A Simple Implementation for Unmarked Road Tracking

M. Bahgat, *Student Member, IEEE*

***Abstract* – This paper presents a simple implementation for unmarked road tracking. It is mainly based on the calculation of road vanishing point position relative to the vehicle. An algorithm is derived to calculate the vanishing point location. This algorithm is made robust by introducing solutions to noisy peeks, unbounded vanishing point location, and irrelevant lines. Also, the paper defines a method to solve surrounding environment inconsistency by adaptation of thresholds. The paper shows experimental results on both modeled and real world environments.**

***Keywords* – image processing, tracking, mobile robots, vanishing point**

I. Introduction

THE need for automating driving systems has grown significantly in the past few years. Target systems vary from warning drivers, as lane departure warning systems [9]; to aiding driver, like breaking the car in emergency cases; and ending up with complete autonomous vehicles which is still in research. The importance of autonomous vehicles grew as the need to their applications increased like commanding a vehicle to drive to its owner as luxury uses or sending vehicles through battle fields and contaminated areas as safety applications.

One of the main challenges for automating vehicles is maintaining road direction within different environments. Some suggested solutions that were based on lane tracking [8]. Those solutions may not be applicable in some cases as driving a vehicle within crowded areas where lanes are hidden by other vehicles, or driving through roads that are not even marked which form a significant percentage in some developing countries. Other solutions were based on collecting information from surrounding environment. Some of them are based on calculating the relative position of the road's vanishing point and heading towards it. Robust algorithms [5], [6] evolved to calculate the vanishing point location. Some of these were based on complex calculations.

This paper presents a simplified algorithm to track the vanishing point for road guidance purposes, trying to avoid, as much as possible, complex and exhaustive calculations so that the algorithm can be implemented on currently used processors in automotive industry. The paper starts with going through some concepts that are needed to read further. This is followed by details of the algorithm implementation. The section after illustrates the faced problems and the corresponding suggested solutions. Then, experimental results are shown from both modeled and real environments. We conclude the paper with further improvements that can be applied to tune the system performance.

M. Bahgat is with the Department of Computer Engineering at, Faculty of Engineering, Cairo University, Cairo, Egypt (e-mail: mabahgat@gmail.com).

II. Image Processing Concepts

In this section some image processing concepts are introduced to provide a base for reading the rest of the paper. The section starts with a definition of image representation in computers and afterwards goes through operations that are needed to be applied to acquired images.

A. Image Representation

Digital images are represented by their pixel intensity values. Images are represented by a 2D matrix for each channel, for example, a colored image that is represented in RGB space is stored in three 2D matrices, while a gray scale one is represented in one 2D matrix. Each cell in the matrix holds the value of channel intensity for the corresponding location of the pixel.

B. Image smoothing

Images may contain sudden changes between neighboring pixels which cause sharpness in these images. Image smoothing is done by decreasing the differences between neighboring pixels.

One simple idea is to calculate for each pixel a new value which is equal to the average values of the neighboring pixels in addition to its value. In fact, a smoothing mask[1] can be regarded as a low pass filter, which passes only low frequencies[2] in the image. The values of neighboring pixels can be weighted if needed to put emphasis on some property, or even change the target use of the mask.

Smoothing can be useful to achieve several goals. First, roads are full of noise due to the presence of trees, people, or other objects. Smoothing removes some types of noise, but it also, weakens the edges. Second, smoothing can be also regarded as an averaging operator which can be used to collect information about a region instead of a single pixel.

C. Edge Detection

Edges are pixels that have sudden changes in intensity values relative to their neighbors. The absolute difference between the pixel values is proportional to the edge strength; i.e. the pixel probability to be an edge.

Edges are useful to detect lines. These lines can be captured from the surrounding area of the vehicle and used to calculate the direction of the road.

A robust edge detection algorithm is Canny Edge Detector [7]. It has lower and upper thresholds. Pixels having higher edge strength than the upper threshold are considered edges at

[1] A mask is a *2D* matrix containing constants that are multiplied by pixel values in corresponding locations relative to a chosen pixel to obtain a new value for that pixel.

[2] 'Frequencies' means the changes between the adjacent pixels. A black pixel with a neighboring white pixel has a maximum frequency, while two neighboring pixels having the same color is of zero frequency.

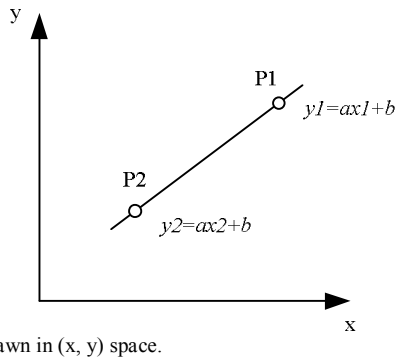

Fig. 1: A line drawn in (x, y) space.

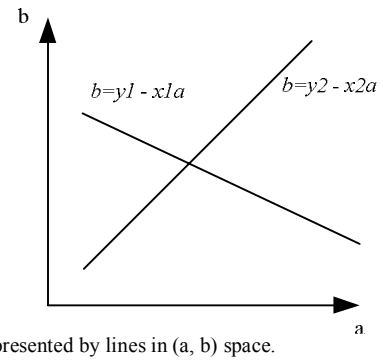

Fig. 2: Points represented by lines in (a, b) space.

once. The lower threshold is used for edge linking, where pixels with weaker edge values than canny upper threshold are considered edges if they are neighboring to other edges. Pixels with edge values under the lower canny threshold are eliminated.

D. Hough Transform [7]

As stated before, images are represented by pixels. So, a line in an image is just a group of pixels positioned beside each other; there is no actual representation for that line which prevents the extraction of any information about it with its representation in that primitive state. Hough transform tries to gather primitive shapes in a given image, like lines or circles.

Our main concern in road guidance is lines detection. A line equation in an (x, y) space is given by (1).

$$y = ax + b \tag{1}$$

where all points belonging to this line satisfies this equation. The constants here are *a* and *b*, which represents the line slope and constant respectively. For a given point, its y-axis and x-axis values are known. So this point lies on all lines where x and y values satisfies their equation.

In *(a, b)* space, where x and y values are known and *a* and *b* values aren't known, each point represents a line with a given slope *a* and constant *b*. A line in that space represents all lines in *(x, y)* space that passes through a certain point having a given *(x, y)* value. The equation of a line in *(a, b)* space is given by (2).

$$b = y - xa \tag{2}$$

A two-line intersection in *(a, b)* space means that there may be a line passing through these two points. When more lines in *(a, b)* space intersects, then that means there are more points on that discovered line. So, to avoid detecting lines that are not really lines, a certain threshold should be used to discard weak lines.

In Fig. 2, the intersection of the two lines in (a, b) space represents the line shown in Fig. 1. The (a, b) space can be called the Cartesian Hough space.

The mentioned representation suffers failure in detecting lines with slope equals to infinity. So another representation was defined based on the normal distance between the line and the origin point and the angle between the line and the horizontal.

Hough transform is used after edge detection to extract surrounding lines. Detected lines are redrawn with a weighting factor for each. A suggestion is to apply another Hough transform to detect their places of intersection in an operation called Cascaded Hough Transform [1].

III. Road Detection

The goal is to keep track of current road direction even if lane markings may disappear. This is achieved by keeping track of the road vanishing point relative to images acquired.

An important issue is to keep the calculations as simple and as fast as possible. Complex calculations were avoided as much as possible, and instead numerous simple ones are used which may, if needed, run in parallel.

This section starts with defining vanishing point and showing the way of calculation used. This is followed by showing enhancements to the way of calculation to solve problems faced.

A. Vanishing Point

The vanishing point lies on the horizon line where the sky and land meet in the image. At this point actual parallel lines seem to converge and intersect.

This point marks the relative direction of the road to the vehicle which is extracted from the parallel lines in the surroundings. The sources of parallel lines can be as follows:

1. Roads boundaries which are normally parallel or semi-parallel.
2. Neighboring vehicles, which are moving in the correct road direction, form parallel lines to the road boundaries and to each other.
3. Buildings which are built on the sides of the road.

Also, other sources can be present other than those stated.

The location of the vanishing point defines the relative orientation of the vehicle to the road. If it is located in the middle of the image then the vehicle is moving in the same direction, i.e. parallel, as the road. If the point is located at the right side of the image that means the vehicle needs to steer right to maintain the road direction, and vice versa for the case where the point lies in the left side.

B. Calculating Vanishing Point Location

The location of the vanishing point can be determined using the following steps.

1. The image is captured in full color into a color image.
2. The color image is converted to grey scale image.
3. The grey scale image undergoes edge detection to produce edges image.
4. The edges image undergoes Hough transformation. The detected lines are redrawn each with intensity equals to 1 unit to produce accumulator image. In case a line passes through a previously marked pixel, the pixel intensity value is incremented.

Accumulator image is scanned to detect the pixel with the highest value. Fig. 3 shows the outputs of the above sequence steps. Note the intensity of grey pixels; as the lines intersect the pixel intensity value increases and as they become sparse it decreases.

C. *Making the Algorithm Robust*

The first problem the previous algorithm faced is the presence of false peak points. An example of this can be seen in Fig. 3(c) where lines coming from the right side form a peak point near the vertical center of the accumulator image. To vote based on peak regions rather than points a smoothing filter, as in (3), is passed through the image to get the average values in the different regions.

$$filter = \frac{1}{25}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} \quad (3)$$

As the vanishing point should be located away from the lower part of the image, the lower part is excluded from the search for peak regions. This avoids detecting false horizon points at the lower part of the image, and also enhances performance.

A more complicated problem appears when using a non wide-lens camera. The horizon point may fall outside the calculation image; i.e. accumulator image. To solve this problem, Cartesian space is divided into subspaces to represent the location of the point outside the image.

A space is needed to represent the point location when its x-axis value tends to go to infinity or negative infinity and another space when its y-axis value tends to go to infinity or negative infinity [3]. Normally, the camera orientation can be adjusted to avoid locating the vanishing point towards y-axis values that tends to infinity, but for the point positioned at the left or right of the accumulator image it can't.

So, points lying at infinity in the original subspace are located at the origin of subspace 2, and points located at the border of subspace 1 are also located at the border of subspace 2.

The peaks from each subspace are weighted and compared, and depending on the implementation a peak is chosen to be the selected vanishing point. Points' locations can be calculated in subspace 2 as shown in (4) and (5).

$$x' = 1/x \quad (4)$$

$$y' = y/x \quad (5)$$

The main advantage of this approach is that the vanishing point can be detected wherever its location is. But, there are also disadvantages. A very efficient implementation is needed for real time applications, as the computations are increased for more than one subspace in addition to the division operations needed. Also the size of subspace 2 has a direct impact on the accuracy of computing the location of the vanishing point.

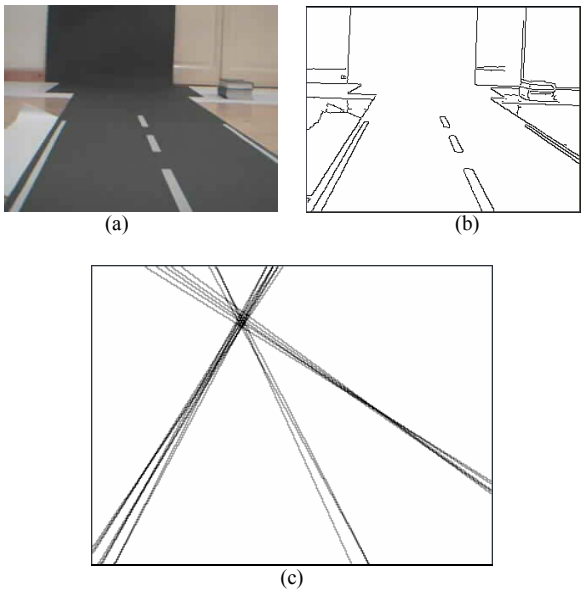

Fig. 3: Vanishing point calculations; (a) step 1, (b) step 3, and (c) step 4.

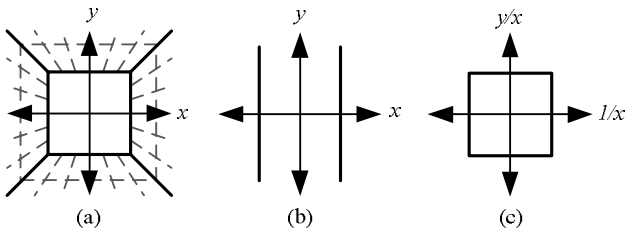

Fig. 4: Cartesian space (a) divided into 2 subspaces; central space shown in (b) (subspace 1) and space containing points having x-axis value tends to infinity shown in (c) (subspace 2)

As a result of camera orientation change to detect the horizon point within the accumulator image, road direction changes may be detected earlier than correct. A first solution can be implemented by inserting a delay between point detection and decision making by queuing the detected points. This isn't a robust solution, since not all direction changes needs the same delay, even some doesn't need at all. Another solution is to reorient the camera angle towards the ground and extend the main subspace with the needed increase in y-axis direction. This solution has also a downside which is decreasing the details seen by the camera.

Another problem is the detection of irrelevant lines. Though Hough transform has an advantage of its robustness even in the presence of noise, captured images may still contain irrelevant lines due to the environment around the road. Those lines can be horizontal lines detected from the back of the vehicle in front or vertical lines coming from surrounding buildings. So, lines are weighted depending on their angle with the horizon as shown in TABLE I.

Another adjustment can be added to fine tune the system is to reject sudden changes in the location of the vanishing point. In addition, the average x-axis values, as we're interested only in steering to left or right, are taken after examining a specified number of frames.

D. Algorithms Adaptations

As the vehicle moves through different environments and different times of day, thresholds need to be dynamically adaptable to changes. For example, driving through a place containing a lot of edges will lead to detection of more noise. For that, both canny edge detection algorithm and Hough transform needs to have their thresholds adjusted gradually depending on their outputs at each frame. Each frame uses the thresholds reached by the previous frame as the environment should slightly change, so adaptation takes less time as the surroundings doesn't usually change suddenly. To avoid falling into local minima where values for the thresholds aren't optimal, the thresholds are pushed back to default values after certain number of frames.

Fig. 5 shows a flow chart describing the sequence.

Each algorithm has its own adaptation adjustment criteria. For Hough transform, the number of lines detected shouldn't go above certain number to avoid taking into account noisy lines. Stronger lines only should be accepted. For canny edge detection, the criterion is the percentage of pixels inside the edges image that are chosen to be edges. If too many pixels are considered as edges, then edge noise percentage is high.

IV. Experimental Results

This section describes the results obtained on running the system. The equipment used is stated first, and then the results for experiments on built model and real world are shown.

A. Equipment

The equipment was chosen to have limited processing power similar to those used in automotives. So, two processors were used interchangeably; an Intel Pentium 3 and another Intel Pentium 4 with 256MB RAM. The camera was a CMOS 1.3 mega pixel that streamed images with a resolution 320 × 240. The same camera was used in the real world and the modeled environments. In the modeled environment a toy car was used to model the real vehicle. Open Computer Vision library was used as a resource to supply algorithms like canny edge detection and Hough transform and other primitive functions.

B. Results in Modeled Environment

The modeled environment was constructed in a lecture hall. The toy car had the camera mounted at the front. The vehicle accessible areas were the bare ground, which is made of grey marble, and some parts of it that was covered by grey paper. Buildings and pavements were modeled by unpainted wooden boxes. There was also green paper covering some areas.

The toy car moved with high percentage of correctness in the modeled environment. It usually reacted correctly to the change in road directions even if it started in an inclined state, though sometimes it misses curves at regions were high reflection of the ground causes wrong estimation for the vanishing point before the system could fully adapt its thresholds especially with the car moving at high speed.

Fig. 6 shows operation images during successful detection of a change in direction towards the right, due to the detection of a peek in subspace 2. The average difference, through a sequence of adapted images, in intensity between the peek region in subspace 1 and 2 is 118 units. Generally, subspace 1

TABLE I
LINE WEIGHTING

Sub-Space	Horizontal lines[3]	Vertical lines[4]	Inclined lines
1	0	0	$\sin(\theta)\times Factor$
2	$\sin(\theta)\times Factor$	0	$\sin(\theta)\times Factor$

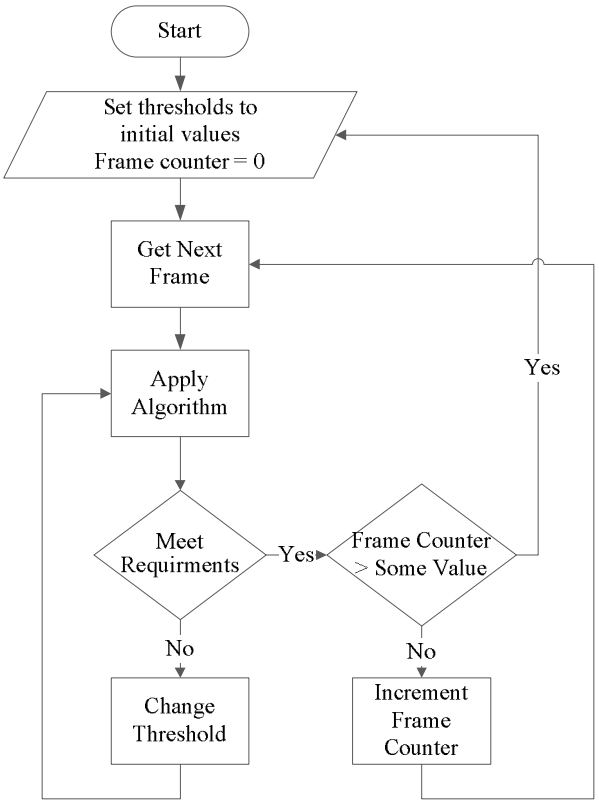

Fig. 5: Thresholds adaptation flow chart

peek value is added to strength constant that increases its value against the peek in subspace 2. The need for this addition is because of the high density of lines in subspace 2 in certain regions because of the use of small calculation image which affects the accuracy. Tracing the recorded images showed that this addition isn't robust and makes identification vanishing point location to be sensitive to the value of the added constant. But from the point of view of the application in question, where we want to decide whether to steer towards the left or right or not steer at all, it's satisfactory to have the point at the correct side regardless of its exact location.

C. Results in Real World Environment

The real world environment experiments were carried out at noon and after noon in sunny conditions. Most of the pictures were taken in city main streets with varying widths, and a single group of images was taken in a side street having a smaller width. Main streets contained significant number of vehicles, and in some cases crowded. For the side street case,

[3] Here horizontal lines are the lines having angles that ranges between 5° to -5° degrees with the horizon.

[4] Here vertical lines are the lines having angles that ranges between 85° to 95° degrees with the horizon.

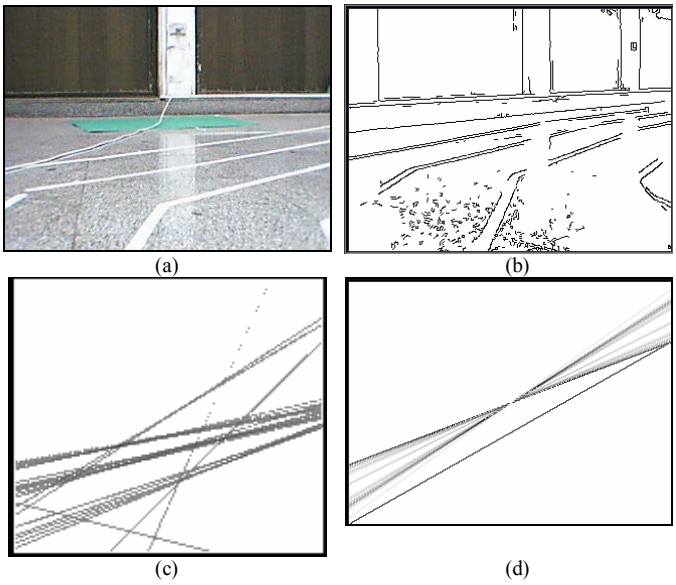

Fig. 6: Experiment in modeled environment result; (a) the original image, (b) edge detection, (c) subspace 1 accumulator image (d) subspace 2 accumulator image

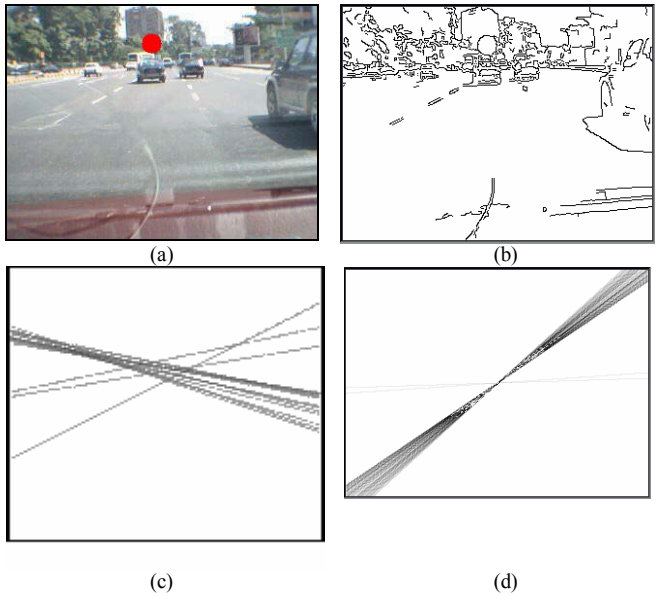

Fig. 7: Experiment in real environment result (vanishing point is the red point); (a) the original image, (b) edge detection, (c) subspace 1 accumulator image (d) subspace 2 accumulator image

it was almost empty and only contained cars parking at both sides. A total of 2366 pictures were captured at speed ranged from 20 km/h to 60 km/h.

The vanishing point location decision rate varies depending on the surrounding conditions. The frame processing rate is done at 1.5 frames per second. On normal conditions, the decision rate can be equal to the frame rate or an average can be calculated over several frames before issuing a decision. On other cases, when lighting conditions change or appearance of surrounding objects causes changes to the detected edges or lines, the adaptation algorithm consumes frames to adjust the algorithms thresholds to reach acceptable results before issuing a vanishing point decision.

The real world experiments presented new conditions which are summarized as follows.

1. The sudden change in the detected vanishing point position is found in significant number of frames. These frames can be rejected to avoid incorrect estimation, but that leads to the decrease in the decision rate.
2. The presence of intersections which introduces several candidate vanishing point locations. Also, if the captured image is dominated by the presence of vehicle turning in that intersection, the vanishing point tends to track that vehicle direction as shown in Fig. 9(a).
3. The presence of obstacles like trees, people, or even other vehicles may hide partially or completely useful details from the road. Also, the sudden change of light or high brightness due to the sun, as in Fig. 9(b), may blind out significant details. At this point, it can be assumed that the direction of the road is equal to the last known good state.
4. The computation with adaptation along with camera accuracy may result in decrease in decision rate as mentioned earlier which may not cope with speed of changes as the vehicle moves quickly.
5. The presence of shadows may improve or degrade detection performance. In some cases, shadows assist locating the vanishing point where these shadows are due to surrounding vehicles and buildings those are oriented in the same direction of the road. Fig. 7 shows a shadow due to a vehicle at the right side which forms a line towards the vanishing point. In some other cases, the appearing shadows are sources of noise especially when these shadows directions are different from the road direction, or if details are hidden from the surrounding area.

To obtain correct horizon point location, the parallel lines used for vanishing point detection should come from both sides of the image. If the lines come from only one side, the location detection becomes sensitive to noise. The lines sources are detected from surrounding vehicles, buildings, road pavements or boundaries, or lane markings. The variety of sources helps increase the robustness of the detection process which overcomes the problem of the disappearance of road boundaries as encountered by [10]. In [11], the proposed method uses texture analysis. This overcomes the need for clear parallel lines. This is achieved at the cost of higher processing power.

The accuracy of vanishing points locations in real environment is less than of the modeled one. In stable light conditions, the percentage of correct detections (without rejecting any frames) is about 60% to 70% which were measured based on two captured image groups in three different main streets. For changing light conditions, the percentage of correct detections falls to 43% based on a single experiment having two main streets common with the previous groups captured at stable light conditions. The main reason for this degradation in accuracy is that the sun light blurs the street boundaries and in some cases hide these boundaries completely especially when exiting shadow areas where light intensity is lower. In this case, the presence of nearby vehicles assisted the detection of the vanishing point locations. At road curves, the correctness percentage was 70%. Errors were mainly due to high light conditions, occurrence of several candidate directions in case of road forking or intersections, or the presence of vehicles parking in different direction other

(a) (b)

Fig. 8: Experiment in real environment result; (a) vanishing point in a side street, (b) detection of road direction change towards the right

(a) (b)

Fig. 9: Negative real environment results; (a) tracking turning cars, (b) edge detection

than curves direction.

Examples of correct decisions are shown in Fig. 7 and Fig. 8. Other examples for incorrect decisions are shown in Fig. 9. A note is that the value of the factor added to subspace 1 peek was increased significantly to obtain correct result.

V. Conclusion and Future Work

As the results were below a safe level of error free decisions, the system, with its current hardware configuration, isn't mature enough to drive a vehicle autonomously within a city. But it can be useful as a driver aiding tool which adds safety and comfort.

Some significant updates can be done on systems with higher configuration. The most important is using a higher resolution camera with wide lens. Another update is to increase the size of subspace 2 to obtain acceptable accuracy. Another is to use more intelligent algorithms to adjust the thresholds of used algorithms. Also, the code needs to be optimized to achieve higher frame rate.

VI. References

[1] T. Tuytelaars, M. Proesmans, and L. Van Gool, "The cascaded Hough transform", *Proceedings, International Conference on Image processing*, Volume 2, 26-29 Oct. 1997, Pages: 736 – 739.

[2] V. Cantoni, L. Lombardi, M. Porta, and N. Sicard, "Vanishing point detection: representation analysis and new approaches", *Proceedings of the 11th International conference on Image Analysis and Processing*, 2001.

[3] K. Seo, JH. Lee, and HM. Choi, "An efficient detection of vanishing points using inverted coordinates image space", *Pattern Recognition Letters*, Volume 2, Issue 2. 2006, Pages: 102 – 108.

[4] J. McCall, and M. Trivedi, "An Integrated, Robust Approach to Lane Marking Detection and Lane Tracking", *Proceedings, IEEE Intelligent Vehicles Symposium*, 2004.

[5] H. Wildenauer, and M. Vincze, "Vanishing Point Detection in Complex Man-made Worlds", *14th International Conference on Image Analysis and Processing*, 2007.

[6] A. Harouni, N. Darwish, and I. Talkan, "Depth estimation from monocular camera in urban environment", 2006.

[7] M. Sonka, V. Hlavac, and R. Boyle, "Image Processing Analysis and Machine Vision Second Edition", PWS Publishing, 1998.

[8] Y. Wang, EK. Teoh, and D. Shen, "Lane detection and tracking using B-Snake", *Elsevier Image and Vision Computing*, 2003.

[9] S. Kol, S. Giml, C. Pan, J. Kim, and K. Pyun, "Road Lane Departure Warning using Optimal Path Finding of the Dynamic Programming", *SICE-ICASE International Joint Conference*, 2006.

[10] Y. Wang, D. Shen, and E. K. Teoh, "Lane detection using catmull-rom spline", *Proceedings on Intelligent Vehicles Symposium*, Volume 1, Oct. 1998, Pages: 51–57.

[11] C. Rasmussen, "Texture-based vanishing point voting for road shape estimation", *Proc. British Machine Vision Conference*, 2004.

VII. Biography

Mohamed Bahgat received his B.S. degree in computer engineering from Cairo University, Cairo, Egypt in 2006. He is currently working toward the M.S. degree in Cairo University.

His research interests include image processing, computer architecture, and parallel programming models.