CHAPTER 4

ROAD SURFACE CONDITION DETECTION

Road surface condition, represented by friction coefficient, plays a significant role in vehicle stability control. The stability control is focused mainly on controlling the vehicle’s yaw rate during vehicle cornering, where the yaw rate is affected by the surfaces’ friction coefficient.

Since the surface’s coefficient of friction is difficult to measure in real-time, it is necessary to relate the surface friction coefficient to certain variables that can be estimated. The relationship between the coefficient of friction and slip ratio has been experimentally determined by Gustafsson (1977).

Advanced driver-assistance systems aim to understand the environment of the vehicle contributing to traffic safety. In this context, not only active sensors (e.g., radar and laser scanner) but also passive sensors (e.g., different types of cameras) play a relevant role. Vision-based approaches are used to address functionalities such as lane marking detection, traffic sign recognition, pedestrian detection, etc.

Here the focus is on vision-based road detection. That is, detecting the free road surface ahead of the ego-vehicle using an onboard camera (see Figure 4.1). Road detection is an important task within the context of autonomous driving. Moreover, it is an invaluable background segmentation stage for other functionalities such as vehicle as suggested by Sun et al (2005)
and pedestrian detection determined by Broggi et al (2009). The knowledge of the free road surface reduces the image region to search for these targets, thus contributing to reach real time and to reduce false detection.

Figure 4.1  Aim of vision-based road detection is identifying road pixels in an image

Road detection is a challenging task since roads are in outdoor scenarios imaged from a mobile platform. Hence, on the one hand, the algorithms must deal with a continuously changing background and the presence of different objects (vehicles, pedestrians, and infrastructure elements) with unknown movements. On the other hand, the algorithms must deal with high intraclass variability since there are different road types (shape, size, and wear down) and different imaging conditions (e.g., varying illumination).

The review of related literature reveals that using a monocular color camera as a sensor as suggested by Broggi et al (2004) is the preferred option. Therefore, texture and color are potential features to characterize the road. However, the imaged road texture varies too much with the distance to the camera; thus, color analysis is the option that is chosen most. Some authors propose the use of depth as an additional cue for adding robustness, as informed by Dahlkamp et al (2006) in which depth comes from active sensors, or inDahlkamp et al (2009), in which depth comes from a stereo rig.
Current road-detection algorithms use different color spaces as features, as well as different classification processes. For instance, the hue–saturation–intensity (HSI) color space is used in Dahlkamp et al (2008). HSI is a color space that mitigates the influence of lighting variations as described by Dahlkamp et al (2000) and Sigal et al (2004). However, HSI does not appropriately behave when the road presents shadowed areas. Thus, additional constraints must be included to improve their performance, e.g., in Sotelo et al (2004). Use road shape restrictions and ad hoc postprocessing to recover undetected shadowed areas. Other authors use standard red–green–blue (RGB) and put more emphasis on the classification process. For instance, a mixture of Gaussians (MoG) is used in article specified by Lee & Crane (2006) and Dahlkamp et al (2006) for road modeling. However, selecting the proper number of Gaussians is not straightforward, and the presence of shadows is still a problem.

Tan et al (2006) use histograms in the rg color space to build a model for the road and another for the background. The road variability is modeled using different histograms, which are dynamically updated from frame to frame. This is not a straightforward process since empirical criteria to incorporate new histograms, remove old ones, fuse similar ones, and fix the number of histograms are required. The background is modeled using a single histogram. This way, background pixels from one frame are used to build the background model in the next frame, which is a kind of temporal constraint. Therefore, errors are propagated, particularly when other vehicles are present in the scene.

In Ramstrom & Christensen, (2006) several color spaces, i.e., UV, rg, and luminance, are combined. For each color space, two MoGs are used: one to model the road and the other to model the background. Then, the results from each color space are combined using a voting approach.
However, selecting the proper number of Gaussians is an issue, as well as background modeling. In addition, the method relies on road shape restrictions.

All these works use road shape constraints, temporal assumptions, or complex models to mitigate the problem of lighting variations and shadows. However, these assumptions and models drop off the performance of the algorithms in many common driving situations. For instance, road shape restrictions are difficult to apply with complex road shapes (crossings or roundabouts) or when the road borders are not clearly visible (due to traffic or in urban scenarios). Temporal restrictions do not hold when abrupt driving changes occur as accelerations/decelerations of the ego-vehicle or appearing/disappearing of other vehicles.

Image processing techniques are employed to develop this algorithm. It is discussed in detail in this section.

4.1 CANNY EDGE DETECTOR

Canny's aim was to discover the optimal edge detection algorithm. In this situation, an "optimal" edge detector means:

Good detection – the algorithm should mark as many real edges in the image as possible.

Good localization – edges marked should be as close as possible to the edge in the real image.

Minimal response – a given edge in the image should only be marked once, and where possible, image noise should not create false edges.
To satisfy these requirements Canny used the calculus of variations – a technique which finds the function which optimizes a given functional. The optimal function in Canny's detector is described by the sum of four exponential terms, but it can be approximated by the first derivative of a Gaussian.

Because the Canny edge detector is susceptible to noise present in raw unprocessed image data, it uses a filter based on a Gaussian (bell) curve, where the raw image is convolved with a Gaussian filter. The result is a slightly blurred version of the original which is not affected by a single noisy pixel to any significant degree.

An edge in an image may point in a variety of directions, so the Canny algorithm uses four filters to detect horizontal, vertical and diagonal edges in the blurred image. The edge detection operator (Roberts, Prewitt, Sobel for example) returns a value for the first derivative in the horizontal direction (Gx) and the vertical direction (Gy). From this the edge gradient and direction can be determined:

\[ G = \sqrt{G_x^2 + G_y^2} \]
\[ \Theta = \arctan(2G_y, G_x) \]

where G can be computed using the hypot function and atan2 is the arctangent function with two arguments. The edge direction angle is rounded to one of four angles representing vertical, horizontal and the two diagonals (0, 45, 90 and 135 degrees for example).
Figure 4.2 (a) Cement road with lane mark, (b) lane mark identified

Figure 4.3 (a) Asphalt road, (b) Edge detected frame

Figure 4.4 (a) Cement road, (b) Edge detected frame

Figure 4.5 (a) Rough road, (b) edge detected frame
Figure 4.6  Edges with insensitive threshold value for different types of road surfaces

Figure 4.7  Edges with sensitive threshold value for different types of road surfaces
Edge detection is used for (i) identification of blurred frames (ii) broad classification among smooth and rough surface (iii) classification of cement and asphalt. The Canny edge detection is performed on the frames with the sensitive threshold values (upper threshold 10000 and lower threshold 4900) and again it is performed with the insensitive threshold values (upper threshold 50000 and lower threshold 9800). If a pixel has a gradient greater than the upper threshold, then it is an edge pixel. If a pixel has a gradient lower than the lower threshold, it is not an edge pixel. If the pixel’s gradient is between the upper and the lower thresholds, then it is considered as an edge, only if it is connected to a pixel that is above the high threshold. The number of edges is then computed.

The Canny edge detector is an edge detection operator that uses a multi-stage algorithm to detect a wide range of edges in images. It was developed by John F. Canny in 1986. Canny also produced a computational theory of edge detection explaining why the technique works.

4.2 CONTOURS

Contour is also known as border following or boundary following. Contour tracing is a technique that is applied to digital images in order to extract their boundary.

Contour tracing is one of many preprocessing techniques performed on digital images in order to extract information about their general shape. Once the contour of a given pattern is extracted, it's different characteristics will be examined and used as features which will later on be used in pattern classification. Therefore, correct extraction of the contour will produce more accurate features which will increase the chances of correctly classifying a given pattern.
The contour pixels are generally a small subset of the total number of pixels representing a pattern. Therefore, the amount of computation is greatly reduced when the feature extracting algorithms are run on the contour instead of on the whole pattern. Since the contour shares a lot of features with the original pattern, the feature extraction process becomes much more efficient when performed on the contour rather on the original pattern.

![Figure 4.8 (a) Rough road, (b) contours](image)

**Figure 4.8 (a) Rough road, (b) contours**

In conclusion, contour tracing is often a major contributor to the efficiency of the feature extraction process - an essential process in the field of pattern recognition.

Contours are connected edges and are used to (i) identify the lane marking (ii) classify rough asphalt from rough road. Contours are drawn on the frame with the threshold value of 210 and again contours are drawn for the threshold value of 75. The reason for choosing the threshold values are mentioned in the section III. The threshold values are chosen such that the pixels brighter than the threshold value are alone identified. The number of contours for the threshold value of 75 is alone computed.

### 4.3 HOUGH TRANSFORM

The Hough transform is a feature extraction technique used in image analysis, computer vision and digital image processing. The Hough
transform is a technique which can be used to isolate features of a particular shape within an image. Because it requires that the desired features be specified in some parametric form, the classical Hough transform is most commonly used for the detection of regular curves such as lines, circles, ellipses, etc. A generalized Hough transform can be employed in applications where a simple analytic description of a feature is not possible. Due to the computational complexity of the generalized Hough algorithm, the main focus of this discussion is restricted to the classical Hough transform. Despite its domain restrictions, the classical Hough transform retains many applications, as most manufactured parts contain feature boundaries which can be described by regular curves. The main advantage of the Hough transform technique is that it is tolerant of gaps in feature boundary descriptions and is relatively unaffected by image noise.

The Hough technique is particularly useful for computing a global description of a feature, given (possibly noisy) local measurements. The motivating idea behind the Hough technique for line detection is that each input measurement (e.g. coordinate point) indicates its contribution to a globally consistent solution (e.g. the physical line which gave rise to that image point).

As a simple example, consider the common problem of fitting a set of line segments to a set of discrete image points (e.g. pixel locations output from an edge detector). Figure 4.9 shows some possible solutions to this problem. Here the lack of a priori knowledge about the number of desired line segments (and the ambiguity about what constitutes a line segment) render this problem under-constrained.
Figure 4.9 a) Coordinate points. b) and c) Possible straight line fittings

Analytically a line segment can be described in a number of forms. However, a convenient equation for describing a set of lines uses parametric or normal notion:

\[ x \cos \theta + y \sin \theta = r \]

where \( r \) is the length of a normal from the origin to this line and \( \theta \) is the orientation of \( r \) with respect to the X-axis. (See Figure 4.10) For any point \((x,y)\) on this line, \( r \) and \( \theta \) are constant.

Figure 4.10 Parametric description of a straight line.
In an image analysis context, the coordinates of the point(s) of edge segments (i.e. $x_i$, $y_i$) in the image are known and therefore serve as constants in the parametric line equation, while $r$ and $\theta$ are the unknown variables. Plot the possible $(r, \theta)$ values defined by each $(x_i, y_i)$, points in cartesian image space map to curves (i.e. sinusoids) in the polar Hough parameter space. This point-to-curve transformation is the Hough transformation for straight lines. When viewed in Hough parameter space, points which are collinear in the cartesian image space become readily apparent as they yield curves which intersect at a common $(r, \theta)$ point.

The transform is implemented by quantizing the Hough parameter space into finite intervals or accumulator cells. As the algorithm runs, each $(x_i, y_i)$, is transformed into a discretized $(r, \theta)$ curve and the accumulator cells which lie along this curve are incremented. Resulting peaks in the accumulator array represent strong evidence that a corresponding straight line exists in the image.

The same procedure can be used to detect other features with analytical descriptions. For instance, in the case of circles, the parametric equation is

$$(x - a)^2 + (y - b)^2 = r^2$$

Where $a$ and $b$ are the coordinates of the center of the circle and $r$ is the radius. In this case, the computational complexity of the algorithm begins to increase as three coordinates are present in the parameter space and a 3-D accumulator. (In general, the computation and the size of the accumulator array increase polynomially with the number of parameters. Thus, the basic Hough technique described here is only practical for simple curves.)
Hough transform is used to (i) identify man made partitions on the surfaces (ii) find the line segments (lane marking) in an image. The Standard Hough transform is used to map each pixel in image space to a line in Hough space and vice versa.

4.4 INTENSITY HISTOGRAM

In an image processing context, the histogram of an image normally refers to a histogram of the pixel intensity values. This histogram is a graph showing the number of pixels in an image at each different intensity value found in that image. For an 8-bit grayscale image there are 256 different possible intensities, and so the histogram will graphically display 256 numbers showing the distribution of pixels amongst those grayscale values. Histograms can also be taken of color images, either individual histogram of red, green and blue channels can be taken, or a 3-D histogram can be produced, with the three axes representing the red, blue and green channels, and brightness at each point representing the pixel count. The exact output from the operation depends upon the implementation. It may simply be a picture of the required histogram in a suitable image format, or it may be a data file of some sort representing the histogram statistics.

Histograms have many uses. One of the more common is to decide what value of threshold to use when converting a grayscale image to a binary one by thresholding. If the image is suitable for thresholding then the histogram will be bi-modal i.e. the pixel intensities will be clustered around two well-separated values. A suitable threshold for separating these two groups will be found somewhere in between the two peaks in the histogram. If the distribution is not like this then it is unlikely that a good segmentation can be produced by thresholding.
Being given a grayscale image with L levels of intensity (for an image having 8 bits / pixel \( L=255 \)), the intensity (grey) level histogram is defined by a function \( h(g) \) that has value, for each intensity level \( g \in [0 \ldots L] \), the number of pixels in the image or in the region of interest that have intensity equal to \( g \).

\[
h(g) = N_g
\]

\( N_g \) – the number of pixels in the image or in the region of interest that have the intensity equal to \( g \).

The function obtained by normalizing the histogram with the number of pixels in the image (in the ROI) is called the probability density function (PDF) of the intensity levels.

\[
p(g) = \frac{h(g)}{M}
\]

where  \( M \)- Image\_height x Image\_width
Figure 4.11 (a) Sand road

Figure 4.11 (b) Histogram of Figure 4.11 (a)
Figure 4.12 (a) Grass road

Figure 4.12 (b) Histogram of Figure 4.12 (a)
Figure 4.13 (a) Asphalt road

Figure 4.13 (b) Histogram

Histogram is used to (i) classify sand from cement/asphalt (ii) classify grass from rough surface. The total number of pixels having intensity values ranging from 150 to 195, 135 to 170 and 50 to 200 are found to classify sand, cement/asphalt and grass respectively.
4.5 SUPPORT VECTOR MACHINE (SVM)

A Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane. In other words, given labeled training data (supervised learning), the algorithm outputs an optimal hyperplane.

![Linearly separable set of 2D-points](image)

Figure 4.14 Linearly separable set of 2D-points which belong to one of two classes and separating straight line

In the above picture you can see that there exists multiple lines that offer a solution to the problem. Is any of them better than the others? Intuitively define a criterion to estimate the worth of the lines:

A line is bad if it passes too close to the points because it will be noise sensitive and it will not generalize correctly. Therefore, goal should be to find the line passing as far as possible from all points.

Then, the operation of the SVM algorithm is based on finding the hyperplane that gives the largest minimum distance to the training examples.
Twice, this distance receives the important name of margin within SVM’s theory. Therefore, the optimal separating hyperplane maximizes the margin of the training data.

![Figure 4.15 Linearly separable set of 2D-points which belong to one of two classes and separated by SVM hyperplane](image)

Let’s introduce the notation used to define formally a hyperplane:

\[ f(x) = \beta_0 = \beta_x^T, \]

where \( \beta \) is known as the weight vector and \( \beta_0 \) as the bias.

The optimal hyperplane can be represented in an infinite number of different ways by scaling of \( \beta \) and \( \beta_0 \). As a matter of convention, among all the possible representations of the hyperplane, the one chosen is

\[ \left| \beta_0 = \beta_x^T \right| = 1 \]
where $x$ symbolizes the training examples closest to the hyperplane. In general, the training examples that are closest to the hyperplane are called support vectors. This representation is known as the canonical hyperplane.

Now, use the result of geometry that gives the distance between a point $x$ and a hyperplane $(\beta, \beta_0)$:

$$
distance = \frac{\beta_0 = \beta_x^T \beta}{\| \beta \|} = 1
$$

In particular, for the canonical hyperplane, the numerator is equal to one and the distance to the support vectors is

$$
distance_{support \ vectors} = \frac{\beta_0 = \beta_x^T \beta}{\| \beta \|} = \frac{1}{\| \beta \|}
$$

Recall that the margin introduced in the previous section, here denoted as $M$, is twice the distance to the closest examples:

$$
M = \frac{2}{\| \beta \|}
$$

Finally, the problem of maximizing $M$ is equivalent to the problem of minimizing a function $L(\beta)$ subject to some constraints. The constraints model the requirement for the hyperplane to classify correctly all the training examples $x_i$. Formally,

$$
\min_{\beta, \beta_0} L(\beta) = \frac{1}{2} \| \beta \|^2 \text{ subject to } y_i (\beta_x^T + \beta_0) \geq \forall i,
$$

Where $y_i$ represents each of the labels of the training examples.
The basic SVM takes a set of input data and predicts, for each given input, which of two possible classes forms the output, making it a non-probabilistic binary linear classifier.

The present research work is focused on developing a Support Vector Machine (SVM) based algorithm for the detection and classification of surface with road surface images. The road surface condition affecting the vehicle dynamics very much is always difficult to determine in real time. To address this problem, an SVM-based road friction coefficient estimation method is proposed.

First of all, the relationship between coefficient of friction and slip ratio depends on the road surface condition. The five surface conditions under study are Asphalt, Cement, Sand, Grass and Rough Road. The relationship between the coefficient of friction and slip ratio has been experimentally determined by Gustafsson (1977). However, this relationship is difficult to be represented by a polynomial function. To overcome this drawback, it is proposed to train the relationship by a Support Vector Machine.