,j),x_,y_,z_) < 9
               M(i,j) = 1;
            end
         end
      end

    for w = 1:size(cyclist_ds, 1)
%       disp(1);
      x1 = cyclist_ds(w,1);
      y1 = cyclist_ds(w,2);
      x2 = cyclist_ds(w,3);
      y2 = cyclist_ds(w,4);

      x_ = cyclist_3dlocation(w,1);
      y_ = cyclist_3dlocation(w,2);
      z_ = cyclist_3dlocation(w,3);
      if round(x2) > size(z,2)
          x2 = size(z,2);
      end
      if round(y2) > size(z,1)
          y2 = size(z,1);
      end
%       sum = 0;
%       c = 0;
%       if round(x2) > size(z,1)
%          x2 = size(z,1);
%       disp(round(x1));
%       disp(round(x2));
      for i = round(y1):round(y2)
%           disp(1);
         for j = round(x1):round(x2)
            if pdistance(world_x(i,j),world_y(i,j),world_z(i,j),x_,y_,z_) < 9
               M(i,j) = 1;
            end
```

```
            end
        end

    end
%     figure;
    end
    end
    figure;
      imshow(M);
end
end
```

OUPUT



f)

```
function segmentation
data = getData([],'test','list');
ids = data.ids(1:3);
for index = 1:3
%     disp(1);
    fprintf("For picture%s \n", ids{index});
    da = getData(ids{index},'test','disp');
    calib = getData(ids{index},'test','calib');
    f = calib.f;
    T = calib.baseline;
    z = (f * T) ./ da.disparity;
    camera_z = z;
    [K, R, t] = KRt_from_P(calib.P_left);
    car_ds = getData(ids{index}, [], 'car_coor_detect');
    person_ds = getData(ids{index}, [], 'person_coor_detect');
    cyclist_ds = getData(ids{index}, [], 'cyclist_coor_detect');
    car_3dlocation = getData(ids{index}, [], 'car_3dlocation_coor');
    person_3dlocation = getData(ids{index}, [], 'person_3dlocation_coor');
    cyclist_3dlocation = getData(ids{index}, [], 'cyclist_3dlocation_coor');
%     disp(car_ds);
       R_t = [1 0 0 0.0598;0 1 0 -0.0004; 0 0 1 0.0027; 0 0 0 1];
%     adding t to R since R is just 1 0 0;0 1 0; 0 0 1;
%  t is [ 0.0598
%    -0.0004
%     0.0027
%       1 ]
   [x, y] = meshgrid(0:size(camera_z,2)-1, 0:size(camera_z,1)-1);
   camera_x = (camera_z/f).*(x - K(1,3));
    camera_y = (camera_z/f).*(y - K(2,3));
    world_x = zeros(size(camera_z));
    world_y = zeros(size(camera_z));
    world_z = zeros(size(camera_z));
    for i_ = 1: size(camera_x,1)
        for j_ = 1: size(camera_x,2)
            A = inv(R_t)* [camera_x(i_,j_);camera_y(i_,j_);camera_z(i_,j_);1];
            world_x(i_,j_) = A(1,1);
            world_y(i_,j_) = A(2,1);
            world_z(i_,j_) = A(3,1);
        end
    end
    M = zeros(375,1242);
    for w = 1:size(car_ds, 1)
```

```
%       disp(1);
      x1 = car_ds(w,1);
      y1 = car_ds(w,2);
      x2 = car_ds(w,3);
      y2 = car_ds(w,4);

      x_ = car_3dlocation(w,1);
      y_ = car_3dlocation(w,2);
      z_ = car_3dlocation(w,3);

      x__ = round((x1 + x2)/2.0);
      y__ = round((y1 + y2)/2.0);
      if round(x2) > size(z,2)
          x2 = size(z,2);
      end
      if round(y2) > size(z,1)
          y2 = size(z,1);
      end
%       sum = 0;
%       c = 0;
%       if round(x2) > size(z,1)
%           x2 = size(z,1);
%       disp(round(x1));
%       disp(round(x2));
%        for i = round(y1):round(y2)
% %            disp(1);
%            for j = round(x1):round(x2)
%               if pdistance(world_x(i,j),world_y(i,j),world_z(i,j),x_,y_,z_) < 9
%                  M(i,j) = 1;
%               end
%            end
%        end
%       disp(x__);
%       disp(y__);
      d = norm([camera_x(y__,x__), camera_y(y__,x__), camera_z(y__,x__)]);
%       disp(d);
      % this is the distance of object to driver
      X1 = camera_x(y__,x__);
      if X1 >= 0, txt = "to your right";
      else txt = "to your left";

      end
      fprintf("There is a %s %0.1f meters %s \n", "car", abs(X1), txt);
```

```
        fprintf("It is %0.1f meters away from you \n", d);
    end

     for w = 1:size(person_ds, 1)
%         disp(1);
       x1 = person_ds(w,1);
       y1 = person_ds(w,2);
       x2 = person_ds(w,3);
       y2 = person_ds(w,4);

       x_ = person_3dlocation(w,1);
       y_ = person_3dlocation(w,2);
       z_ = person_3dlocation(w,3);
       if round(x2) > size(z,2)
            x2 = size(z,2);
       end
       if round(y2) > size(z,1)
            y2 = size(z,1);
       end
%         sum = 0;
%         c = 0;
%         if round(x2) > size(z,1)
%             x2 = size(z,1);
%         disp(round(x1));
%         disp(round(x2));
%          for i = round(y1):round(y2)
% %              disp(1);
%              for j = round(x1):round(x2)
%                  if pdistance(world_x(i,j),world_y(i,j),world_z(i,j),x_,y_,z_) < 9
%                      M(i,j) = 1;
%                  end
%              end
%          end
%         disp(x1);
%         disp(x2);
       x__ = round((x1 + x2)/2.0);
%         disp(x__);
       y__ = round((y1 + y2)/2.0);
%         disp(y__);
%         disp("happy");
       X1 = camera_x(y__,x__);

        d = norm([camera_x(y__,x__), camera_y(y__,x__), camera_z(y__,x__)]);
```

```
Trial>> segmentation
For picture004945
There is a car 10.3 meters to your right
It is 48.8 meters away from you
There is a car 3.2 meters to your right
It is 7.6 meters away from you
There is a car 3.4 meters to your left
It is 34.9 meters away from you
There is a car 22.1 meters to your left
It is 51.6 meters away from you
There is a person 5.1 meters to your left
It is For picture004964
There is a car 2.9 meters to your right
It is 31.8 meters away from you
There is a car 12.9 meters to your right
It is 30.7 meters away from you
There is a car 6.3 meters to your left
It is 19.0 meters away from you
There is a car 10.4 meters to your right
It is 15.8 meters away from you
There is a person 5.3 meters to your left
It is There is a cyclist 8.4 meters to your right
It is For picture005002
There is a car 1.3 meters to your left
It is 24.5 meters away from you
There is a car 6.7 meters to your left
It is 16.9 meters away from you
There is a car 13.6 meters to your right
It is 22.6 meters away from you
```

```
    % this is the distance of object to driver
    if X1 >= 0, txt = "to your right";
    else txt = "to your left";

    end
    fprintf("There is a %s %0.1f meters %s \n", "person", abs(X1), txt);
    fprintf("It is %0.1 meters away from you \n", d);

end
```

```
    for w = 1:size(cyclist_ds, 1)
%       disp(1);
      x1 = cyclist_ds(w,1);
      y1 = cyclist_ds(w,2);
      x2 = cyclist_ds(w,3);
      y2 = cyclist_ds(w,4);

      x_ = cyclist_3dlocation(w,1);
      y_ = cyclist_3dlocation(w,2);
      z_ = cyclist_3dlocation(w,3);
      if round(x2) > size(z,2)
          x2 = size(z,2);
      end
      if round(y2) > size(z,1)
          y2 = size(z,1);
      end
%       sum = 0;
%       c = 0;
%       if round(x2) > size(z,1)
%           x2 = size(z,1);
%       disp(round(x1));
%       disp(round(x2));
%        for i = round(y1):round(y2)
% %             disp(1);
%            for j = round(x1):round(x2)
%               if pdistance(world_x(i,j),world_y(i,j),world_z(i,j),x_,y_,z_) < 9
%                  M(i,j) = 1;
%               end
%            end
%        end
      x__ = round((x1 + x2)/2.0);
      y__ = round((y1 + y2)/2.0);
       d = norm([camera_x(y__,x__), camera_y(y__,x__), camera_z(y__,x__)]);
       % this is the distance of object to driver
       X1 = camera_x(y__,x__);
       if X1 >= 0, txt = "to your right";
       else txt = "to your left";
       end
       fprintf("There is a %s %0.1f meters %s \n", "cyclist",abs(X1), txt);
       fprintf("It is %0.1 meters away from you \n", d);


   end
```

```
%     figure;
   end
   end
%     figure;
%        imshow(M);
```

OUTPUT

3.
Bonus.m

```
%% prepare workspace
clear variables;
close all;
clc;

%% load camera params and data
camera_params
load('rgbd.mat');
%% Construct camera calibration matrix
K = [fx_d  0     px_d;
    0     fy_d  py_d;
    0     0     1];

 % We are working with camera coordinates so we
 % don't need rotation or 3D translation
 P = K * eye(3, 4);

%% Construct grid of image coordinates for each pixel
[x, y] = meshgrid(0:size(im,2)-1, 0:size(im,1)-1);
y = size(im,1)-1-y;

%% Solve for real-world X and Y in camera coordinates
Z = depth(1:480, 1:640);
X = (Z / fx_d) .* (x - px_d);
Y = (Z / fy_d) .* (y - py_d);

surf(X,Z,Y,Z,'EdgeColor','none');
xlabel('x');
```

```
ylabel('z');
zlabel('y');

%% Compute coordinates of all objects
nobj = 7;
Xobj = zeros(1, nobj);
Yobj = zeros(1, nobj);
Zobj = zeros(1, nobj);
boxZ = zeros(nobj,2);
boxX = zeros(nobj,2);
boxY = zeros(nobj,2);

for i=1:nobj
   [y,x] = find(labels==i);
   y = size(im,1)-1-y;
   Z = depth(labels==i);
   X = (Z / fx_d) .* (x - px_d);
   Y = (Z / fy_d) .* (y - py_d);

   Xobj(i) = mean(X);
   Yobj(i) = mean(Y);
   Zobj(i) = mean(Z);

   boxZ(i,1) = min(Z);
   boxZ(i,2) = max(Z);
   boxX(i,1) = min(X);
   boxX(i,2) = max(X);
   boxY(i,1) = min(Y);
   boxY(i,2) = max(Y);
end

for i=1:nobj
   for j=1:nobj
      jZ_mean = (boxZ(j,1) + boxZ(j,2))/2;
      imin_Z = boxZ(i,1);
      imax_Z = boxZ(i,2);
      jX_mean = (boxX(j,1) + boxX(j,2))/2;
      imin_X = boxX(i,1);
      imax_X = boxX(i,2);
      jmin_Y = boxY(j,1);
      jmax_Y = boxY(j,2);
      imax_Y = boxY(i,2);
      imin_Y = boxY(i,1);
```

```
%       so for j to lie on i, j must be having a higher y coordinate than
%       i, and j's average must lie inside the x and z coordinates of i
%       object and the j's min Y has to be smaller than i's max Y
%       j's max Y has to be bigger than i's min Y
      if imin_Z<= jZ_mean && jZ_mean<=imax_Z && imin_X <= jX_mean && jX_mean<=
imax_X && (Yobj(j) > Yobj(i))&& jmin_Y < imax_Y && jmax_Y > imin_Y
          fprintf("%d Object is sitting on top %d object \n", j, i);
      end
   end
end
```

OUTPUT

```
1 Object is sitting on top 6 object
5 Object is sitting on top 6 object
3 Object is sitting on top 7 object
```