ROBUST VEHICLE DETECTION FROM PARKING LOT IMAGES

Siming Liu

Dec 20, 2005

Boston University

Department of Electrical and Computer Engineering

Technical report No. ECE-2005-06

BOSTON
UNIVERSITY

ROBUST VEHICLE DETECTION FROM PARKING LOT IMAGES

Siming Liu

Boston University
Department of Electrical and Computer Engineering
8 Saint Mary's Street
Boston, MA 02215
www.bu.edu/ece

Dec 20, 2005

Technical report No. ECE-2005-06

Summary

The objective of this project is to develop an algorithm for the detection of vehicles in parking lot images. With the proposed method, each image region corresponding to a parking cell is segmented for initialization, and temporal differencing is used for state tracking. Therefore, reference images taken in vacant state are not needed, hence it can be easily applied to parking lots in continuous service. The proposed method was tested on an actual outdoor parking lot under various weather conditions. The results confirmed the efficiency of the algorithm.

Contents

List of figures and tables

1 Introduction

Parking vacancy monitoring can play an important role in efficient use of parking space and guiding cars to vacant cells. It is particularly true for parking lots with busy traffic such as those located in big cities, airports, or near large halls. Monitoring techniques can be divided into two categories: the first approach is to estimate occupancy of an entire parking lot, for example by counting incoming vehicles, while the second type is to check the presence of a vehicle in each parking cell. Numerous methods have been employed to monitor individual parking cells using ultrasonic or magnetic sensors placed within each cell, or cameras placed above cells. The first method requires many of sensors, while the second method requires only one camera since it can cover relatively wide area.

Regarding image-based vehicle detection, various techniques have been developed such as motion tracking using temporal subtraction, comparison with reference images using normalized principle component of feature characteristics, or path tracking. Methods based on reference images are not really practicable because acquisition and update of reference images are difficult with a parking lot in service or with varying lighting conditions, weather patterns (outdoor parking), etc. In tracking based on temporal subtraction, reference images are not required, and also tracking can be performed even at nighttime by following headlights. However, relatively high processing speed is required to deal with cars moving fast. In addition, nighttime tracking will be impossible with headlights off. There are many other methods that do not rely on reference images or path tracking, for example, car detection through hidden white lines, or using typical shape of car elements. When using white lines for vehicle detection, images must be captured from such a view angle that some white lines are hidden by parked vehicles, which is complicated in terms of restriction on the placement of cameras against white lines. With respect to extraction of car element features, a large number of pixels per vehicle are needed, hence not many cells can be covered by a single camera.

This project aims at the development of a technique providing occupancy check for each individual parking cell. The following assumptions are being made: parking lot in continuous service, simple configuration of the processing system, no need for reference images or path tracking, robustness under varying weather conditions.

The method presented in this study uses grayscale images to detect vehicle presence. In order to determine the initial state of a parking cell, its image is segmented using gray levels, and the obtained segments are analyzed for vehicle presence. The change of parking cell status (occupied versus empty) is detected by means of temporal frame difference and double-checked by segmentation and segment analysis. The efficacy of the proposed algorithm is confirmed experimentally by application to an actual outdoor parking lot under various weather conditions.

2 Car presence detection

A view of typical camera placement in the considered monitoring system is shown in Fig.1. Images of the parking lot are taken through a wireless camera placed, for example, on the roof of a neighboring building or on a high pole. In this way, the locations of all parking cells can be assumed known. Occasional pedestrians or incoming/outgoing vehicles are not supposed to stay in one place for a long time, and hence are neglected in the processing. Since the proposed algorithm is intended for outdoor parking lots, the detection must be stable under various weather conditions. Nighttime illumination is assumed sufficient for image acquisition.

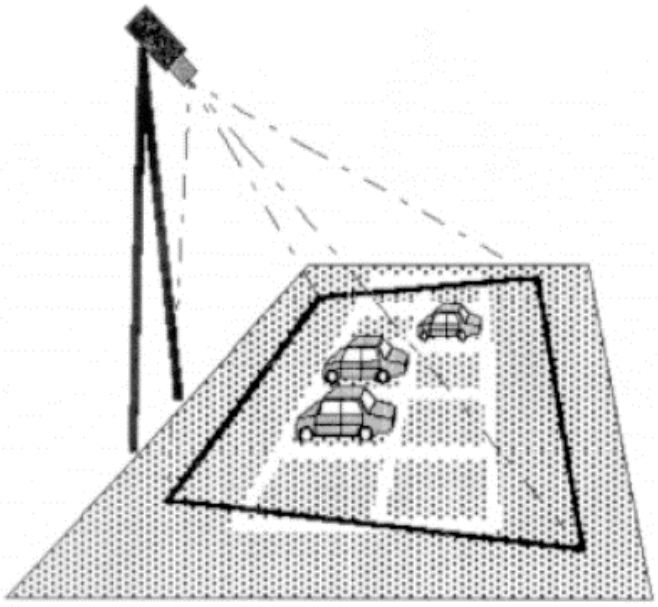

Fig. 1. Parking lot monitoring system.

2.1 Principle of the detection

Vehicles have diverse shapes (e.g., sedans, minivans, trucks) and colors. On the other hand, the surface of a parking lot does not offer any of those varieties, though it might contain white lines, casual shadows, or water puddles. Therefore, when detecting a car, one should use features common to all vehicles while distinct from the aforementioned parking surface. The detection methods employed in this project are based on the following approaches.

Vehicles are composed of numerous components such as the hood, windows, headlights, bumpers, front grille, and license plate. In an image, such components are usually rendered as segments offering common properties such as color, density, and texture. In this project, intensity (gray level) is used for ease of processing. Assuming that every car component is rendered as a segment of

similar intensities, if a cell is occupied the number of segments must be quite high. However, vacant cells would hardly offer such characteristics, even though white lines, building shadows, or water puddles may exist.

Based on the above reasoning, the following solution is proposed. Images of parking cells are segmented by gray level, and a cell is recognized as occupied if it contains a relatively large number of segments; otherwise, the cell is recognized as vacant.

Once the detection is made, the state of the parking cell can be maintained by temporal frame subtraction. The advantage of temporal subtraction is its computational simplicity and robustness to weather/illumination changes (if capture rate is sufficiently high). The disadvantage is that any detection errors get accumulated. Once a mistake is made, it will carry over until another mistake happens. One solution to this problem is to confirm the state of a parking cell by image segmentation once in a while.

2.2 Implementation

In this project, quadtree decomposition is used for image segmentation. I used Matlab build-in function qtdecomp to perform a quadtree decomposition. This function works by dividing a square block into four equally-sized square blocks, and then testing each block to see if it meets some criterion of homogeneity (e.g., if all the pixels in the block are within a specific dynamic range). If a block meets the criterion, it is not divided any further. If it does not meet the criterion, it is subdivided again into four blocks, and the test criterion is applied to those blocks. This process is repeated iteratively until each block meets the criterion. The result might have blocks of several different sizes, as Fig. 2 below shows.

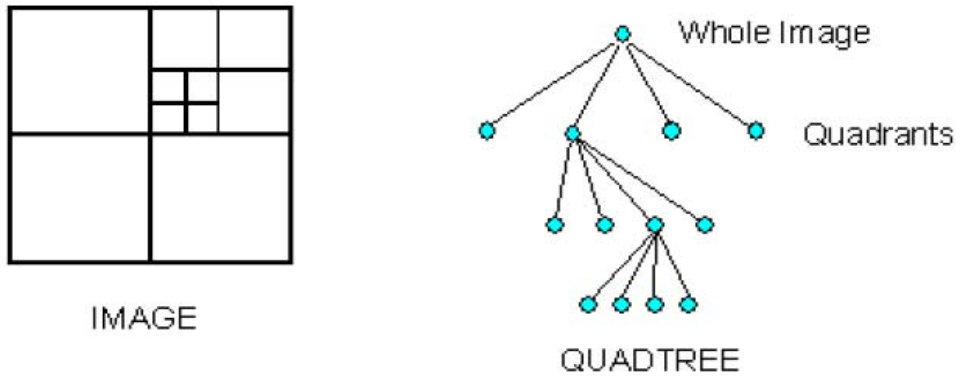

Fig. 2 Quadtree decomposition

The number of resulting blocks in an occupied cell is much larger than it in a vacant cell. The difference is significant and can be a good indication whether the cell is occupied or not; a threshold on the number of blocks can be easily established experimentally. Fig. 3 below shows the result of quadtree decomposition on two parking cells (occupied and vacant). The image of each cell is first resized to a square 128x128 image. Since 128 is power of 2, it makes the subdivision much easier.

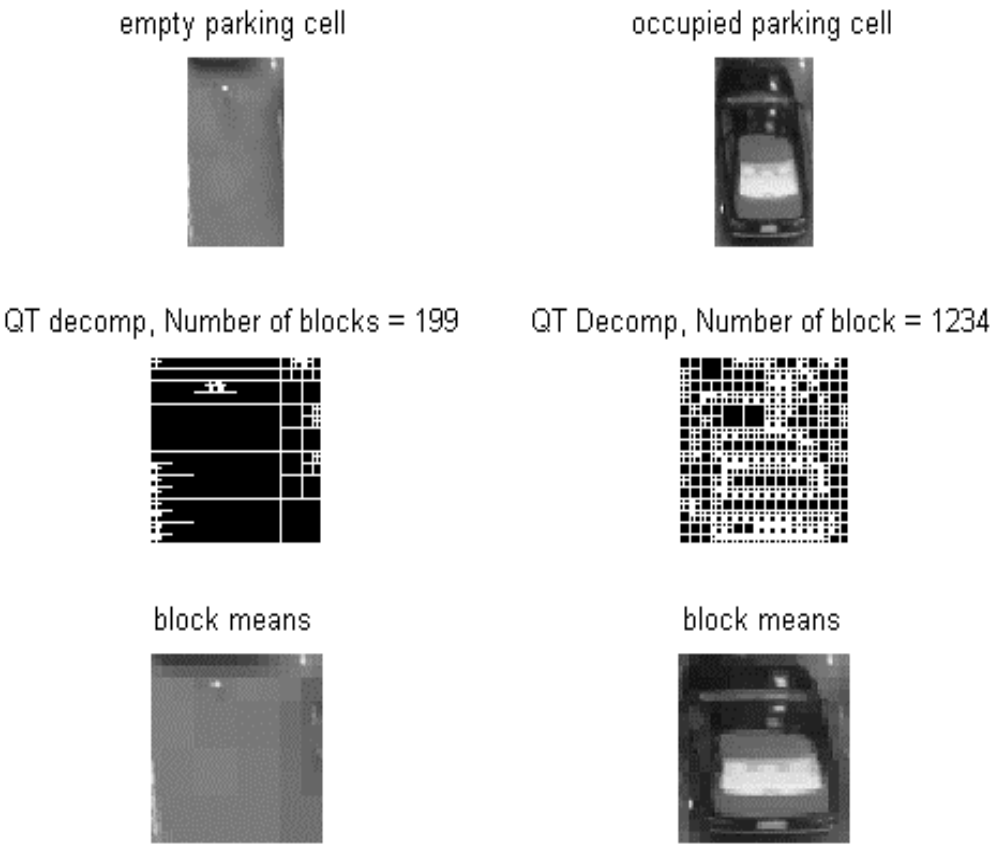

Fig. 3 Example of quadtree decomposition

Once the state of each parking cell is determined by image segmentation, we use temporal frame subtraction to track the state change. The reason behind this is that a parked vehicle usually does not move frequently. On the other hand, it a relatively high computational power is needed to perform image segmentation while it is not necessary to perform such operation on each frame. The state of each cell can be maintained by temporal subtraction. A temporal subtraction does not require much computation compared to image segmentation. If there is no car movement, there will be little temporal change, except minor one due to noise and

illumination variation, etc. However, a significant change can be detected when a car moves in or out of a parking spot.

The temporal subtraction is performed as follows. First, the absolute average pixel change between the current and previous frames is calculated (Fig. 5). Clearly, when there is no car movement, the absolute average pixel change (AAPC) is very low, never higher than 2 (intensity assumed in the 0 to 255 range). If there is car movement, the AAPC is much higher. If AAPC is higher than a threshold, the previous frame is saved as a reference image. For images from Fig. 4, frame number 97 is saved. Once the AACP approaches zero again, the current frame is compared with the reference frame. If the comparison yields a significant change, then a state change is registered, otherwise there is assumed no change. Fig. 6 shows the evolution of a parking cell status (initialized as 0 or empty by the image segmentation procedure) obtained by temporal subtraction.

Fig. 4 Example of image sequence of a parking car

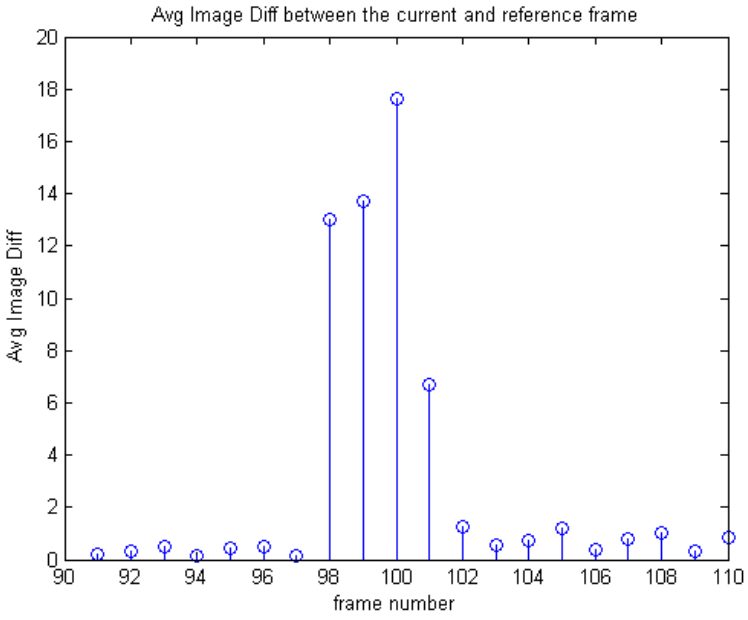

Fig. 5 Average absolute image difference between consecutive frames

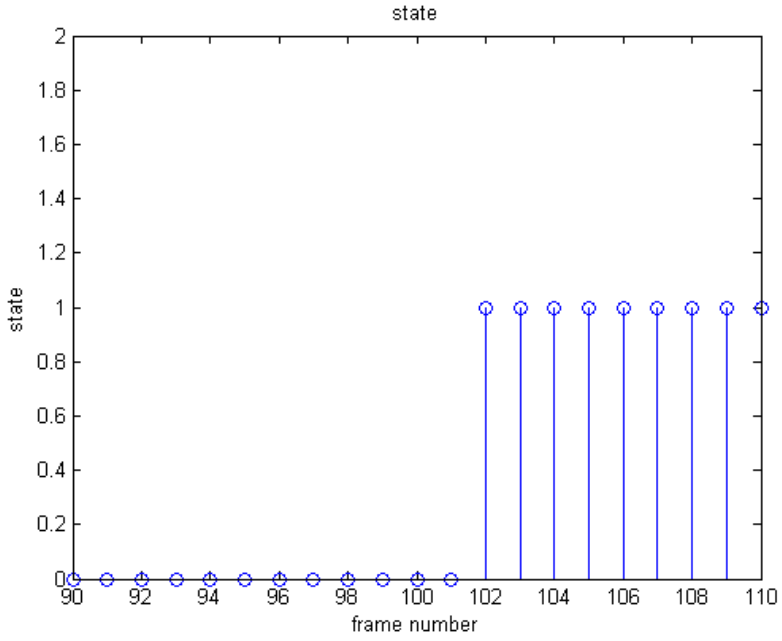

Fig. 6 Evolution of parking cell status along the image sequence

3 Experimental results

A wireless camera was mounted on a tripod and placed in a 4-th floor office of the Photonics Building. The camera overlooks a small parking area in the alley behind the building. Every 5 seconds, a frame is recorded and used for processing.

Since there is no predefined parking cell in the test parking I area (Fig. 7), I had to manually define the parking cell for each video sequence. That takes quite a lot of time, which is the reason I did not test too many video sequences.

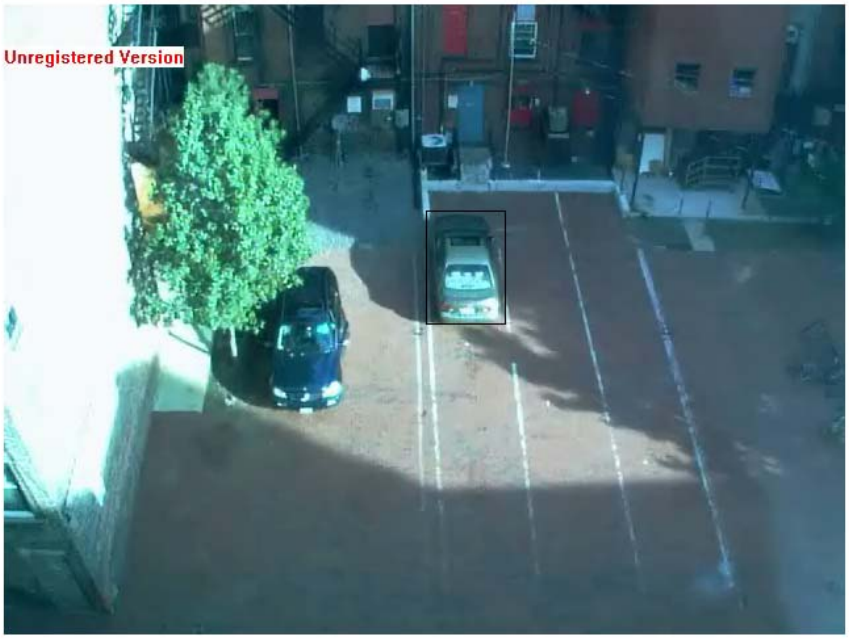

Fig. 7 Example of parking lot image used in experiments

The threshold that determines whether a parking cell is occupied or not based on the number of quadtree blocks, is determined experimentally. Its value is drawn from 100 samples of the data (Fig. 8). The quadtree decomposition was applied to

100 sample parking cells; Fig. 8 shows the histogram of the number of blocks per parking cell image. In vacant-cell images, the number of blocks is typically around 200-300. On the other hand, in occupied-cell images, the number of blocks varies from about 600 to over 1800. It seems that a threshold of 500 should be a good value.

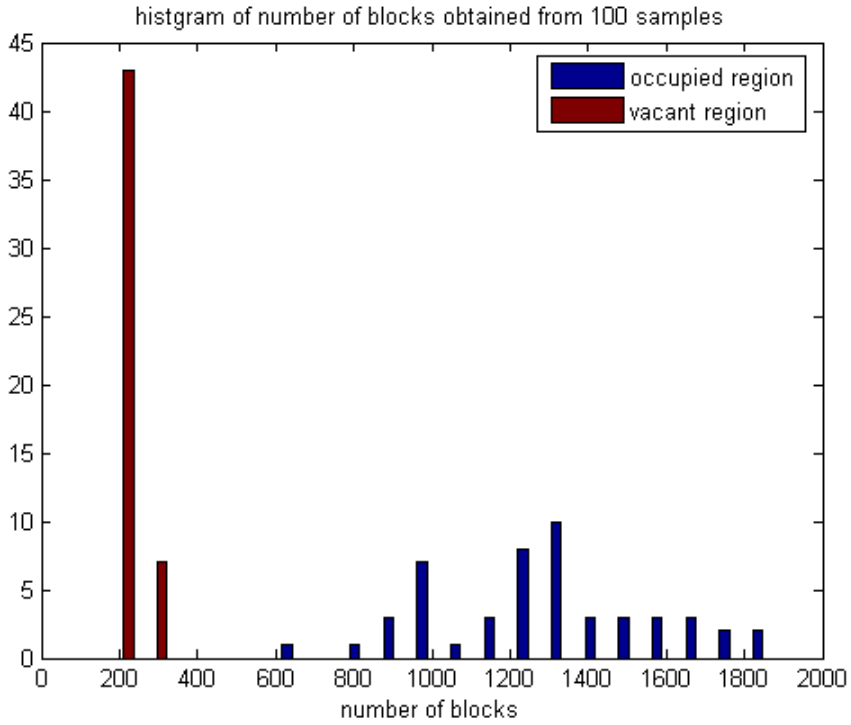

Fig. 8 Histogram of the number of blocks from 100 sample images

Detection results for video clips recorded at different times and under different weather conditions are presented in Table 1. Only video sequences with car movement have been selected for testing.

Table 1 Vehicle presence detection results for various weather conditions

Weather	total number	occupied	vacant	Total	detection rate
rain	54	48	6	54	100
clear	96	68	28	93	96.87
cloudy	134	99	35	134	100
total	284	215	69	281	98.94

The results show very high accuracy of detection. The only misses are caused by poor image quality. In one missed frame, the parking cell was under direct sunshine, while the rest of the area was under building shadow; the camera could not adjust it's gain so the parking cell was totally washed out. In consequence, since details were missing the quadtree decomposition resulted in rather few blocks.

4 Conclusions and possible improvements

Although, the experiments showed great promise of the proposed method, the testing included relatively little video data. Further testing might be needed to confirm the result.

The criterion for splitting in quadtree decomposition was based on the difference of maximum and minimum intensity in a given block. The algorithm will suffer if the noise level is high in the image. Other criteria could also be used, for example color and average intensity might be better candidates for noisy images.

Another important issue is that of parameter selection. There are three parameters in the proposed method: thresholds for block splitting and the number of blocks in quadtree decomposition, and a threshold on the temporal frame difference. These threshold values directly affect the accuracy of the results. They have been all determined experimentally. Different sets of threshold values need to be found if we change image resolution or distance between the camera and the parking lot. Therefore, a method to automatically determine parameter values is needed.

Appendix

Below is listed Matlab source code developed for this project. It can be downloaded from the address below:

mmread - read virtually any media files in Windows

http://www.mathworks.com/matlabcentral/fileexchange/loadFile.do?objectId=8028&objectType=file

```
function Avg_Img_diff=imageAnalysis(referenceImage, currentImage)
% Author: Siming Liu
% Data: 10/27/2005
%
% Function: Perform image absolutel subtraction between two frames,
% and the average absolute pixel difference is returned

[m n] = size(referenceImage);

% convert the images to grayscale
picRef_gray = rgb2gray(referenceImage);
picCur_gray = rgb2gray(currentImage);

img_diff = imabsdiff(picRef_gray,picCur_gray);

total_diff = sum(sum(img_diff));
Avg_Img_diff = total_diff/(m*n);


% the function performs Quadtree decomposition, the min block size is 2x2
% the thresold for decompose critiria is .18, .18x255 = 45 gray-scale lvl.
% Total number of blocks in a decomposed image is returned.

function Totblocks = qtseg(I)

I = rgb2gray(I);
I1 = double(I);
S = qtdecomp(I,.18, 2); % min block size is 2x2

[height width]=size(I);

blocks = repmat(uint8(0),size(S));
Totblocks = 0;
numblocks = 0;   %initialization
```

```
for dim = [512 256 128 64 32 16 8 4 2 1];
   numblocks = length(find(S==dim));
   Totblocks = numblocks+Totblocks;
   if (numblocks > 0)
      values = repmat(uint8(0),[dim dim numblocks]);
      values(1,1,:) = 255;
      blocks = qtsetblk(blocks,S,dim,values);
   end
end

 im = repmat(uint8(0),size(S));

% the code below are used to generate images for illustration purpose,
% not necessary needed for vehicle detection

% block mean representation
% for i=1:height
%       for j=1:width
%            if(S(i,j)~=0)
%                 dim = S(i,j);
%                 temp = I1(i:i+dim-1,j:j+dim-1);
%                 mean = sum(temp(:))/(dim*dim);
%
%                 for l=i:i+dim-1
%                      for m =j:j+dim-1;
%                           im(l,m)=mean;
%                      end
%                 end
%
%            end
%       end
% end
%
% figure;
% subplot(1,2,1);
% imshow(uint8(im));

%   blocks = repmat(uint8(0),size(S));
%            for dim = [512 256 128 64 32 16 8 4 2 1];
%              numblocks = length(find(S==dim));
%              if (numblocks > 0)
%                 values = repmat(uint8(1),[dim dim numblocks]);
%                 values(2:dim,2:dim,:) = 0;
%                 blocks = qtsetblk(blocks,S,dim,values);
%              end
%            end
```

```
%             blocks(end,1:end) =1;
%             blocks(1:end,end) = 1;
%         subplot(1,2,2);
%         imshow(blocks,[])



function ROIregion = ROI(image)

% roi = ROI(image);
%
% Selects the Region of Interest within the 'image', using mouse clicks.
% Only the top left and bottom right corner are needed to spicify the
% region.
%
% 1. Left     mouse button assigns a vertex at the mouse-click location.
% 2. Middle mouse button cancels the last vertex assignment.   (Undo)
% 3. Right   mouse button computes ROI using the selected vertices.
%
% ARGUMENTS
% image = 2-D array of doubles.   Used as a reference to define ROI.

close all;
row = size(image,1);
col = size(image,2);

% Draw the image.

image=rgb2gray(image);

figure; imshow(image);   %display the image;%
hold on;
%


xx = []; yy = [];   % The vertices list.   Initial values are null.
ver = 0;     % # of vertices.   Initialvalue = 0;
b = 1;       % initialize button. Left=1, Middle=2 Right=3.

while b ~= 3

   [x,y,b] = ginput(1);
   if x<1 x=1; elseif x>col x=col; end
   if y<1 y=1; elseif y>row y=row; end
```

```
  if b == 1   % If button is LEFT, add a vertex.
      ver = ver + 1;
      xx= [xx x]; yy= [yy y]; % Update the vertices list.
  elseif b == 2;   % If button is Middle, delete the last vertex, redraw!
    if ver > 1
      ver = ver - 1;
       xx = xx(1:length(xx)-1); yy = yy(1:length(yy)-1); % delete last vertex.
    else
      ver = 0;
      xx = []; yy = [];
    end
  end

end

if (b==3)
    c1=[xx(1), yy(1)];       % coordinates of top left corner
    c2=[xx(2), yy(2)];         % or bottom right
end


xcoord = min(single(c1(1)),single(c2(1)));
ycoord = min(single(c1(2)),single(c2(2)));
width =   abs(single(c1(1))-single(c2(1)));
height = abs(single(c1(2))-single(c2(2)));

ROIregion = [xcoord, ycoord, width, height];

rectangle('position', ROIregion);

hold off;

roi_image = imcrop(image, ROIregion);


end




clear all;
close all;
clc;

video = mmread('test1.wmv',frame_beg:frame_end);
```

```
frame_beg = 1;              % assign beginning frame ane end frame
frame_end = 400;

num_frames = frame_end-frame_beg+1;
t_read = frame_beg:frame_end;
t_process = frame_beg+1:frame_end;


I = video.frames(1).cdata;

roi_region = roi(I);      % assign parking cell region

% initilization of state of the parking cell (1-occupied, 0-vacant)
roi_empty = imcrop(video.frames(1).cdata, roi_region);

img_empty = imresize(roi_empty,[128 128]);   % resize roi to 128x128

numblocks = qtseg(img_empty);

if (numblocks > 500)              % if the number   of blocks is greater than
     state(1) = 1;                % 500, it's determined as occupied
else
     state(1) = 0;
end

state_change(1) = 0;
marker = 0;       % indicate whether there is some moderate change happened


roi_Current = imcrop(video.frames(1).cdata, roi_region);

for i=2:num_frames
       picReference = video.frames(i-1).cdata;
       roi_Reference = imcrop(picReference, roi_region);
       picCurrent = video.frames(i).cdata;
       roi_Current = imcrop(picCurrent, roi_region);
       Avg_Img_diff(i-1)=imageAnalysis(roi_Current, roi_Reference);

       if (Avg_Img_diff(i-1) < 2 && marker ==0)
            state_change(i) = 0;

       elseif (Avg_Img_diff(i-1) >=2 && marker == 0 )
                 img_special=roi_Reference;
                 state_change(i) = 0;
                 marker = 1;
```

```
        elseif (Avg_Img_diff(i-1) >2 && marker == 1 )
                state_change(i) = 0;

        else
            Diff = imageAnalysis(roi_Current,img_special);
                if (Diff > 5)
                    state_change(i) = 1;
                else
                    state_change(i) = 0;
                end
                marker =0;
        end

        if (state_change(i)==0)
            state(i) = state(i-1);
        elseif (state(i-1) ==0)
                 state(i) = 1;
        else
                state(i) = 0;

        end

end


figure;   % avg absolute pixel change
stem(t_process,Avg_Img_diff);
axis([frame_beg frame_end 0 20]);
xlabel('frame number');
ylabel('Avg Image Diff');
title('Avg Image Diff between the current and reference frame');


figure;   % change of state or selected parking cell
stem(t_read,state_change);
axis([frame_beg frame_end 0 2]);
xlabel('frame number');
ylabel('state');
title('state change');

figure;   % the state of the parking cell
stem(t_read,state);
axis([frame_beg frame_end 0 2]);
xlabel('frame number');
ylabel('state');
title('state');
```

References

[1] K. Yamada and M. Mizuno, “A Vehicle Detection Method Using Image Segmentation”, *Electronics and Communications in Japan, Part3, Vol. 84, No. 10, 2001*

[2] Yamada K et al. A vision sensor having an expanded dynamic range for autonomous vehicles. IEEE Trans Vehicular Technol 1998; 47:332-341.