A Simple Implementation for Unmarked Road Tracking

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 – **vanishing point, autonomous vehicles, image processing, road tracking**

I. INTRODUCTION

The need for computerized 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 of its applications increased as commanding the vehicle to come to its owner as luxury uses or sending vehicles through battle fields and contaminated areas as safety functions.

One of the main challenges for automating vehicles is maintaining road direction within different environments. Some suggested solutions were based on lane tracking [8]. This solution may not be applicable in some cases as driving the vehicle within crowded areas where lanes are hidden by other vehicles, or using 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 calculation to be implementable 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.

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 to be applied on them.

A. Image Representation

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

B. Image smoothing

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

The simple idea is to calculate for each pixel a new value which is equal to the average values of the neighboring pixels including it. 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 used 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 averaging operator which can be used to collect information about a region instead of a single pixel.

[1] A mask is the pixels that are used to calculate the value for a specific 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.

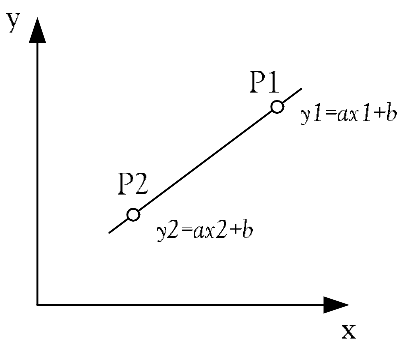


Figure 1: A line drawn in (x, y) space.

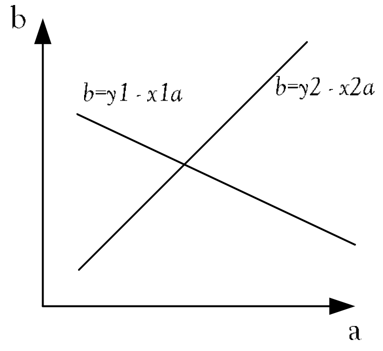


Figure 2: Points represented by lines in (a, b) space.

C. Edge Detection

Edges are the pixels that have a sudden change, represented their intensity, in brightness relative to neighboring pixels. 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 a lower and a higher threshold. Pixels having higher edge strength than the higher thresholds are considered an edge at once. The lower threshold is used for edge linking, where edges with weaker values lower than the canny higher threshold when neighboring to edges then they are considered edges.

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 preventing the extraction of any information about the line if it remains in its primitive state. Hough transform try 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

$$y = ax + b \qquad (1)$$

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 know and *a* and *b* values aren't known, each point represents a line with a give 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

$$b = y - xa \qquad (2)$$

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

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

Hough transform is used after edge detection to perform surrounding lines extraction. 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 though 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 possible, 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 that means 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 point lies in the left side.

B. Calculating Vanishing Point Location

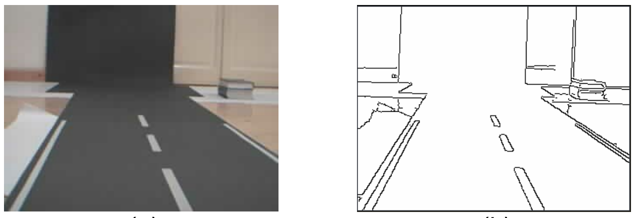
The location of the vanishing point can be marked using the following steps.

1. The image is captured in full color.
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.
5. Accumulator image is scanned to detect the pixel with the highest value.

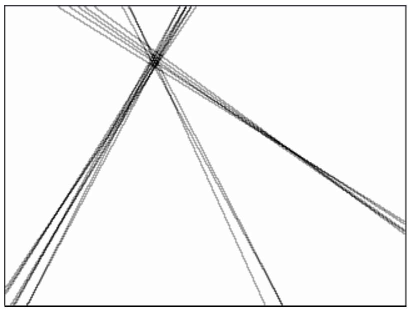
Figure 3 shows the steps of the above sequence. Note the intensity of grey pixels, as the lines intersect the pixel intensity value increases and as they become sparse it decreases.

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 Figure 3(c) where lines coming from the right side near the middle of the accumulator image. To vote based on *peak regions* rather than points a smoothing filter, as in equation 3, is passed through the image to get the average values in the different regions.



(a) *(b)*



(c)

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

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

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 cameras, 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 to 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* infinity, but for the point lying at the left or right of the accumulator image we can't.

So, points lying near the origin tend to be located at infinity, and points lying at the edges of the subspace 1 lye at the edge of subspace 2. Similarly, subspace 3 can be evaluated.

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

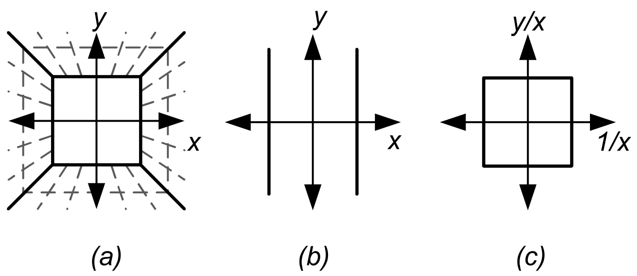


Figure 4: Cartesian space (a) divided into 2 subspaces; centeral space shown in (b) (subspace 1) and x-axis infintyshown in (c) (subsapce 2)

$$x' = {}^{1}\!/_{x} \tag{4}$$

$$y' = {}^{y}\!/_{x} \tag{5}$$

The main advantage of this approach is that we can detect the vanishing point 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.

As a result of camera orientation change to detect the horizon point within the accumulator image, road direction changes may be detected earlier than supposed to. A first solution can be implemented by inserting a delay between detection and decision making by queuing the detected points. This isn't a robust solution, since not all direction changes needs delay, even some doesn't need at all. Another 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 which the vehicle is moving through like the horizontal lines detected form the back of the vehicle in front or vertical lines coming from buildings. So, lines are weighted depending on their angle with the horizon as shown in Table I.

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	$\sin(\theta)\times Factor$

3 Here it's included in the definition of horizontal lines the lines that are mostly horizontal, 85° to 95°

4 Same as horizontal lines, but for the vertical case

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 through 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 adjust their thresholds 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. Figure 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 that are chosen to be edges. If too many pixels are detected, then noisy edges are detected.

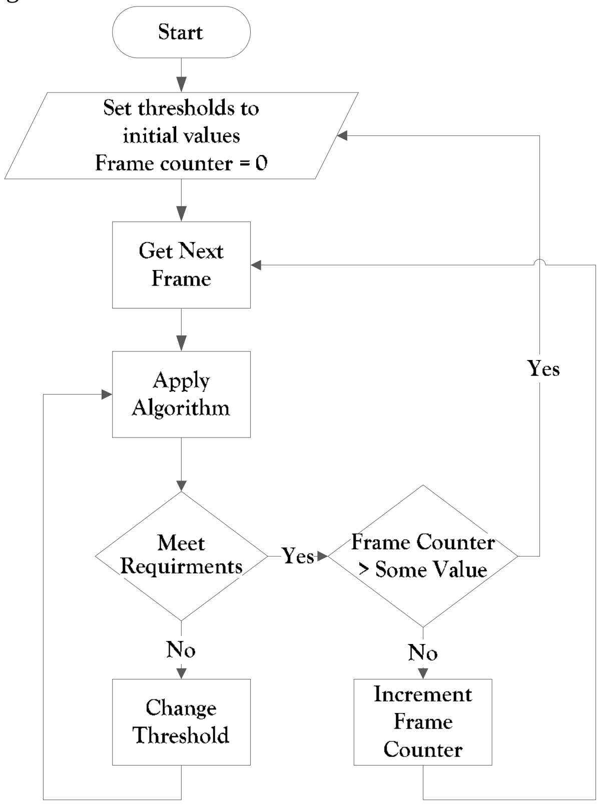


Figure 5: Thresholds adaptation flow chart

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 as 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 megapixel 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 toy car moved with percentage of correctness in the modeled environment. It usually it 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 cause wrong estimation for the vanishing point before the system could fully adapt its thresholds.

Figure 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 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 correctness of vanishing point location degrades in the real environment. Reasons for this were:

1. The presence trees and people passing within the road which may hide useful details.
2. The computations with adaptation along with camera accuracy couldn't cope with the speed of changes as the vehicle moves quickly.
3. The presence of intersections which introduces a lot of candidate directions.
4. The sudden reflection of light which blinds out significant details.
5. The presence of obstacles that may completely cover the road. (We may assume in this case that we should go the same direction as the last known good state).

Nevertheless, significant percentage of images captured during the test drive could point out the road direction correctly as Figure 7. A note to be mentioned is that the adding factor to subspace 1 peek value needed to be increased significantly to obtain correct result.

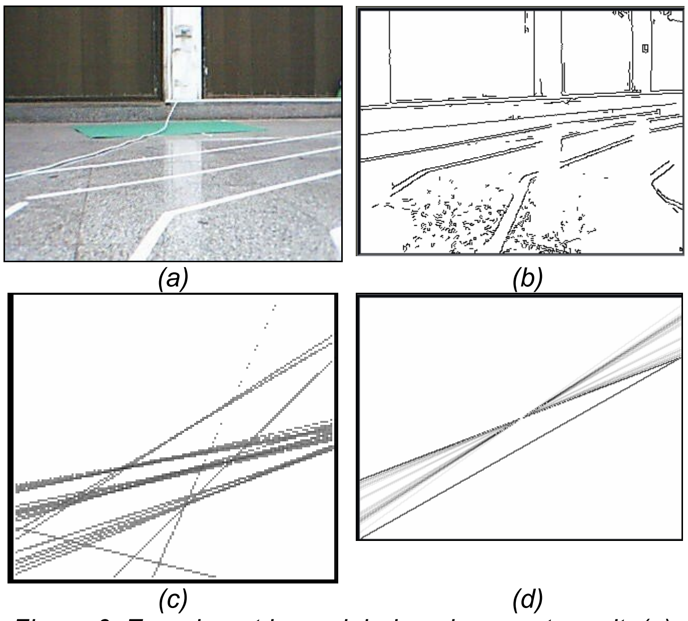


Figure 6: Experiment in modeled environment result; (a) the original image, (b) edge detection, (c) subspace 1 accumulator image (d) subspace 2 accumulator image

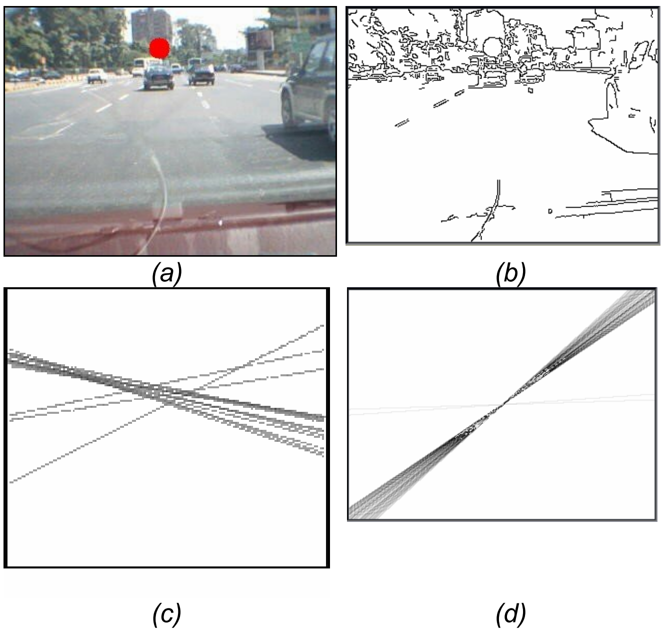


Figure 7: Experiment in real environment result; (a) the original image, (b) edge detection, (c) subspace 1 accumulator image (d) subspace 2 accumulator image

V. CONCLUSION AND FUTURE WORK

As the results were bellow a safe level of error free decisions, the system, with its current equipment, 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 more powerful systems. One is to increase the size of subspace 2 to obtain acceptable accuracy. Another update is to use more intelligent algorithms to adjust the thresholds of used algorithms. In addition, higher resolution cameras with wide lens would increase the performance of the system.

REFERENCES

1. Tuytelaars, T.; Proesmans, M.; and Van Gool, L.; "The cascaded Hough transform", *Image Processing, 1997. Proceedings, International Conference on Volume 2*, 26-29 Oct. 1997, Pages: 736 – 739.
2. Cantoni, V.; Lombardi, L.; Porta, M.; and Sicard, N.; "Vanishing point detection: representation analysis and new approaches", *Proceedings of the 11th International conference on Image Analysis and Processing*, 2001.
3. Seo, KS.k; Lee, JH.; and Choi, HM.; "An efficient detection of vanishing points using inverted coordinates image space", *Pattern Recognition Letters, Volume 2, Issue 2*. 2006, Pages: 102 – 108.
4. McCall, J.; and Trivedi; M.; "An Integrated, Robust Approach to Lane Marking Detection and Lane Tracking", *Proceedings, IEEE Intelligent Vehicles Symposium*, 2004.
5. Wildenauer, H; and Vincze, M.; "Vanishing Point Detection in Complex Man-made Worlds", *14th International Conference on Image Analysis and Processing*, 2007.
6. Harouni, A,; Darwish, N.; and Talkan, I.; "Depth estimation from monocular camera in urban environment", 2006.
7. Sonka, M.; Hlavac, V.; and Boyle, R.; *Image Processing Analysis and Machine Vision Second Edition*, PWS Publishing, 1998.
8. Wang, Y.; Teoh, EK.; and Shen, D.; "Lane detection and tracking using B-Snake:, *Elsevier Image and Vision Computing*, 2003.
9. Kol, S.; Giml, S.; Pan, C.; Kim, J.; and Pyun, K.; "Road Lane Departure Warning using Optimal Path Finding of the Dynamic Programming", *SICE-ICASE International Joint Conference*, 2006.