Municipal Road Infrastructure Assessment Using Street View Images

by

David Abou Chacra

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Systems Design Engineering

Waterloo, Ontario, Canada, 2016

This thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Statement of Contributions

Two publications have resulted from the work presented in this thesis, mainly concerning road segmentation:

1. [8]: David Abou Chacra and John Zelek. Using superpixels for road segmentation in street view images. *Vision Letters*, 1(1), 2015

2. [9]: David Abou Chacra and John Zelek. Road segmentation in street view images using texture information. In *13th Conference on Computer and Robot Vision (CRV) 2016*, 2016

Abstract

Road quality assessment is a crucial part in Municipalities' work to maintain their infrastructure, plan upgrades, and manage their budgets. Properly maintaining this infrastructure relies heavily on consistently monitoring its condition and deterioration over time. This can be a challenge, especially in larger towns and cities where there is a lot of city property to keep an eye on.

Municipalities rely on surveyors to keep them up to date on the condition of their infrastructure to prevent this failure before it happens. This is both to prevent injuries and further damage from occurring as a result of infrastructure failure, and since it is can be more cost effective to maintain property rather than have to replace it. Surveying can either be done manually or automatically, but it is not done frequently as it is expensive and also time consuming. Manual surveying can be inaccurate, while a large portion of automatic surveying techniques rely on expensive equipment. To solve this problem, we propose an automated infrastructure assessment method that relies on Street View images for its input and uses various computer vision and pattern recognition methods to generate its assessments.

First, we segment the image into 'road' and 'background' regions. We propose a road segmentation algorithm specifically aimed at segmenting roads from street view images. We use Fisher vectors calculated on SIFT descriptors to encode small windows extracted from the main image at multiple scales. Then we classify these patches using an SVM and utilize a Gaussian voting scheme to obtain a segmentation. We additionally utilize a spatial prior to improve this segmentation. Optionally, we improve the segmentation further by making use of a weighted contour map calculated on a shadow-free intrinsic image, and a find an optimal segmentation by utilizing a purity tree. Our algorithm performs well and outputs a good segmentation for further use in road evaluation. We test our method on the KITTI road dataset, and compare it to the state-of-the-art on this dataset, along with a manually annotated subset of Google Street View.

After segmenting the road, we describe an algorithm aimed at identifying distressed road regions and pinpointing cracks within them. We predict distressed regions by reusing the computed Fisher vectors and classifying them with a different SVM trained to distinguish between road qualities. We follow this step with a comparison to the weighed contour map within these distressed regions to identify exact crack and defect locations, and use the contour weights to predict the crack severity. Promising results are obtained on our manually annotated dataset, which indicate the viability of using this cost-effective system to perform road quality assessment at a municipal level.

Acknowledgements

I would like to thank my supervisor Professor John Zelek for all the guidance and help he offered me during our time working together. I would also like to deeply thank my thesis readers Professor Andrea Scott and Professor David Clausi for their invaluable input and aid. I would like to thank the Systems Design Department staff, especially Janine Blair, Lauren Gatchene, and Vicky Laurence, for all their help.

I would like to thank Public Sector Digest (PSD) for their support throughout the project and for their supplying of Google Street View Data to test on.

I would also like to thank Mitacs, the Ontario Graduate Scholarship program, and NSERC for their financial support.

Dedication

I dedicate this thesis to all my friends and family who supported me throughout this pursuit.

First and foremost, this is for Mom and Dad, and your unwavering support all throughout my life. Everything I am, ever will be, and much much more, I owe to you. This is for Salma and Walid, my biggest motivators and best friends. This is for Issam and Daniel, the two brightest stars in my life.

This is for all my Ammo's, Amto's , and Khelto's who always kept me motivated (and well fed). This is for all my cousins, nieces and nephews, and all the fun and adventure we shared. This is for Hanan, who still teaches me, guides me, and helps me when I go astray.

This is for my friends, whether from LES, AUB, Waterloo, or anywhere else, you're the family I never expected to have, but one I can't live without.

Table of Contents

List of Tables

List of Figures

List of Abbreviations and Terminology

ASCE - American Society of Civil Engineers

BEV - Bird's Eye View

CNN - Convolutional Neural Network

DSIFT - Dense Scale Invariant Feature Transform

F-measure - A measure of performance based on precision and accuracy

FNR - False Negative Rate

FPR - False Positive Rate

FV - Fisher Vectors

GMM - Gaussian Mixture Model

IMU - Inertial Measurement Unit

LCMS - Laser Crack Measurement System

LRIS - Laser Road Imaging System

LRMS - Laser Rut Measurement System

PCI - Pavement Condition Index

RCI - Roughness Condition Index

SCR - Surface Condition Rating

SIFT - Scale Invariant Feature Transform

SV - Street View

SVM - Support Vector Machine

UCM - Ultrametric Contour Map

Chapter 1

Introduction

Road maintenance is a significant concern at municipal, provincial, and national levels. In the United States alone, the American Society of Civil Engineers (ASCE) calculated that $91 billion a year is invested in road infrastructure [59]. However, at this rate of investment, roads are predicted to decline in quality, as just maintaining their current condition would require an additional $10 billion in investment, and improving overall road condition would require a further $69 billion dollars in investment [59]. This decline is also predicted to happen in Canada, as the average reinvestment rate in roads and bridges is around 50% of what is required to maintain them at high quality [13]. Roads must be kept in good condition to prevent damage to cars, car accidents, and potholes among others. The ASCE predicted that poor roads cost US motorists a total of $67 billion in additional operating and repair costs every year, which amounts to $324 a year per motorist and roughly 32% of America's roads are in poor or mediocre conditions [59].

Municipalities often have asset management plans in place that allow them to maintain their roads and, given that their budget is already stretched, allocate their funds properly. For Canadian municipalities, 71% perform data collection about their roads at least every five years[13]. Road quality assessment is currently usually done via surveyors who drive along the roads and check their conditions and can be manual, semi-automated or fully automated. This is an expensive and time-consuming task, especially in larger municipalities. We aim to use computer vision to further automate this task.

We can split road quality assessment into two main tasks: gathering road data and analyzing that data. Gathering data can be done in a multitude of ways which will be discussed further in this dissertation. Fortunately, there is a comprehensive and useful collection of data that is already publicly available, namely street view (SV) images. Road

image databases, like Google Street View, contain large amounts of data and are updated quite frequently, approximately every two years for large cities. These databases are accessible to the public, and we can automate mining of street view data to assess municipal assets and infrastructure.

The long-term vision is that other assets such as bridges, tunnels, and sidewalks (which are all a part of the transportation infrastructure) will be similarly mined. Google Street View data also provides a history of images that can be used to track the degradation of municipal assets and determine the optimal time and method of addressing the deterioration of these assets. The most common type of data in street view databases is natural images, i.e., ordinary color images. For our work, we limit the scope to road assessment and provide a demonstration of the applicability of the proposed method on this section of infrastructure with the intention to expand to other assets in the future.

On the data analysis side, we explore various methods to best generate an appropriate quality assessment for roads. We start off by exploring road segmentation from natural images. We first focused on a semantic segmentation, i.e., classifying every pixel in the image as a road or background pixel. This major step allows any road quality assessment algorithm to operate under the assumption that is only being fed road pixels, and classify the condition of said road accordingly. After segmenting the street view image we focus on its road portions and try to find cracks, potholes, or any other road degradation artefacts. Our main focus is on road texture, since it differs so much from the texture of other objects found in a typical street view image, and also since good quality road and damaged road can be differentiated using texture descriptors. As such, we built upon the describable texture dataset (DTD) methods which achieve the state-of-the-art in texture detection [11].

In brief, our algorithm for segmentation systematically samples windows from the input image densely and at multiple scales, then finds SIFT descriptors in those windows. These descriptors are then encoded using the Fisher Vector formulation [63]. The encoded features are then classified using an SVM classifier. This is followed by a window-by-window voting scheme. We then utilize an ultrametric contour map (UCM) [5], which is compared to an illumination invariant image (to prevent errors due to shadows) [21], and follow it up with a decision tree to preserve proper boundaries.

After the relatively complex segmentation step we end up with a new image which contains only road, and the quality assessment step is much simpler in comparison. The same features from the windows that were classified as road are then classified using a separately trained SVM, and separated into "good quality" and "damaged" road. We reapply the voting scheme to generate a new segmentation that detects road defects. These

defects can be detected even more precisely when we use the UCM inside damaged road regions.

Our segmentation algorithm achieves an F-measure (combination of precision and accuracy) of over 90% in perspective image space, making it a desirable and powerful input to the crack detection stage. The segmentation is tested on the KITTI Dataset[22], mainly to compare it to other methods, along with a subset of Google Street View Images which were manually collected and annotated, to demonstrate the method's effectiveness on a commercial SV database. The quality assessment algorithm also proves to be accurate and properly detects cracks and damaged road. We only perform our tests on the Google Street View subset, as we were able to annotate cracks and defects for training in this dataset. We tried to explore the segmentation and classification effectiveness of a small amount of carefully engineered features, versus the complex but comprehensive convolutional neural network architectures.

1.1 Contributions

The overall aim of this thesis is to describe, implement, and demonstrate a fully automated road quality assessment system that can be used on any natural image street view database.

1. We provide an alternative to the existing automated methods that are already in use, but require special equipment, such as expensive data collection vehicles, LIDAR data, among others, and build on existing camera-only methods both described in Chapter 2.

2. We review and discuss various existing segmentation methods, both road and non-road specific, and compare them to our algorithm (Chapter 2). We also discuss how we utilize a combination of them for key portions of our algorithm (Chapter 3). We discuss possible sources of street view images for our algorithm, and the benefits resulting from selecting each database(Chapter 2).

3. We propose a novel road segmentation algorithm based on engineered features that utilizes multi-scale patch classification by relying on texture features along with a spatial prior and Gaussian voting scheme (Chapter 3). We further improve on the segmentation by utilizing cues from an Ultrametric Contour Map and an illumination invariant image transformation. We test 4 different variants of our segmentation algorithm against other methods and discuss shadows, transfer learning, and the

effects of evaluating in the Bird's Eye View (BEV) transformed space in the context of road segmentation (Chapter 4).

4. We propose a road quality assessment algorithm that operates on a segmented image, and uses texture information to single out cracks and detect road deterioration artefacts (Chapter 4).

Two publications have resulted from the work presented in this thesis, mainly concerning road segmentation:

1. [8]: David Abou Chacra and John Zelek. Using superpixels for road segmentation in street view images. *Vision Letters*, 1(1), 2015

2. [9]: David Abou Chacra and John Zelek. Road segmentation in street view images using texture information. In *13th Conference on Computer and Robot Vision (CRV) 2016*, 2016

1.2 Thesis Outline

We will review relevant literature in Chapter 2, where we discuss the current state of road quality assessment in 2.1, natural image segmentation algorithms with a focus on road segmentation in 2.2, and street view databases in 2.3. We then discuss our algorithms in depth in Chapter 3 by first starting with our basic segmentation algorithm in section 3.1.1, then discussing the improved method utilizing the UCM in section 3.1.2, and the defect detection algorithm in section 3.2. We then discuss our experiments, results and observations in Chapter 4, focusing on road segmentation first in section 4.1 and crack detection in 4.2. We then conclude with our final discussions and conclusions in Chapter 5.

Chapter 2

Background

2.1 Road Quality Assessment

In this section, we will discuss the way road quality is currently assessed. The methods used can be split between manual and automated methods.

2.1.1 Manual Methods

Manual methods are somewhat primitive and quite time consuming and expensive. There is no single universal method to do road quality assessment, however, the process is smilar across different methods with the most variation coming about with the way the road defects are graded. Surveyors drive along every road in a municipality and note any defects they see[69], the severity is also noted. Operators are usually aided by a device made especially for this purpose, such as the VIZIROAD, shown in Figure 2.1, for them to quickly be able to mark the defects they see [61]. Additionally, some systems include an on-board IMU (inertial measurement unit) or accelerometer that records surface roughness [2]. Some systems even use a camera to record the road which allows viewing the data and evaluating it offline.

Some examples of types of surface distress are shown in Figure 2.2, these include [24]:

1. Alligator Cracks are a grid of interconnected cracks resembling the scales on an alligator. These are also referred to as fatigue cracks and are mainly due to poorly stabilized bases under the asphalt. They can be detected visually.

(a) (b)

Figure 2.1: (a) VIZIROAD is a system used by road surveyors to track road quality. (b) Various keys on the available keypad correspond to different road defects [32].

2. Longitudinal Cracks: Occur linearly in the direction of the road, which can happen due to the edges of underlying old pavement. Can be detected visually

3. Transverse Cracks occur perpendicular to the road direction, and can result from daily temperature cycles. They can be detected visually.

4. Potholes are bowl-shaped holes in the asphalt, which occur when alligator cracks are untreated and pavement is displaced by traffic. These can cause serious damage to vehicles. They can be detected visually.

5. Patching is where a portion of the asphalt which has been replaced and is mismatched with its surroundings. It can be the start of other types of distress. It can be detected visually, or noted in a prior assessment.

6. Rutting is the plastic deformation of the asphalt over time due to high loads, causing a shifting of pavement material away from the path of the vehicle wheels. It is difficult to detect visually, and must be detected using an IMU or other sensor.

The surface distress types are ranked based on their severity (low, medium, high), or on a point scale. Depending on the convention followed, different weights are assigned to different surface distress types and their severities, and these are used to determine

(a) Severe alligator cracks [2]

(b) Severe longitudinal crack [34]

(c) Severe transverse crack [2]

(d) Large pothole surrounded by alligator cracks [2]

(e) Road patching [35]

(f) Severe rutting [2]

Figure 2.2: Sample images of various types of road surface distress.

the surface condition rating (SCR). In the case where an IMU is present, a measure of roughness called the roughness condition index (RCI) is also calculated. The SCR and RCI are combined to generate the pavement condition index (PCI). The United States Department of Transportation follows the convention in equation 2.1 [2].

$$PCI = 0.6 * SCR + 0.4 * RCI \tag{2.1}$$

After the city is surveyed, operators compile collected data and give every city block a PCI based on the number and severity of defects in that block. The PCI is calculated differently depending on the firm conducting the assessment. This results in a map where stretches of road between intersections share a common condition index which roughly describes their condition. This alerts the municipality to roads that require maintenance. There are some non-visual methods as well, such as drilling into the road to take samples that can be tested in a lab; both visual and non-visual types of methods work equally well.

Road assessment is done in different time intervals depending on the agency conducting the evaluation and the municipality. Usually, it is done in intervals of one, two, or three years, but rarely more [50]. There are many flaws to this method. A condition index shared along a stretch of road is not truly indicative of its quality, a pothole in an otherwise good road, will have a similar PCI to a very poor quality road that is arguably more urgent to fix. Also, there are usually several surveyors that go around parts of the city, and their condition evaluations can vary significantly as surveyors can assess defects differently or even miss defects completely leading to variability between the cited PCI and the actual PCI [38]. More inefficiency is introduced when traversing larger roads and highways, as surveyors have to traverse these multiple times in the same direction, then in the other direction.

This method is still very popular and used in most road assessments. It is relatively archaic, but was very useful in the recent past when the technology was not yet available to automate this process. It gave municipalities useful insight and helped them allocate their budget properly. The output map is a potent representation of road quality, and our method aims to improve it by adding finer details that are more accurate and helpful in pinpointing problematic sections of road. Our automated method strives to be a more objective one, minimizing the subjectivity introduced by human error and observation. Municipalities can use this data to more efficiently plan road repairs. We will discuss our scheme to automate these methods in later Chapters 3 and 4.

2.1.2 Automated Methods

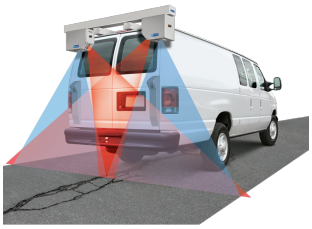
The frequency of performing road quality assessment led to a good deal of work aimed at automating it. The most basic of these methods are actually semi-automated, where their main aim is reducing the number of frames surveyors must sift through offline. They analyze the collected data, and flag frames as either containing distressed roads or not, like the work in [45] which uses adaptive thresholding to achieve this.

The prevalent automation methods involve attaching an apparatus to a truck that also drives along the city [6, 48, 61]. This apparatus can be attached to several locations on the vehicle, for example in tow, on the top or bottom of the vehicle. The main sensing portion contains a combination of one or more cameras pointed orthogonal to the road, LIDAR or LASER scanners, GPSs, IMU's, and accelerometers, among others. Notable among these are the Laser Road Imaging System (LRIS), Laser Crack Measurement System (LCMS), and Laser Rut Measurement System (LRMS) produced by Pavemetrics [62], shown in Figure 2.3 and Appendix A. The information from these sensing apparati is aggregated and different vision and signal processing methods are used to determine the pavement quality. These methods are quite powerful and yield very precise results.

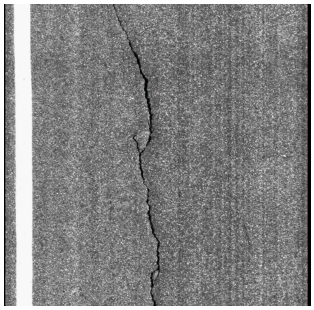
A significant amount of research work is aimed at perfecting fully-automated defect detection from primarily camera-collected data. Gabor filters have been convolved with images from an LRIS system, and cracks are found by thresholding the output of this convolution, which can pinpoint crack pixels with an average precision of 81% and an average recall of 84% [66]. In another approach, haralick texture features (based on statistical measurements) and an SVM were used to detect cracks as well [30]. Since different types of cracks and imperfections need to be handled differently, some methods are primarily aimed at correctly identifying the type of defect sensed. CrackIT [60, 61] merges multiple preprocessing techniques and clustering algorithms (K-means, Gaussian Mixture Models, among others) to detect cracks and their types and also yields favourable results with a 93.5% F-measure for their best performing algorithm .

Laser sensors allow for more advanced techniques utilizing 3D depth data to be used. The LCMS comes with proprietary software to detect defect type and severity [42], it utilizes all aspects of the LCMS, from the texture and depth information (to detect and measure crack depth), to the IMU information that detects rutting. In [72], a crack fundamental element is defined by detecting connecting cracks and clustering them into larger defects. Features of these clusters (such as the defect's total length, average and largest crack width, number of interconnected cracks) are used in a regression model that can classify the defect. Additional data from the accelerometer, for example, complements the other sensors, whereas a GPS is used to pinpoint the location where the sensory data was

(a) The LRIS sensors

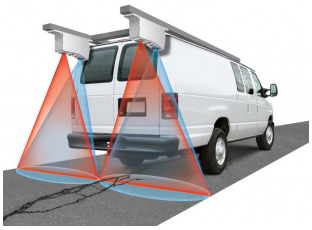


(b) The LRIS attached to a vehicle

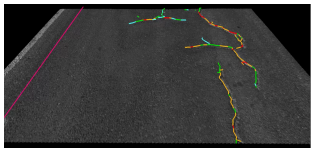


(c) Sample LRIS output

(d) The LCMS sensors



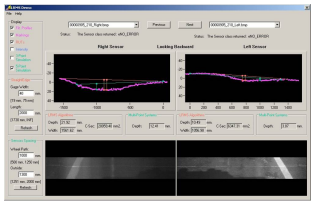
(e) The LCMS attached to a vehicle



(f) Sample LCMS output with marked cracks

(g) The LRMS sensors

(h) The LRMS attached to a vehicle



(i) The LRMS computer interface

Figure 2.3: The LRIS (a,b,c) captures high resolution 2D road data in a four meter-wide range, it utilizes line lasers to eliminate undesirable illumination effects. The LCMS (d,e,f) uses a combination of cameras and lasers to capture textured 3D road data. It is an all-encompassing road defect sensor that includes proprietary crack detection libaries that annotate cracks based on severity by utilizing the measured crack depth, and can detect rutting, potholes and other defects. The LRMS (g,h,i) uses lasers to detect road rutting specifically. Photos courtesy of [62]

captured.

Automated methods are being used more and more often, and, depending on the algorithm, they can be quite precise. The data analysis portion is automated and can be done offline. The main caveat is that the input sensory data can be expensive and tedious to procure. The sensing apparatus has a limited scope of vision, specifically 4 meters wide for the Pavemetrics devices [62]. It is usually "looking down" at the road (pointing orthogonal to the road), so the vehicle still needs to drive up and down every road lane in the area to be checked. This also means parts of the road that are not surveyed by the vehicle will not be sensed, like portions with parked cars, or portions blocked off, or even road shoulders. Sensing requires high resolution cameras (almost 1 pixel per millimetre of road in the case of the LRIS), and expensive LIDAR scanners, so there are usually only a few sensing vehicles that can do the sensing, which would require more time in large areas. Another drawback is that the raw data is somewhat large in size; the Pavemetrics devices use up to 1 GB per Km. While this method definitely has the benefit of having extra detail going into road analysis, it comes at a cost that is both monetary, in the form of expensive sensory devices in the range of a hundred thousand dollars, and temporal, in the form of long data acquisition and processing times.

We could only find a few other methods that utilized street view images for road quality assessment [75]. These earlier works [53, 54] utilize a simplified (and cheaper) version of the LIDAR methods mentioned above attached to city vehicles, namely buses and police cars, along with a dashboard camera. While interesting, it falls prey to sensor noise, however it is an order of a hundred times cheaper than the above methods. In their work, Varadharajan et al. use a dashboard camera and drive around the city of Pittsburgh, Pennsylvania to collect their own data, essentially similar to street view data [75]. They detect the ground plane, then over-segment it into superpixels and generate various descriptors for these superpixels which give way to a binary classifier used to identify cracks inside the superpixels. The superpixel is given a binary flag: either it contains a crack, or it does not. A different approach filters the input image rigorously by background subtraction followed by wavelet-based de-noising, then, locally thresholds patches from the image (by utilizing Otsu's method to quickly find these local thresholds) resulting in a binary image [6]. Horizontal and vertical histograms are used to classify the crack type; the system can distinguish between longitudinal, transverse and alligator cracks. These methods are effective, but slightly simplistic and can be improved with better road segmentation, and a more powerful classifier. A survey of other methods [73] also tests the performance of older methods, and highlights the fact that these of crack detection methods operate locally and can be fooled by the low signal-to-noise ratio (SNR) in pavement images.

Unfortunately, we could not find a standardized road defect dataset to train or test our

method, and we resorted to creating one ourselves by manually annotating defect regions in street view images. This prevented us from being able to compare our results to any of the methods presented in this portion of the review as the datasets they tested on are not publicly available. With this in mind, our work focuses on finding any and all road defects in an input image, with the knowledge that classifying these defects can be done per the methods mentioned here.

2.2 Road Segmentation from Natural Images

A logical step in emulating the road surveyor's visual method of road assessment is by first segmenting road from the input images. Detecting solely road pavement portions of an input image allows us to have a much simpler follow-up step in detecting cracks, as any method used for crack/imperfection detection will only be handling pixels it knows are road, and can be less stringent in its evaluation. This is arguably the most difficult part of the process to automate, as it ideally detects all the road pavement in the image and discards any other information. Cars, side-walks, trees, and pedestrians, among other frequently occurring objects in street view images, must be filtered out of the input image.

2.2.1 Road Segmentation Specific Methods

There are numerous methods that explicitly target road segmentation.

Road Lane Mark Detection

Many algorithms try to solve autonomous vehicle problems, and hence focus on road and lane detection using lane markers [4] and several other methods discussed in the taxonomy by Hillel et. al [29]. These methods utilize the fact that lane markers are inherently different in their appearance to the road itself, as they are differently coloured and can be continuous, dashed or even circular. Gradient-based approaches are quite popular for detecting the lane markings, for example steerable filters [49], which allow for detecting various lane markings using only three convolutions. This was done to aid in a driver assistance system that alerts the driver when they leave their lane. Simpler methods that use box-filters, among others, for detecting abrupt color changes also have been deployed [29]. A transformation into Birds Eye View (a projection of the pixels into what they would look like from above), which can be done using calibrated camera parameters allows these

systems to function without the perspective effect which can improve their performance in some cases.

Road Detection with Engineered Features

However, lane markers do not always exist, as such other algorithms try to go about without using this cue. Some research goes in the direction of predicting the vanishing point of a road [28, 39]. Other methods use 3D data to detect an abrupt change in elevation which happens between roads and curbs [29]. Other algorithms rely on detecting road features, namely color and texture [68], using additional sensor information [78], or using a spatial prior [3] to predict a road's location in the image. One particular method of interest is the structured random forests (SRF) method which uses a structured random forest is trained on texture, illumination invariant, color, and location features of image patches [81]. Patches are then classified using this forest, and road probability map is generated by accumulating individual patch probabilities. This method is the most similar to our segmentation method since its features are engineered and then passed through a trained classifier. It performs well on the KITTI dataset attaining an average F-measure of 82.44%.

Road Detection with Learned Features

The DDN [55] algorithm uses deep deconvolutional networks for its segmentation, essentially it uses convolutional neural networks (CNNs) to extract features, and then attempts to go back to the original input image by de-convolving those features, followed by a fully connected layer which is trained to segment based on the de-convolved features. It achieves the state of the art on the KITTI dataset with an F-measure of 93.65%. The hierarchical inference machine (HIM) approach [57] splits the image into multiple class inference problems on a fine to coarse scale and is also one of the top perofmers on the KITTI dataset, with an F-measure of 90.07%. The Fused CRF method [80] utilizes additional LIDAR sensor data, and maps this data to the image pixels, this is followed by using a CRF and then a graph-cut method to get the final segmentation. All three of these methods achieve top performance on the KITTI road segmentation dataset. Another top performing method [52] utilizes fully convolutional neural networks to classify patches from the input image and generate a binary label for each patch. The results of the methods benchmarked on the KITTI dataset are shown in Table 4.3 and comparatively discussed in Chapter 4. We note that this is not an exhaustive taxonomy of road segmentation algorithms, but covers a lot of various approaches to this problem.

2.2.2 General Natural Image Segmentation Methods

Image segmentation is an open problem that continuously gets attention from scientists in various fields. A significant portion of research addresses the problem of segmentation in general, from a natural image, with little to no knowledge of its contents in advance.

Classical Segmentation Methods

Classical approaches tend to rely very heavily on mathematical models. In a sense, they were very much engineered approaches to segmentation. Graph cut methods [7] were extremely powerful for their time, and still get used and improved on today. These algorithms find a segmentation using a min-cut/max-flow framework and an energy function of some sort, like the Gibb's function for example. Grab-Cut [65], is an improvement by that utilizes Gaussian Mixture Models and runs iteratively to generate a reasonably good segmentation utilizing pixel color information. Some research has gone into expanding it to use other cues, such as texture [51, 33]. More recent methods tried to segment the image into small, yet meaningful portions called superpixels [1, 74]. Good superpixels can be assumed to belong to the same class in a segmentation, and often precede the segmentation as a pre-processing step. While these classical methods are computationally efficient and well engineered, newer methods, especially ones relying on deep learning, have outperformed them.

Segmentation Using Convolutional Neural Networks

Given the huge rise in computing power over the last decade, learning-based approaches have gained immence popularity in image processing, and with good reason. Introduced in 1995 [43], Convolutional Neural Networks (CNN's) work by convolving windows in an input image by the same filters; these filters are learned by training this network. Larger and larger windows of the image are convolved deeper in the network due to its max-pooling layers. Deep CNN's (DCNN's), which are CNN's with several hidden convolution and pooling layers, recently gained popularity after they performed extremely well on image classification tasks, specifically the image-net challege [41] outperforming the previous state of the art by a more than ten percent. Much like in the classic implementation of the convolutional neural network [43], in the image-net network the convolutional layers convolve local window inputs with a bank of filters then apply a certain point-wise non-linearity. The fully connected layers each apply linear filters to all inputs from the previous layer before applying another point-wise non-linearity (see Figure 2.4). Some key features

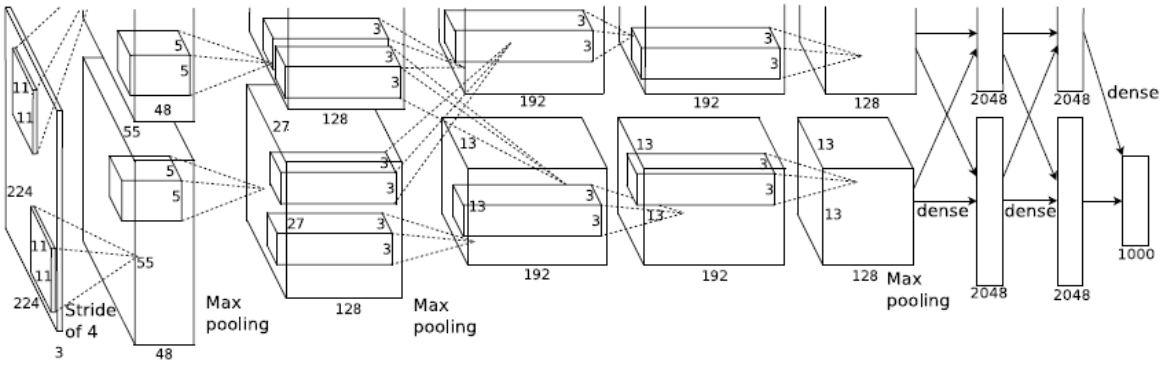


Figure 2.4: A visualization of a portion of the deep convolutional neural network known as AlexNet from [41]. The layers shown (from left to right) are the input layer which is a 256x256 image, the first convolutional layer that takes the largest chunks of the image, and the second to fifth convolution layers that take smaller chunks respectively. In between these layers is a max pooling step to preserve global performance. The final 3 layers are fully connected and apply directly to the image-net challenge on which this network was trained.

of this network, dubbed 'Alexnet' after its creator, is that it is trained with Rectified Linear Unites (ReLUs) that allow for faster neural network training, it also has a GPU application which is convenient for training and usage, it uses overlapping pooling which reduces overfitting, along with using Data augmentation (different orientations, scales, reflections) and the dropout method to reduce overfitting.

The method described of Chen et al. [10] utilizes DCNN's, the authors coupled the DCNN's responses at the final layer with a Conditional Random Field, which allowed them to produce competitive results. A similar method was employed by Long et al. [46], where the authors modified the pretrained image-net model into a fully convolutional network, where the last layer was a deconvolution layer that produced a semantic segmentation. Another approach uses a CNN at multiple scales to extract local features in a pixel's vicinity [18], these results are then coupled with a contour preserving segmentation (superpixels or segmentation trees) to create the final segmentation. Since these methods are learning methods they could be trained to optimize their performance on specific tasks such as road segmentation.

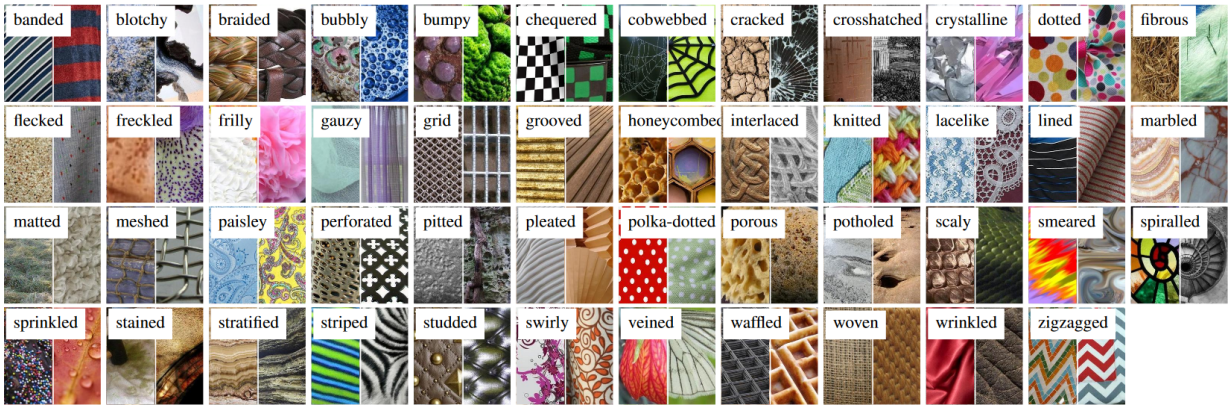


Figure 2.5: Sample images from the 47 different classes that were separable using the DTD method [11].

Texture-Based Segmentation

Texture information can vastly improve the quality of a segmentation. A thorough review of the various methods that incorporate textures into segmentation is presented by Ilea and Whelan [33]. These methods vary from simple, to more complicated. Some use color and texture simultaneously, and some operate on each of them independently and merge the results. Another notable split is the use of statistical methods like K-means and fuzzy clustering versus probabilistic approaches such as Markov random fields.

One texture classification method that seemed appropriate for out problem is the "describing textures in the wild" method [11]. The authors use Fisher Vectors to encode SIFT features from images, then train a simple classifier on these vectors. They show that this can be a robust and powerful method of texture classification. After training, they are successfully able to separate between forty seven different texture categories (see Figure 2.5) and achieve the state of the art on various texture datasets, outperforming the reviewed methods of Ilea and Whelan [33]. This method is one of the main methods we use for our final segmentation, and we will discuss it in further detail in the coming section. An improved version of this texture classifier [12] utilizes convolutional neural networks for feature detection as opposed to SIFT descriptors. We opted for the original method since we wanted to compare how engineered features for segmentation fare against learned methods.

2.2.3 Ultrametric Contour Maps for Edge Detection

Edges can be quite the informative cue, and can guide a segmentation well. A very useful algorithm for contour detection is proposed in [5], which attempts to segment an image by first finding and weighting all the edges in an image, and forming what is called an Ultrametric Contour Map. It achieves the state of the art on multiple edge detection datasets such as the BSDS500 and NYU depth dataset[5]. We will restrict our discussion on edge detection to this method.

A faster method to find the UCM [17, 84] builds on previous work [44], which utilizes local edge structure to find the UCM. This algorithm combines structured learning with random decision trees to detect edges, and takes advantage of the fact that local edges exhibit certain structures. Notably, this method allows edge detection to be done in real time allowing it to be integrated into our system at little computational cost, and its details provided valuable insight that help us with our own problem.

Structured Learning

Structured learning approaches were shown to have a significant impact in image processing applications [58]. The authors used learning methods like SVMs and Random Fields on undirected graphical models to model complex relationships between variables in data and their interactions, and learn the parameters of those models. This allowed for learning mappings between complex input and output spaces.

In their approach, however, Dollar and Zitnick [17] deviate from the structured learning methods discussed in [58]. They use a standard input space (images) and a structured output space (specific edge patches). Their outputs are only edges that have been previously encountered in training images.

Sketch Tokens

The inherent structure in edges was first noticed and learned in an earlier publication [44]. Here, the authors learned the structures of possible edges using hand-drawn edges on training images. The result was 35x35 pixel windows with the center pixel being part of an edge. These windows, dubbed 'sketch tokens' were clustered using K-means into 150 categories. They reflect what shapes the edges in natural images can possibly take. This is visualized in Figure 2.6.

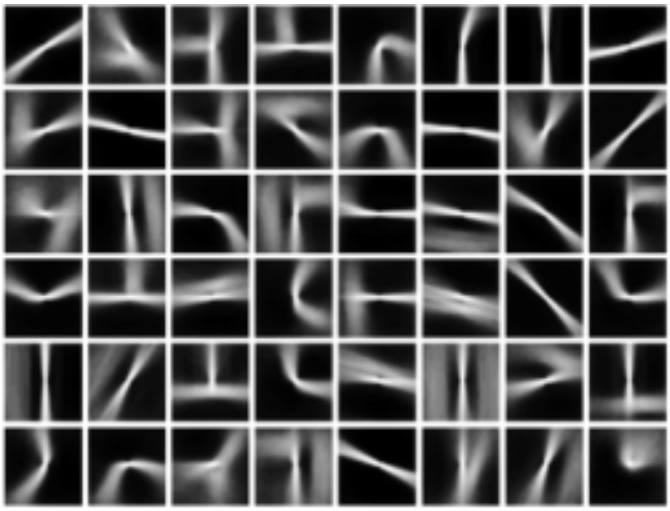


Figure 2.6: Sample sketch tokens taken from the aggregation of human labelling of multiple image edges.

Random Forests

A decision tree is a simple classification method that works by making a series of binary decisions [16]. For an input x, the tree starts at the root, and keeps moving x down the tree, choosing either a left or a right leaf node to go to based on a binary split function. The end result is a leaf node at the end of the tree with no further leaf nodes. When x reaches that final leaf node, either it is classified into a certain label y, or distribution over multiple labels Y. Sometimes the split function is simply comparing x to a threshold, called a 'stump'.

A decision forest is when x is classified by more than 1 tree, and the ensemble result is taken. This can be done by doing a majority vote on the result of several trees, or an average, or even more complex models. Decision trees were shown to exhibit high variance and over fitting, so the idea of training multiple trees on random training data de-correlates the trees, and produces a more reliable output by combining the individual tree's output.

We also apply decision trees, specifically a variant called purity trees [18], in a separate part of our algorithm.

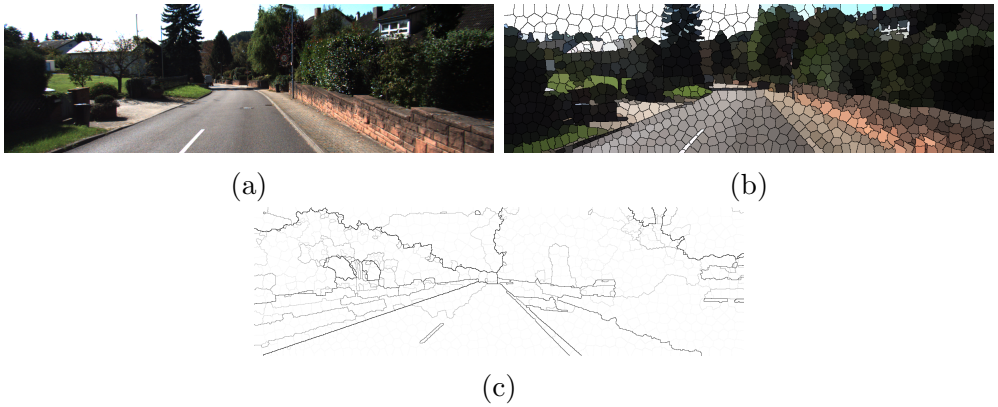


Figure 2.7: (a) The original image, and (b) its computed superpixels and (c) Ultrametric Contour Map, using the microsoft algorithm [17]. Both these edge methods preserve natural edge boundaries in the image, which serve as powerful cues to aid in road segmentation.

Structured Forests Algorithm

The structured forests algorithm is based on the three concepts described earlier. The problem becomes not whether to output whether a pixel lies on an edge or not, but what kind of structure could the edge the pixel lies on have. Thus, structured forests approach takes in a 16x16 mask from the input image at multiple scales and orientations. Then, for every mask x and all patches, it maps this mask into a new feature space where similarity measures are easy to compute (Euclidean distance) and finds the difference between this mask x and a label y [17].

The decision trees are trained to classify these input patches based on a human-defined separation of edge patches. So each leaf node produces lower and lower entropy splits, and edge patches look more and more similar. Note that the dimensionality of the problem is decreased by utilizing the sketch tokens, so instead of being a 2256 dimensional problem (of all possible combinations of pixels in a 16x16 image patch), the problem is reduced to 151 dimensions (150 'positive' edge patches and 1 non-edge patch that summarizes everything else). Since they are taking 16x16 masks, each pixel actually gets 256/4=64 votes per tree used. The number of trees used is 1-4. Furthermore the ensemble model that combines outputs is a medoid function. The end result of the structured forest algorithm is the UCM. The UCM provides valuable edge information, in that edge weights can be thresholded to generate different segmentations. This method is also capable of producing accurate superpixels compared to SLIC [1], SEEDS [74] and others.

Shadow Removal using Intrinsic Images

Despite all its merits, the UCM still falls into the trapping of falsely detecting shadows as edges. Unfortunately, in the case of strong shadows , these edges carry large weights in the UCM and can lead a segmentation astray. Several off the shelf shadow removal methods claim to be able to remove shadows from natural images with no prior knowledge. We chose to use the method of finding intrinsic images to reduce the effect of shadows. First proposed in [21], this method for shadow removal operates by first converting the input the image into log-chromacity space. Then the authors note that for a specific camera (under the assumptions that the lighting is planckian, the surfaces in the image are lambertian surfaces, and the camera sensors are sufficiently narrowband) all data in log-chromacity space is ideally required to fall on two parallel lines. Once the direction of these lines (the invariant angle) is known, we can remove shadows by projecting in that direction. The authors then showed that this method works regardless of whether the assumptions are true.

Initially, the invariant angle was found by calibrating the camera, however in later works [20, 19] the authors found a method that allows them to find this angle for any image from any uncalibrated camera. This is done by entropy minimization, where data from the log-chromacity space is projected along all possible lines (the angle ranges from 0°to 180°). The angle that minimizes the entropy is then chosen as the invariant angle of projection, and the shadows are removed by projecting along this angle. One very useful result of this property is that for images in a database that are taken using the same camera (which is the case for the KITTI dataset, and even Google Street View), the invariant angle is the same. This method was used by several works to remove shadows from road images specifially, for path planning [40] and outdoor localization [15].

2.3 Street View Data

Images obtained at street level are indispensable in emulating the visual methods used for road assessment, and fortunately, street view databases are very popular. We are not the first to tap into this goldmine of data, as researchers have used street view data in numerous other projects as they provide real-world data that can be a key component of a numerous applications.

2.3.1 Street View Databases

Commercial Databases

The largest street view databases are created and maintained by large companies. This industry is very competitive and includes Google Street View, Bing Street Side, Here street view (backed by BMW, Audi and Daimler), Mapillary Street Level Photos (a crowd sourced effort) among many others. These databases offer data from around the globe at high image resolutions. They are well maintained, and frequently updated, which is especially true of Google Street View.

While these databases can be accessed by the public, and even mined (a morally dubious endeavour) using their respective APIs and a data scraper, they do not provide their data at a high resolution for free. Google, for example, has a paid model that gets cheaper with the number of images purchased. Moreover, these databases only contain the raw data without any ground truth annotations for segmentation or road quality assessment, meaning that a training data for any learning algorithm must be generated manually.

Academic Databases

Given that many academics are using street view data in their research, some academic resources have generated street view data to aid in this work. The two main datasets are the KITTI dataset [22] and the Cityscapes dataset [14]. The KITTI dataset is slightly older, but has been used extensively by numerous researchers, for a while it was the only annotated street view dataset available to academics, hence its popularity. It consists of data captured using a camera rig on top of a vehicle driving around the city of Karlsruhe, Germany. Stereo data is captured, and there are several labelled subsets which can be used in scene flow, visual odometry, object tracking, and segmentation. The Cityscapes dataset was recently released and provides high resolution and more finely annotated data from numerous cities in and around Germany. The main purpose behind it is giving researchers data to train and benchmark more complex models (like Deep Convolutional Neural Networks) on. The images contain ground truth annotations for 30 different classes that can be seen in street view images.

A lot of effort has been put into creating both of these databases, and making them publicly available at their full resolution and at no charge. We opted to use the KITTI dataset for our road segmentation tests given that it only contains two classes (road and not road), and we are only interested in segmenting roads. This allows us to compare our

road segmentation results with other methods, which would not have been possible on the Cityscapes dataset due it containing 30 classes that must be semantically labelled.

Collected Databases

As mentioned in [54], data can actually be collected quite easily. A dashboard camera rig with a GPS is more than enough to get a full view of the road in front and geo-tag its location. When attached to public vehicles that regularly go around the city, like garbage trucks, buses and police cars, these rigs provide the required street view data at a very low cost. We did collect some data ourselves, but opted not to use it as it required manual annotation for training.

2.3.2 Other Research utilizing Street View Data

Street view data has been mined for the detection of how accessible wheelchair ramps are in certain geographic regions [27]. This application started off as manual labeling, but then was automated using machine intelligence methods. Image segmentation was also done on the database [79]. The authors segmented road images semantically into various labels: road, building, car, sky, and pedestrian. The ability to localize and detect objects in Street view data has also been recently addressed [82]; in the process this group has also collected a standard database of Google Street View images that can be used for evaluating algorithms. They used nearest neighbor matching to geolocate the images. Street view imagery has also been used to aid visually impaired bus riders to find bus stops in new locations [26].

The abundance of street view data has also been capitalized upon for learning methods for driver assistance systems, and are being tapped into by self-driving car researchers [67]. Here, Google Street View has been used for training various obstacle avoidance and detection systems. In an interesting application, street view imagery has also been used to classify street view storefronts to determine the type of business present [56]. Text recognition, in particular text appearing in street signs has also been another area of activity [71, 36]. Given the prevalence of imagery available, it only makes sense that this extensive database has also been used for the navigation and geolocalization based on cell phone images [82, 70]. The assessment of the condition of sidewalks has also been discussed [25].

2.4 Summary

Manual road quality assessment, done via surveyors, has many shortcomings: from subjectivity in assessment, and lack of precision in pinpointing road defects to a high cost both in terms of money and time. Automated collection methods that rely on specific equiment with LIDAR scanners and high resolution cameras, as in the case of the LCMS [42] yield precise and consistent results, but fail to sense defects in roads they cannot reach or detect and come at a sizeable monetary cost which is not feasible for smaller municipalities. Some automated methods rely solely on camera data which is cheap to acquire and attempt to emulate the results of the human surveyors [53, 75, 6], these methods bridge the gap between the manual assessment and the more expensive automated methods however they exhibit certain drawbacks such as classifying too broadly in some cases (classifying a superpixel as containing a defect or not), operate on local regions only, or get fooled by noise. We aim to improve upon the camera-only methods by focusing on segmenting the road pixels in an image accurately first by leveraging texture information inherent to roads, and then using this same texture information to find the regions containing road defects and zero in on those defects by utilizing contour information.

With the eventual aim of classifying road quality, we found that texture information in images is a fundamental cue. While various texture-based segmentation methods exist [33], we base our segmentation on the texture classification method utilizing fisher vectors [11, 63, 64] that leverages engineered SIFT features and expand it to a segmentation method. We further improve the segmentation by utilizing a UCM [17], which outputs a weighted contour map that corresponds well to crack severity, as well as a purity tree [18]. We conveniently use the same engineered features for road segmentation and crack detection, and only train a different SVM classifier for the two tasks. We test our segmentation method on an academic street view dataset [22] and demonstrate segmentation and crack detection on a subset of Google Street View.

Chapter 3

Algorithm and Methodology

Our segmentation algorithm is visualized in Figure 3.1, and our crack detection algorithm is visualized in Figure 3.2. This chapter is an overview where we discuss the building blocks of both algorithms, we provide more details and implementation specifics in chapter 4.

3.1 Road Segmentation

3.1.1 Basic Method

Our basic method utilizes Fisher Vectors calculated for Dense SIFT features, extracted on image windows at multiple scales. This is followed by support vector machine classification of these Fisher Vectors, and hence the patch they represent, and a voting scheme to generate the basic segmentation. As shown in Chapter 4, this method is sufficient as an input to the crack detection method.

Multi-Scale Window Extraction

We first extract small windows from the original image while ensuring some overlap between neighbouring windows. The extracted patches are simply small images that will be classified to achieve the final segmentation. Window extraction is done at multiple scales to allow global cues to influence the segmentation.

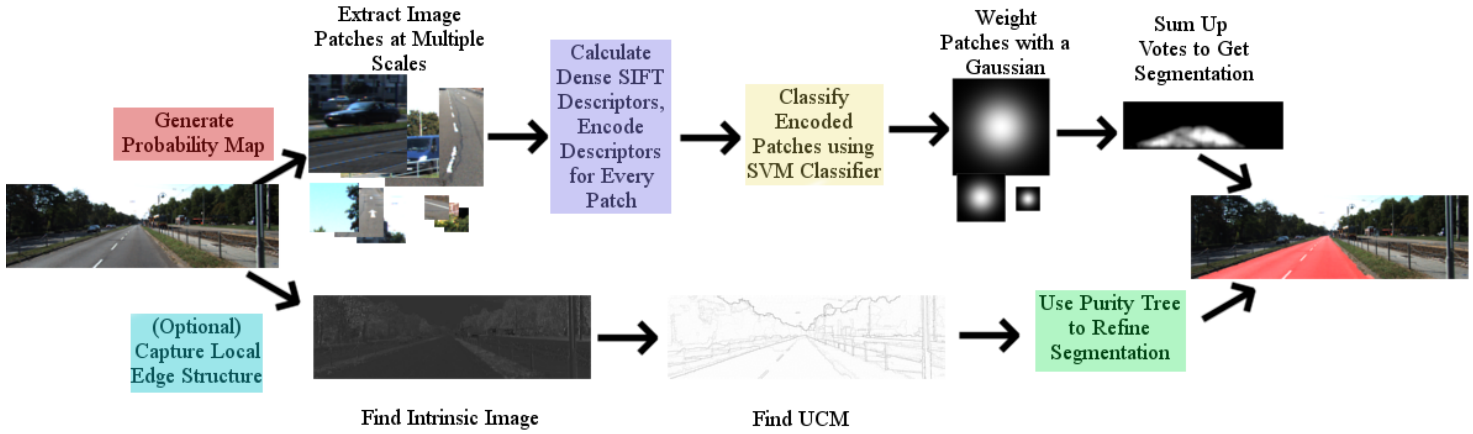


Figure 3.1: A visualization of our segmentation algorithm. We perform our segmentation by first extracting image windows (or patches) densely and at multiple scales from the original large image. These windows are then classified as either a 'road' or 'background' patch, using Fisher vectors computed on SIFT descriptors then classified with an SVM classifier. Based on the result of the classification and a precomputed spatial prior, the patches are assigned a weighted Gaussian mask that spans their respective size. The Gaussian masks for all the patches, and all the scales, are then added onto their window's original position in the large image. Finally, we optionally combine this result with that of an edge detector or superpixel method to generate our final segmentation.

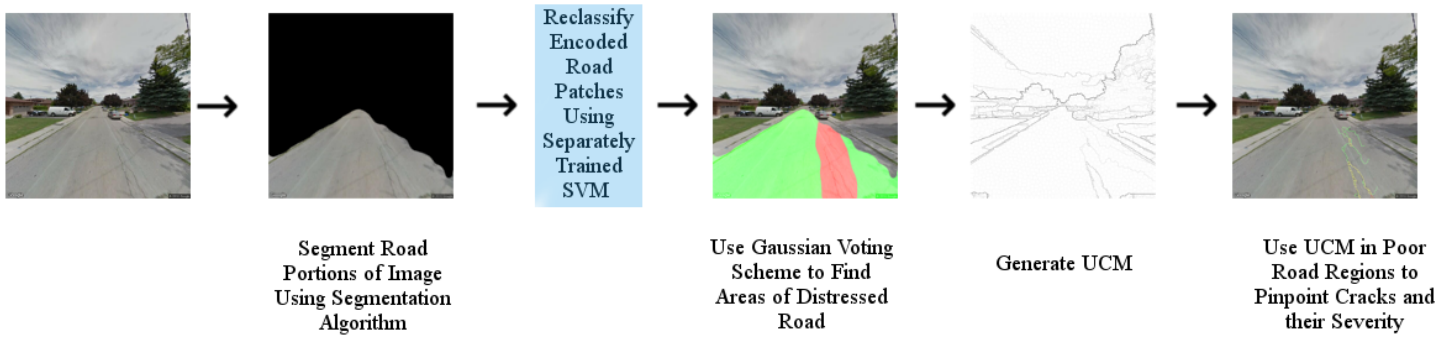


Figure 3.2: A visualization of our crack detection algorithm. We first segment the image into 'road and 'background' using our segmentation algorithm. We then single out the Fisher vectors of the road pixels only, which were already computed during the segmentation step. The Fisher vectors are run through an SVM to classify the road region into 'good road' and 'distressed road'. An ultrametric contour map is computed for the image, and cracks are detected by finding locations of high UCM response within 'distressed road' regions. These cracks are further distinguished by color-coding them according to severity which is also deduced from the UCM response.

Dense SIFT Features

Scale-invariant feature transform (SIFT) descriptors, which compute a histogram of gradients based on pixel magnitude and orientation, have proven very valuable in object classification [47]. Dense SIFT (DSIFT) is an extension of SIFT, where the SIFT descriptors are calculated densely over a specified grid in the image rather than at 'key points' found by an interest operator [76]. When computed on the extracted patches, they allow us to capture local image features and use them with the Fisher Vector formulation.

Fisher Vectors

Fisher vectors can act on local descriptors computed on an image, in our case DSIFT descriptors found on the image patches. The key lies in their use of a Gaussian Mixture Model (GMM) to quantize these descriptors, followed by calculating the gradient of the log-likelihood with respect to the GMM parameters [63]. In the context of texture detection, they outperform the Bag of Visual Words method (BOVW), of which they are an extension of, as they use a gradient representation rather than a histogram of occurrences of BOVW [11]. Each Gaussian in the GMM represents a visual word, which has been shown to be quite useful in properly quantizing descriptors to achieve a separable space. Fisher Vectors are described as a method that combines the benefit of a generative model (because of their use of GMMs), along with discriminative models, as they can be directly used in a linear classifier. In our work we use the improved Fisher vector (IFV) formulation [64], which adds an extra step of normalization as it allows us to directly use the output in an SVM classifier, and was shown to yield excellent results in texture classification [11].

SVM Classifier

The IFV formulation produces a vector that can be separated with a linear classifier such as an SVM classifier with a linear kernel. While several other discriminative classifiers may have also worked well, we chose to go with an SVM, given it is a fast classifier when using a linear kernel. This reliable performance, at a low computational cost due to not using a complex SVM kernel, is useful since we train on hundreds of thousands of image patches. We input the Fisher Vectors which describe an image patch into the SVM classifier, and obtain how well this patch corresponds to the trained classes of 'road' and 'not road'.

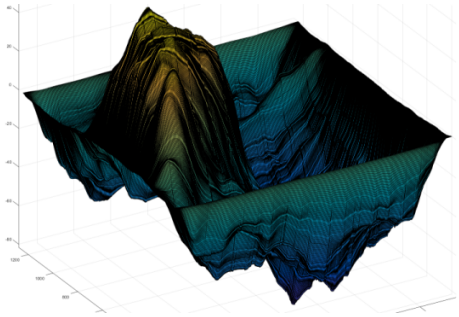


Figure 3.3: The Gaussian voting windows contribute to creating a probability map similar to a topographical map where higher 'peaks' are more likely to be road pixels, and lower 'valleys' are more likely to not be road pixels.

Gaussian Window Voting Scheme

The SVM output for every patch is indicative of how close it is to each of the two classes. This output can be used to give a measure of how likely the portion of the original image which the window was extracted from belongs to one class or another. Depending on which class it belonged, the patch is given a positive, or a negative weight. The patch weights are summed up across the different scales to produce a probability map similar to a 3D surface where positive values of this surface correspond to the road in the image (see Figure 3.3). This voting scheme produces a more accurate segmentation, as a pixel's class is heavily influenced by its surroundings, so instead of having a single vote for every pixel, it receives numerous votes to determine its class.

On its own, this information would result in a 'blocky' segmentation. We found that representing the patch with a 2D Gaussian function centered on its center pixel was a better method, as it was more representative of the truth, which is: the center of the patch is most likely to belong to the class that was predicted by the SVM, and this probability decreases as we move further from this center towards any of the edges. This produced smoother and better performing results on early testing data, and was the methodology adopted.

Spatial Prior

One key observation that can be made on the data is that the concentration of road pixels is in a specific part of the image. For example, there are never road pixels occurring at the

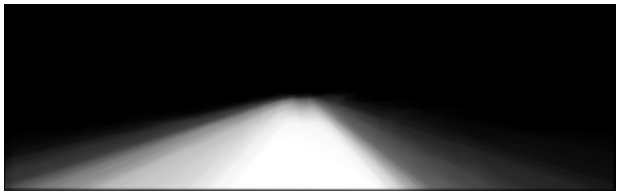


Figure 3.4: The spatial prior is generated for the road class by calculating the times a pixel adopted the road class in the training set. The segmentation relies on the prior to weight its Gaussian windows based on the location they are applied to.

top of the image, whereas they frequently occur towards the bottom part of the image. If we aggregate the ground truth for all images we have, we obtain a probability map of where road pixels can possibly occur in a test image, based on how frequently they occurred in a certain position in our training set. This is what we refer to as a spatial prior, it is shown in Figure 3.4 and is a key step in reducing error from erroneously classified pixels.

The prior is used to weight every Gaussian window added to the segmentation. The weight is generated by comparing the Gaussian to the pixels in the same location in the spatial prior.

3.1.2 Improved Method

We attempt to improve our basic method by considering edges and how they relate to possible regions in the image.

Shadow Removal

Shadows do not affect the FV and SVM side of the segmentation since SIFT features are partially illumination invariant, and also since they are learned by the SVM. An edge detector, however, is very susceptible to shadows, as they usually form strong edges in natural images. We decided to use the intrinsic image method to remove shadows [20]. This method is extremely fast, as once the invariant angle of a camera is known, it allows us to find intrinsic images of all images produced by that camera. In the case of KITTI, the database is entirely generated using the same camera, therefore the invariant angle is the same for the entirety of it. We find the invariant image by simply projecting along that angle in log-chromacity space.

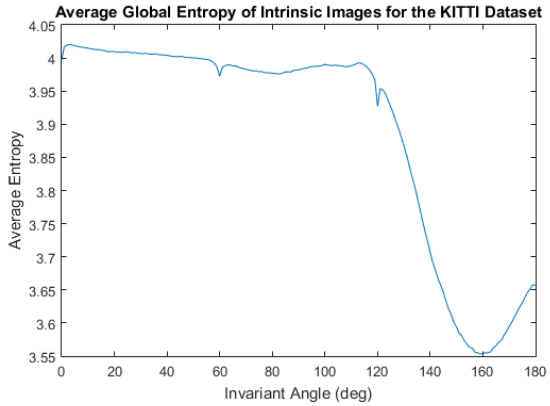


Figure 3.5: A plot of the average entropy of the invariant images projected at all angles from the KITTI training set. The minima is found to be at 160°.

While entropy minimization can be done on an image by image basis to find this invariant angle (and intrinsic image generation follows), the calculated angle is offset by noise in the image. To combat this, we perform entropy minimization on the entire training set of the KITTI database, and find the proper invariant angle of the KITTI camera that minimized the global entropy. The normalized global entropy is shown in Figure 3.5, and its minima is calculated to occur at 160°.

Superpixels or Ultrametric Contour Map

We further improve our basic segmentation by using a weighted edge map (like a UCM) or superpixels computed on the image. Accurate superpixels, namely those generated using the Structured Forests method in [17], preserve natural edge boundaries in the original image. Alternatively, the UCM goes beyond a regular edge map by weighting edges, thereby giving a likelihood that the edge in question corresponds to a real boundary between the two classes in an image. When combined with the segmentation found earlier, we can generate a better structured segmentation that utilizes this edge information. The UCM cannot distinguish shadow edges from object edges, so to properly utilize it, a shadow removal step has to be applied beforehand.

Purity Tree

The UCM is actually a hierarchical map, in that edges form closed boundaries. This property lends itself very nicely to the purity tree optimization step done on an oversegmentation output to generate a better segmentation [18]. We specify the pixels as the leaf nodes, and move up the tree by finding segments enclosed by increasing UCM weights to generate a tree. Using the segmentation generated by the basic method, we find the entropy of every branch based on how well the classes of pixels it clusters together agree, and weight it based on the weight of the UCM edge surrounding it. We first move up through the tree to generate this value for all branches, starting with the superpixel segmentation (which is the UCM at its lowest threshold). We then move down the tree and determine the optimal cut. Our method favours less entropic cuts surrounded by a high UCM weight.

3.2 Road Quality Detection

After segmenting the road from the non-road pixels, we can now focus on finding cracks, potholes and other road artefacts that occur in roads of bad quality. Road quality detection is now very simple and fast to achieve, and it utilizes data already found in the segmentation step.

3.2.1 Reclassify Road Patches

By design, the segmentation step is conservative, and usually only retains image portions it is highly confident are actually road. We can therefore operate under the assumption that any non-road objects have been removed. We train another SVM classifier only on the Fisher vectors computed on the DSIFT features of the segmented road. This time, our two classes are reflective of road quality, 'good road' and 'distressed road'. The road patches are now classified based on the quality of road they contain. We follow this step with the same weighted Gaussian voting scheme mentioned above to get a 'segmentation' based on road quality. This step is much faster due to us reusing the already evaluated feature vectors for every patch.

3.2.2 Crack Detection

We can further improve our road quality detection by pinpointing cracks in the road. This can be done using the precomputed UCM. We simply detect strong UCM responses inside

regions classified as 'distressed road' to focus on the cracks present in the road. The weight of the UCM at a crack is actually reflective of its severity, where larger cracks usually illicit a high UCM weight.

Chapter 4

Experiments

4.1 Segmentation of Images into Road and Non-Road

4.1.1 Original Method

All our testing and experimentation was conducted on the KITTI Road Detection Dataset[1] [22]. The dataset consists of 289 training images for which ground truth is provided, and 290 testing images for which the ground truth is not given. It contains a mix of urban roads, both small and large, a third of which have no lane or boundary markings. The images are approximately 1242x375 pixels in size, but vary slightly by image. Evaluation on the testing data is done on the KITTI server by uploading the results obtained. We also tested our method on a subset of images from Google Street View. These images were obtained by querying the Google Street View API in the Hamilton region. Overall, we obtained 250 images, 150 were used for training, 50 for validation, and 50 for testing. We annotated the images manually using the MATLAB tool LibLabel [23].

Our experiments were carried out as described in Section 3. We sampled patches of three sizes: 45x45 pixels, 91x91 pixels and 181x181 pixels. These square patches were taken from the full resolution image. We initially planned on taking same-sized patches at different scales, however the low image resolution of the datasets (1272 x 375 for the KITTI dataset and 640 x 640 for the Google Street View subset) forced us to instead sample patches of different sizes to maintain what little detail exists in those patches. The

[1]The KITTI dataset for road segmentation can be found at: http://www.cvlibs.net/datasets/kitti/eval_road.php

patch sizes were chosen after testing on several different sizes, and finding that a 45x45 patch is almost as small as we can go to capture finer details, while still having enough information in the patches to create a separable space for classification.

The windows were sampled every 20 pixels (both in the vertical and horizontal directions) for the 45x45 squares, every 32 pixels for the 91x91 squares, and every 64 pixels for the 181x181 squares. This skip length was chosen so that there is significant overlap between neighboring windows, to allow for better sampling of the image space, and make the voting scheme more efficient. Memory constraints of our hardware testing platform prevented us from sampling the 45x45 squares every 16 pixels. We chose to make patch sampling significantly fine to eliminate disproportionate voting that occurred when the skip length between consecutively sampled windows was large.

DSIFT features were extracted using the functions provided in the VLfeat toolbox [76]. We followed the values that proved effective in [11], and chose to sample descriptors every two pixels at scales $2^{i/3}(i = 0, 1, 2 \ldots \text{etc})$, with spatial bins extending 6x6 pixels. We also used the Fisher vector encoding function implemented in the VLFeat toolbox, with 100 visual words and 20 PCA dimensions for the 45x45 patches, and 256 visual words and 60 PCA dimensions for the 91x91 and 181x181 patches. An increase in visual words did not affect our results greatly, as even the 181x181 patches are still relatively small, and using more words only improved experimental results slightly.

We trained an SVM as our classifier. We used a linear SVM kernel due to it being significantly faster to train compared to other methods (or a non-linear SVM kernel) on a large number of data points while yielding excellent results on texture classification [11]. We also used the SVM implementation in the VLFeat toolbox [76].

The Gaussian windows are the same size as the patches they represent in the original image. These Gaussians are centered on the center pixel in the patch's selection window. All the Gaussian votes are summed to get a 3D topographical map where negative z-values indicate that a pixel is more likely to belong to the 'not road' class and positive z-values indicate that a pixel is more likely to belong to the 'road' class. The Gaussians are weighed differently based on size, their predicted class and their location in the image. We determined the best performing weights to be 2 for the 45x45 patches, 3 for the 91x91 patches and 1 for the 181x181 patches, hence a vote coming from a 91x91 patch is weighted three times more than a vote from the 181x181 patch. This is slightly counter-intuitive, and we initially tried the logical weights of 1 for the 45x45 patches, 4 for the 91x91 patches (since they fit four 45x45 patches inside them at a skip length of 16 pixels), and 16 for the 181x181 patches (as they fit four 91x91 patches and sixteen 45x45 patches), but obtained a five percent improvement from the modified weights we eventually used. Furthermore,

the standard deviation of every Gaussian is set to a fifth of its size, which also gave good results experimentally.

The above mentioned weights are not the only values weighting the Gaussians, since, for some experiments, we also took the spatial prior into consideration. First, a spatial prior as seen in Figure 3.4 is created from the training data. We used a percentage reflecting how frequently a certain pixel in the training images is a road pixel. We calculate this prior by a simple averaging across all the ground truth in the training set. We also calculate the 'inverse' of this prior to find how likely a pixel is a background pixel based on its location in the image (which is created by calculating the probability a pixel in the training set is a background pixel). At runtime, for every patch classified, we look at a window of the spatial prior of its predicted class in the same location as the classified patch, and also weight this window by the same Gaussian of the patch's size, and calculate the likelihood that this classification is correct. This likelihood is found by taking the sum of pixels of the Gaussian-weighted spatial prior window and dividing it by a pre-calculated 'ideal' window (which is the result of this sum if all elements in the spatial prior belong to the positive class), which yields a probability (based on all previous segmentations) that the class predicted could exist in the location of the patch in question. This value is directly used to weight the Gaussian that will be added to the final segmentation.

The above text describes what we refer to as our basic method. It results in a weight map with positive weights indicating predicted road regions, which can be thresholded to produce a binary segmentation. Edge cues are not taken into consideration in this method. This method is visualized in the top branch of the algorithm in Figure 3.1.

4.1.2 Improved Method

We improved our method by leveraging edge cues found in the image. Preliminary testing showed that our edge detector, despite its merits, still falters around edges. We generate an illumination invariant, or "intrinsic", image as previously proposed [21]. Generating the invariant image requires the invariant angle, which can be found by calibrating the camera, which is something we cannot perform as we are using data from an online source. Instead, as discussed in Chapter 3, we used MATLAB to implement the intrinsic image method [20] to find the entropy of a projection at a specific angle, with this method, we can go over the range of all angles and find the one with the least entropy, which gives a very good approximation to the actual invariant angle. We found the entropy for every image in the KITTI training set at every angle, knowing that the invariant angle is a fixed parameter intrinsic to the image sensors. We then summed the entropies at every angle to

generate the plot shown in Figure 3.5, and deduced that the invariant angle is 160°. We were then able to generate the intrinsic image at this angle for every image in the KITTI dataset, which is our shadow-free projection of the original images.

We use the toolbox provided by [17] to generate a UCM, which is hierarchical map. The UCM is generated for both the natural image, and the intrinsic image. Early testing showed merits and drawbacks to both images. The edges of the natural image, while susceptible to shadows, generate strong edges at actual object boundaries in the image. Whereas the edges of the intrinsic image do not get stuck at shadows, but they miss key boundaries in the natural image like those between the road and the side-walk, for example. This is mainly due to the UCM using colour as a cue, which gets skewed in the invariant image, thereby making it lose some of its accuracy around similarly coloured objects. We want to take advantage of both these images and hence average the edge maps, which preserves natural boundaries, and reduces the edge strength of shadows. We ran our experiments on both the natural image UCM, and the UCM obtained by averaging both natural image UCM and Intrinsic image UCM.

Using our own MATLAB implementation, we generate a tree based on the hierarchy of weights of the UCM. The leaf nodes are taken to be the image pixels, but the lowest possible segmentation is at the lowest UCM threshold, which generates superpixels. Higher thresholds generate larger regions bounded by higher UCM weights. Every cut is given the class of the dominant class of the pixels it includes, however, it is also given an entropy measure based on those pixels as well. A 'purity' value is assigned to each cut by considering edge weight and entropy, as in equation 4.1.

$$Purity = (UCM\_Threshold)^{0.5} * (Entropy)^{1.5} \tag{4.1}$$

The powers to which the UCM threshold and Entropy are raised in equation 4.1 were set so the purity is slightly more biased to less entropic cuts. Note that entropy is calculated as in 4.2 for every proposed region by the UCM, using the classes found by the original method's segmentation.

$$Entropy = p(road) * ln(p(road)) + p(background) * ln(p(background)) \tag{4.2}$$

The tree is traversed upwards to generate the purity value, then downwards to find the optimal cut. An optimal cut must have the highest purity value compared to all the cuts it contains, however, any non-optimal cut follows the class of the most optimal cut including and above it. This method allows larger, more uniform cuts to be produced,

Table 4.1: Results on perspective images of the KITTI Dataset using cross validation. Our basic method builds upon the texture classifier in [11], and our improved method utilizes a UCM [5] and purity tree [18].

Method	F-measure	Precision	Recall	FPR
SRF [81]	85.96%	83.92%	90.43%	4.13%
Our Basic Method without Spatial Prior	90.20%	87.51%	**93.01**%	2.59%
Our Basic Method with Spatial Prior	**90.34**%	88.45%	92.31%	2.40%
Our Improved Method without Shadow Removal	90.10%	**89.70**%	90.49%	**2.12%**
Our Improved Method with Shadow Removal	89.33%	87.27%	91.49%	2.61%

Table 4.2: Results on perspective images of a Google Street View subset using cross validation. Our basic method builds upon the texture classifier in [11].

Method	F-measure	Precision	Recall	FPR
Our Basic Method with Spatial Prior	93.12%	93.48%	92.77%	2.92%

while still allowing small and entropic cuts (like cars, and other obstacles in the road) to be represented and follow their own class.

4.1.3 Results

Our results are shown in Tables 4.1, 4.2, and 4.3. Sample segmentations in perspective view can be seen in Figures 4.1 and 4.2 for the KITTI Dataset, and Figure 4.3 for the Google Street View Dataset.

The metrics used for evaluation are precision, recall, false-positive rate (FPR), and F-measure. Precision reflects how much of the proposed road segmentation is truly road, recall measures how much of the overall road in the image was detected, the false-positive rate reflects how much of the background class was improperly classified, and the F-

Figure 4.1: Sample segmentation results of our basic method (with the spatial prior) on testing images from the KITTI Dataset. The predicted road segmentation is overlaid in red. This method exhibits invariance to shadows and other road artefacts, however it does not precisely adhere to road boundaries.

Figure 4.2: Sample segmentation results of our improved method on testing images from the KITTI Dataset. The predicted segmentation is also overlaid in red. This method adheres to natural boundaries more than the basic method, but can be fooled by shadows and other edges.

Figure 4.3: Sample segmentation results of our basic method (with the spatial prior) on testing images from the Google Street View Subset. The predicted segmentation is overlaid in red. Performance is similar to performance on the KITTI Dataset, where shadow resilience and resilience to road artefacts is exhibited.

Table 4.3: Results on BEV images tested on the KITTI servers. Our basic method builds upon the texture classifier in [11], and our improved method utilizes a UCM [5] and purity tree [18].

Method	**F-measure**	**Precision**	**Recall**	**FPR**
DDN [55]	**93.25%**	**94.79%**	91.77%	**2.70%**
HIM [57]	90.01%	91.06%	88.98%	4.71%
Fused CRF [80]	87.95%	83.07%	**93.55%**	10.62%
SRF [81]	76.43%	75.53%	77.35%	11.42%
Our Basic Method without Spatial Prior	81.15%	89.04%	74.68 %	5.37%
Our Basic Method with Spatial Prior	78.78%	91.36%	69.35%	3.79%
Our Improved Method without Intrinsic Images	82.69%	89.56%	76.80%	4.08%
Our Improved Method with Intrinsic Images	83.48%	93.60%	75.34%	5.66%

measure is the harmonic mean that measures a trade-off between precision and accuracy. The metrics are calculated as Precision $= \frac{\text{TP}}{\text{TP+FP}}$, Recall $= \frac{\text{TP}}{\text{TP+FN}}$, FPR $= \frac{\text{FP}}{\text{FP+TN}}$, $\text{F} - \text{measure} = \frac{\text{Precision} \times \text{Recall}}{(1-\alpha) \times \text{Precision} + \alpha \times \text{Recall}}$, where $\alpha = 0.5$.

The results in Table 4.1 are obtained by cross validation on the supplied training set, where we used 200 images for training, 60 for validation and 29 for testing. The image classes (training, validation, and testing) are chosen randomly, and the training and testing processes were repeated five times to reduce noise. The reported numbers represent the performance on the testing images, averaged over the five runs. We present the results of our algorithm with and without the spatial prior. The measurements in this table are found on the original perspective images with no projection, and the numbers are based on comparing the generated segmentation to the ground truth at the pixel level. We cite our results using four different approaches: The original approach, with and without the spatial prior, and the improved method, with and without the use of intrinsic images, applied on the image segmented using the original method with the spatial prior. We also show the results found in [81], which was the only reference that cited their results in perspective

(a) (b)

Figure 4.4: (a) The basic method's segmentation in perspective view, where green represents true positives, red represents false negatives and blue represents false positives. (b) The basic method's segmentation in BEV space, where the false negatives that were a small portion of the perspective image pixels are now a significant portion of the segmentation heavily impacting recall rate (and F1-measure).

space[2].

The results in Table 4.3 are obtained for the supplied testing images. Testing is done on the KITTI evaluation servers by uploading binary masks of the proposed segmentations in projected Bird's Eye View (BEV) space. This space has the entire visible road in an image rectified using camera parameters to look as if seen from above. The KITTI servers evaluate the segments in BEV space, and return the results. We also present the results of our basic method with and without the spatial prior, and our improved method, with and without intrinsic images, applied on the basic method with the spatial prior. We show the results averaged over all the supplied KITTI dataset image classes (Unmarked Road, Marked Road ... etc.). Since KITTI allows for anonymous submissions we only compare our results to those of the top performing algorithms that include references to the methods used.

We did not test other variants of our algorithm, with tuned parameters, due to the upload limitations imposed by the KITTI evaluation server. We opted to only test the methods that performed the best using cross validation.

4.1.4 Observations and Disussions

Basic Method

Our basic algorithm performs reasonably well, and manages to identify a significant portion of the road; this is especially true of our measurements in perspective space (i.e., without projection into Bird's Eye View). While both precision and recall are important, our focus is more on precision, as it is the more important factor when it comes to our eventual aim of road quality assessment. We, therefore, do not mind the reduction in F-number that came about after introducing the spatial prior, as it also brought an increase in precision. Nevertheless, we still want our algorithm to have a high recall rate, which it does in the more applicable perspective space. The most challenging points for our basic algorithm to detect are road boundaries. This is likely a result of the larger 181x181 and 91x91 windows seeing a mixture of road and background class at those boundaries and classifying the patch as background. This is the main reason the weight of the Gaussian representing the 181x181 patch was reduced. Furthermore, since 45x45 pixels were our finest sampling on a relatively low-resolution image, it added to the problem of edges being misclassified. We believe this problem would not persist on a dataset with higher-resolution images.

Our basic method does not perform extraordinarily well compared to other methods when tested in BEV space. While we do attain a high precision rate with the prior (one of the highest), and a low false positive rate which are desirable for our application, our major drawback comes from a low recall rate in BEV space that in turn decreases F-value. The conversion to, and evaluation in, BEV space is a major factor in the apparent decrease in performance of the algorithm, since in this space smaller, and farther away portions of the road are weighted heavily, which can be seen in Figure 4.4. These road portions, while possibly significant to some applications such as vehicle navigation, only constitute a small amount of the road seen in the un-transformed, perspective, image, and are expendable in our larger context of road quality assessment. The reason we use the BEV measurement is because it is the only space where we have results for several other algorithms available, as the authors of the papers describing the top performing algorithms do not provide code for testing their methods, or evaluation results in perspective space.

In BEV space, for which we have data for the other algorithms, our basic method ranks on average, tenth overall between referenced submissions to the KITTI evaluation server. Note that while the three methods whose results are shown in Table 4.3 (DDN, HIM and FusedCRF) are the three top performing identified algorithms, they, on average, rank third, eleventh, and thirteenth respectively when anonymous submissions are counted.

[2]Note that we averaged these results for the different road types (Unmarked, Marked, Multiple Lanes).

Unfortunately none of the top-performing algorithms on the KITTI website give access to their code, which prevented us from comparing our algorithm to theirs in perspective space. However, we were able to compare our method to the method used in [81], since the authors do cite their results in perspective space. We found that our method outperforms [81] in both perspective and BEV space.

The results on the Google Street View subset were impressive as well. We were able to achieve our target and segment the road successfully in a street view image. We did not have the camera parameters required to project into BEV for evaluation; however, the results in perspective space were indicative of the success of this method, and this can be seen visually when looking at the suggested segmentation in Figure 4.3. The method was resilient to shadows and other road artefacts, especially cracks and defects (of which there were many). It also did not include objects like cars or side-walks into the segmentation, which is crucial as these can be false positives in the crack detection step. Overall, we found that the basic method was sufficient input to the crack detection algorithm and decided not to improve segmentation on this dataset.

Improved Method

Our improved algorithm performs slightly better than our basic method in BEV space (Table 4.3). It does its job of improving overall performance and recognizing natural boundaries and sticking to them. The improvement is more apparent in BEV space where our algorithm performance becomes more competitive on the KITTI dataset. This is especially true of when we added intrinsic images to remove shadows, which also improved results. The improvement is seen in the recall rate, which drives the F-measure up. The improved method with intrinsic images also attains one of the highest precisions on the dataset (93.6%), which is key as our aim is to correctly capture road more than it is to capture all the road. Figure 4.6 is generated with the KITTI evaluation and reflects the F-measure performance of the algorithm as a function of the actual distance of the road segmented; it is extremely useful in illustrating where our method fails. We notice that for a distance less than 30 meters away from the camera, an F1-measure of above 90% is attained (which signals good performance). It is only at farther away portions of the road where the method's performance takes a hit, and these are fewer pixels that translate to a larger area in BEV, which impacts performance negatively. On the other hand, the improvement in perspective space was marginal, and even caused an increase in the False Positive Rate, which is undesirable in our application.

We found that shadows are indeed the bane of computer vision methods, and we learned this the hard way with the UCM failures around shadow edges. While the Intrinsic Image

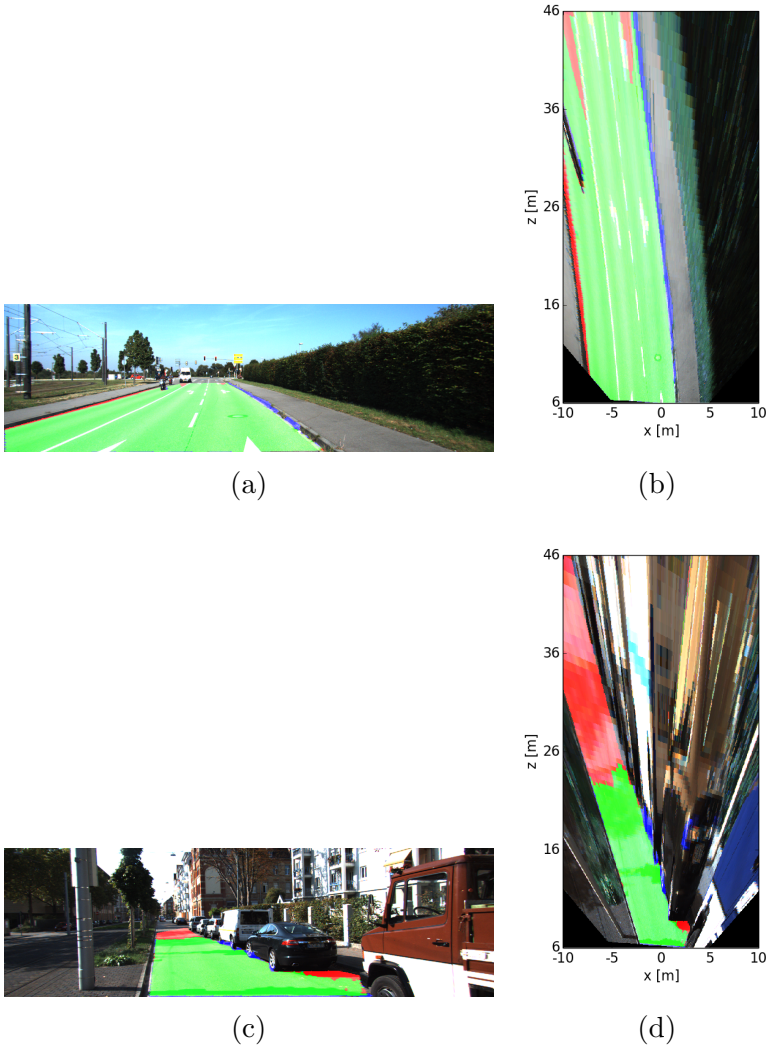


Figure 4.5: (a) The improved method's segmentation in perspective view, where green represents true positives, red represents false negatives and blue represents false positives. (b) The improved method's segmentation in BEV space, where the false negatives that impacted the basic method's performance significantly are now reduced. (c) Another improved method segmentation in perspective view, where the method falters due to shadows. (d) The very strong impact shadows can have on the improved method when projected in BEV space.

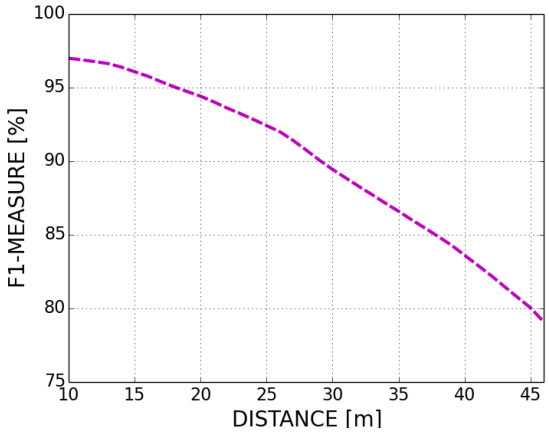


Figure 4.6: A plot of the F1-measurement across all test images as a function of distance from camera. This is especially useful to illustrate why our performance deteriorates in BEV, as our algorithm does not capture distant road pixels well.

transformation did alleviate the error generated by shadows slightly, it did not remove it completely and shadows remained the main reason the improved method did not achieve the state of the art on the KITTI dataset. We found that shadows need to be specifically targeted for removal if we hope to improve the segmentation method further. In the case of the basic method, shadows were learned as part of the road, and given their numerous appearances in the training set, the basic method's SVM learned not to mis-classify them. This was compounded with the effects of the voting scheme that ensured the neighbourhood around a pixel had a say in its label (at multiple scales as well). The purity tree approach, relied mostly on the hierarchical maps it received as an input, and these faltered heavily around shadows, often producing weak responses between road and background boundaries inside shaded regions and strong responses between road regions in and out of shadows, and opting to maximize the purity by separating at the strong edge. Ours is not the only method to fall prey to shadows, the method of [57] cited similar problems on the KITTI Dataset, while numerous other vision methods in different research spaces also list shadows as a main issue. Learning methods, like [55], sidestep this problem by learning what shaded, and partially shaded road is, and classifying it correctly, much like our basic method.

We also found transfer learning to be a problem with our methods. One of the main problems we encountered was the inability to use the same trained model on another

dataset. This is a problem frequently encountered in machine learning, however we thought that since our features were engineered, we would be able to overcome this. We deduced that camera quality is detriment to our method, a different camera gives the road a different texture and fools our method. While we can still somewhat surmise our trained categories, the top performance cannot be achieved unless we train on this new data.

As mentioned earlier, we do aim for the best possible classifier, however, we favour more precise methods over ones that increase recall. Since we are already above the 90% range of recall (in perspective space), the small improvement brought about by the improved method is not crucial, especially since it mostly adds the farther away and boundary pixels which do not contain as much usable information when finding cracks. This is why we do not use the improved method in our crack detection experiments, and rely only on the basic segmentation which is more than sufficient for proper crack detection.

4.2 Crack Detection

We only tested our crack detection algorithm on the subset of images from Google Street View, obtained by querying the Google Street View API in the Hamilton region. We annotated the 250 images using LibLabel [23], and used the same 50 images as before for validation, and the same 50 images for testing. We initially manually annotated the road in the images into regions of 'good', 'medium', and 'poor' road. 'Good' road was defect free, whereas 'medium' road contains small and medium cracks and defects, and 'poor' road contained potholes and larger cracks and defects. The annotation marked lobular regions around the defects, but we should note that pinpointing the exact defects and their severity is not our forte. However, we noticed only a few samples contained 'poor' road, which was insufficient for training an SVM, and decided to merge 'medium' and 'poor' road labels into one 'distressed' road label. We could not find any annotated street view road defect database, and therefore only tested on our manually annotated one.

A rigorous road segmentation step allows us to have a more forgiving crack detection method. We start by separating the patches that belong to the road class, based on the segmentation. We only use 91x91 patches, and we simply reuse the Fisher Vectors calculated on these patches, since our skip length was equal to the one used in the segmentation. We then classify the patches using an SVM trained on 'good' and 'distressed' road patches, and classify each patch accordingly. We re-apply the gaussian voting scheme to generate a segmentation that distinguishes between road qualities. This step is extremely fast, as it boils down to simply classifying the patches, since the features were already computed.

Table 4.4: Results on crack detection for images of a Google Street View subset.

Method	F-measure	Precision	Recall	FPR
Basic Method with Spatial Prior	92.84%	86.64%	100%	0.02%

We zero in on specific crack locations by utilizing the UCM. We find strong UCM responses inside the regions classified as 'poor' and can deduce specifically where the crack, pothole, or other defect is located. We store all responses, keeping in mind that a larger UCM value corresponds to a more severe crack. We separated the crack severities according to our own knowledge acquired from [50, 38, 35] among other sources on road quality. While not ideal, we were able to make more educated assumptions as to what qualifies as 'mild', 'medium' and 'severe' defects.

We compute quantitative results on crack detection by comparing crack detection in predicted 'distressed' road regions to the cracks detected on the 'distressed' road regions specified by the ground truth. We weight all three different crack severities equally in this evaluation, and this serves as a good method to evaluate how well crack regions are captured using the segmentation method.

4.2.1 Results

Table 4.2 shows the road segmentation results on this dataset, Table 4.4 shows the crack detection results of this algorithm. We show sample road segmentation results in Figure 4.3, and sample defect detection results in Figure 4.7.

Despite not having quantitative results to compare against, we saw that the method was able to single out cracks and poor road regions well, and perform quite favourably. The recall of 100% is due to the crack detection method being more lax, which is a reasonable path to take given the strictness of the segmentation algorithm. This is where the high precision of the segmentation algorithm comes into play, as it provided us with a reliable segmentation on which to detect cracks. We can also see that the portions of the road the segmentation algorithm missed on, in the distance and on the boundaries, were not detrimental to the crack detection algorithm. The precision rate could be improved by making the algorithm stricter, however, we did not deem that necessary after qualitatively assessing the algorithm output. Despite the low quality and coarseness of our manually labelled data, the classifier was able to truly learn what 'distressed' road looks like. Texture was the key cue in detecting poor roads, and it is the one we utilized, and it predictably generated good results. Whether the data is collected from an online database like Google

Figure 4.7: Sample crack detection results of our algorithm. The street view image (left column) is shown. The image is segmented using the initial region segmentation scheme (middle column) is annotated with a green overlay for detected 'good' road and red for detected 'distressed' road. The crack detection scheme (right column) displays severities of detected defects in green for 'mild' defects, orange for 'medium' and red for 'severe' defects.

Street View, or on a municipal level, the method is still applicable. As long as the images are of a resolution high enough to see these artefacts, the algorithm can be trained to detect them.

4.2.2 Observations and Disussions

After studying the results quantitatively and qualitatively, we can conclude that a road assessment system based off of street view data is viable. This points to the fact that we indeed can, and have, emulated a crucial portion of a surveyor's process of road assessment. We hypothesize that training a simple regression model on the percentage of 'good' and 'poor' road may be enough to predict the Pavement Condition Index (PCI), however a more precise method would identify the types of cracks and utilize the literature formulas to calculate a standardized PCI. To our dismay the lack of any sort of training data on PCI rendered us unable to to prove this. However, it is clear that the algorithm demonstrated the ability to detect cracks, and hence we would only need to acquire training data for what different cracks mean in terms of road quality to fully automate road quality assessment at an even finer level as done in [61] or [6].

The question of how well our method performs when it comes to predicting the type of crack exhibited is left unanswered. We were unable to find any ground truth that would allow us to test the proficiency of our method in this area nor were we able to label the defect types ourselves in our manually labelled ground truths, and hence could not definitively answer this question. We did find that when we used 3 levels of 'road quality' measurements (good, medium and poor) the multi-class SVM was insufficiently trained and did not output favourable results. However this would not have been accurate at all at predicting road quality, as certain defect types affect overall PCI much more than others [2]. The ideal defect detection and labelling method must label all defect types and also specify their severity to be able to generate a reliable SCR, which is combined with accelerometer data in the form of the RCI to produce a reliable PCI. The generated PCI would emulate how a single ideal surveyor would score the road if they did not miss a single defect and were in the street view collection vehicle.

Chapter 5

Conclusion

Throughout our work we were usually debating the merits of engineered versus learned features. Our method uses engineered features; the Fisher vector formulation relies on Gaussian Mixture Models computed on engineered SIFT features, and it combines the power of generative and discriminative approaches. Our spatial prior is engineered, since we know where road is likely to be in a given image, and know this will be consistent in every image. Even our improvement, which utilizes a UCM by optimizing a purity tree, was engineered as well. We leveraged a priori knowledge to achieve competitive results and solve a problem we faced; we engineered a solution.

On the flip side, the most competitive road segmentation methods used a convolutional neural network, which learns its features. The drawback of learning methods is the long training time, and the large amount of data needed to train them well, along with enormous computational power. However, both engineered and learned methods perform comparably; arguably, advanced learning methods (namely a deep CNN) have the capability to consistently outperform engineered methods when more data is present. This is because in their own way, CNN's engineer their own features automatically by learning from large datasets. This explains the recent shift of research to the realm of fully learned methods, i.e., methods trained end-to-end which take in raw data, and output desired results. Are these methods inherently better? A lot of researchers' main qualm with learning methods is that their inner workings are hidden and often complex or too difficult to understand. In the case of CNN's, we are learning a plethora of weights and filters to convolve with, what those weights really mean is less straightforward to understand. Research in [83] even visualized the weights of the layers of the image-net CNN and they noticed that remarkably the first few layers learned edge features, which are a basic feature in image processing and even in the human visual system. Deeper layers tended to learn more specialized filters,

starting with patterns (like a zigzag or circle shape), to more complex features (like a general face), and even extremely specialized features (like a cat's face) in the deepest layers. We do not really know whether performance would be the same on any other dataset, networks trained extensively theoretically should converge to the same minima, however, would a network trained on a different object recognition dataset learn the same shallow features (edges)?

Looking forward we hope to be able to train and test the entire assessment method on PCI data. Ideally we would classify the road defect individually, and assess its effect on PCI based on its size and severity, which would likely yield a precise and consistent method on all collected data. This method has the potential to surpass human surveyor's assessments as it would be unbiased and all-encompassing. We could optionally couple the data with that of an accelerometer that would aid in finding road segments in distress, and allow us to use the formulas used in road assessment today. We would need to have a database of labelled defects that we could train the SVM on, and would utilize the same method for feature extraction as we've done in this thesis. Our method's main merit is in the fact that it uses the details it needs, for crack region detection the only relevant detail is texture information.

Furthermore we also faced the issue of generalisability of our method. Rudimentary testing showed that the classifier trained on the KITTI dataset images did not generalize well to Google Street View images. While precision remained somewhat high, recall was very low. The classifier was tuned to the features of the KITTI dataset roads and had trouble classifying the detected features in Google Street View roads. The difference was expected as the two datasets differ with regards to image resolution, camera make, field of view, geographic location of road (which translates to differing pavement qualities), lighting, noise level among many other differences between the images found in the datasets. The only real similarity is they both contain images of road taken at street level, surrounded by the various background scenery that could exist around streets. Which leads to the question of what road really is and how can we detect road in any database reliably without having to train another classifier to do so. Is there one set of features that are shared by all possible roads? Should these features be engineered, or would a deep network be able to learn them with enough training examples?

Also, while our method is specific to road quality assessment, other infrastructure can be similarly mined and assessed. Side-walks and power lines are among the many facets of municipal instructive that are visible above ground. While specialized techniques would have to be developed for detecting these objects, visual assessment can be replaced with automated assessment once they are detected.

References

[1] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Susstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2274–2282, 2012.

[2] Federal Highway Administration. *Pavement Distress Identifiation Manual*. United States Department of Transportation, 4 edition, 2009.

[3] José M Alvarez, Theo Gevers, and Antonio M López. Vision-based road detection using road models. In *IEEE International Conference on Image Processing (ICIP)*, pages 2073–2076. IEEE, 2009.

[4] Mohamed Aly. Real time detection of lane markers in urban streets. In *IEEE Intelligent Vehicles Symposium*, pages 7–12. IEEE, 2008.

[5] Pablo Arbelaez, Michael Maire, Charless Fowlkes, and Jitendra Malik. Contour detection and hierarchical image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(5):898–916, 2011.

[6] Guanqun Bao. *Road distress analysis using 2D and 3D information*. PhD thesis, The University of Toledo, 2010.

[7] Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(11):1222–1239, 2001.

[8] David Abou Chacra and John Zelek. Using superpixels for road segmentation in street view images. *Vision Letters*, 1(1), 2015.

[9] David Abou Chacra and John Zelek. Road segmentation in street view images using texture information. In *13th Conference on Computer and Robot Vision (CRV) 2016*, 2016.

[10] Liang Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. *arXiv preprint arXiv:1412.7062*, 2014.

[11] Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3606–3613, 2014.

[12] Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, and Andrea Vedaldi. Deep filter banks for texture recognition, description, and segmentation. *International Journal of Computer Vision*, 118(1):65–94, 2016.

[13] CIRC. Canadian infrastructure report card. Technical report, Canadian Construction Association, Canadian Public Works Association, Canadian Society for Civil Engineering, Federation of Canadian Municipalities, 2016.

[14] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[15] Peter Corke, Rohan Paul, Winston Churchill, and Paul Newman. Dealing with shadows: Capturing intrinsic scene appearance for image-based outdoor localisation. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2085–2092. IEEE, 2013.

[16] Antonio Criminisi, Jamie Shotton, and Ender Konukoglu. Decision forests: A unified framework for classification, regression, density estimation, manifold learning and semi-supervised learning. *Foundations and Trends® in Computer Graphics and Vision*, 7(2–3):81–227, 2012.

[17] Piotr Dollár and Lawrence C Zitnick. Structured forests for fast edge detection. In *IEEE International Conference on Computer Vision*, pages 1841–1848, 2013.

[18] Clement Farabet, Camille Couprie, Laurent Najman, and Yann LeCun. Learning hierarchical features for scene labeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1915–1929, 2013.

[19] Graham D Finlayson, Mark S Drew, and Cheng Lu. Intrinsic images by entropy minimization. In *European conference on computer vision*, pages 582–595. Springer, 2004.

[20] Graham D Finlayson, Mark S Drew, and Cheng Lu. Entropy minimization for shadow removal. *International Journal of Computer Vision*, 85(1):35–57, 2009.

[21] Graham D Finlayson, Steven D Hordley, and Mark S Drew. Removing shadows from images. In *European conference on computer vision*, pages 823–836. Springer, 2002.

[22] Jannik Fritsch, Tobias Kuehnl, and Andreas Geiger. A new performance measure and evaluation benchmark for road detection algorithms. In *International Conference on Intelligent Transportation Systems (ITSC)*, 2013.

[23] Andreas Geiger, Martin Lauer, Christian Wojek, Christoph Stiller, and Raquel Urtasun. 3d traffic scene understanding from movable platforms. *Pattern Analysis and Machine Intelligence (PAMI)*, 2014.

[24] Kathleen T Hall, Carlos E Correa, Samuel H Carpenter, and Robert P Elliot. Rehabilitation strategies for highway pavements. *National Cooperative Highway Research Program, Web Document*, 35, 2001.

[25] Kotaro Hara. Scalable methods to collect and visualize sidewalk accessibility data for people with mobility impairments. In *Proceedings of the adjunct publication of the 27th annual ACM symposium on User interface software and technology*, pages 1–4. ACM, 2014.

[26] Kotaro Hara, Shiri Azenkot, Megan Campbell, Cynthia L Bennett, Vicki Le, Sean Pannella, Robert Moore, Kelly Minckler, Rochelle H Ng, and Jon E Froehlich. Improving public transit accessibility for blind riders by crowdsourcing bus stop landmark locations with google street view: An extended analysis. *ACM Transactions on Accessible Computing (TACCESS)*, 6(2):5, 2015.

[27] Kotaro Hara, Jin Sun, Robert Moore, David Jacobs, and Jon Froehlich. Tohme: Detecting curb ramps in google street view using crowdsourcing, computer vision, and machine learning. In *Proceedings of the 27th annual ACM symposium on User interface software and technology*, pages 189–204. ACM, 2014.

[28] Mohamed A Helala, Ken Q Pu, and Faisal Z Qureshi. Road boundary detection in challenging scenarios. In *IEEE Ninth International Conference on Advanced Video and Signal-Based Surveillance (AVSS)*, pages 428–433. IEEE, 2012.

[29] Aharon Bar Hillel, Ronen Lerner, Dan Levi, and Guy Raz. Recent progress in road and lane detection: a survey. *Machine Vision and Applications*, 25(3):727–745, 2014.

[30] Yong Hu, Chun-xia Zhao, and Hong-nan Wang. Automatic pavement crack detection using texture and shape descriptors. *IETE Technical Review*, 27(5):398–405, 2010.

[31] Jianping Huang, Wanyu Liu, and Xiaoming Sun. A pavement crack detection method combining 2d with 3d information based on dempster-shafer theory. *Computer-Aided Civil and Infrastructure Engineering*, 29(4):299–313, 2014.

[32] IGM. Viziroad. "http://www.igm.fr/fr/In-Situ/Essais-routiers/VIZIROAD,120", 2010.

[33] Dana E Ilea and Paul F Whelan. Image segmentation based on the integration of colour–texture descriptorsa review. *Pattern Recognition*, 44(10):2479–2501, 2011.

[34] Asphalt Institute. Asphalt pavement distress summary. http://www.asphaltinstitute.org/asphalt-pavement-distress-summary/.

[35] Pavement Interactive. Pavement distress. http://www.pavementinteractive.org/article/general-guidancepavement-distress/, April 2009.

[36] Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman. Deep features for text spotting. In *Computer Vision–ECCV 2014*, pages 512–528. Springer, 2014.

[37] Christian Koch and Ioannis Brilakis. Pothole detection in asphalt pavement images. *Advanced Engineering Informatics*, 25(3):507–515, 2011.

[38] Christian Koch, Kristina Georgieva, Varun Kasireddy, Burcu Akinci, and Paul Fieguth. A review on computer vision based defect detection and condition assessment of concrete and asphalt civil infrastructure. *Advanced Engineering Informatics*, 29(2):196–210, 2015.

[39] Hui Kong, Jean-Yves Audibert, and Jean Ponce. General road detection from a single image. *IEEE Transactions on Image Processing*, 19(8):2211–2220, 2010.

[40] Tomáš Krajník, Jan Blažíček, and Joao M Santos. Visual road following using intrinsic images. In *Mobile Robots (ECMR), 2015 European Conference on*, pages 1–6. IEEE, 2015.

[41] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[42] John Laurent, Jean François Hébert, Daniel Lefebvre, and Yves Savard. 3d laser road profiling for the automated measurement of road surface conditions and geometry. In *7th RILEM Int. Conf. on Cracking in Pavements, Springer, Netherlands., Google Scholar*, 2012.

[43] Yann LeCun and Yoshua Bengio. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

[44] Joseph Lim, C Zitnick, and Piotr Dollár. Sketch tokens: A learned mid-level representation for contour and object detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3158–3165, 2013.

[45] Huidrom Lokeshwor, Lalit K Das, and Savita Goel. Robust method for automated segmentation of frames with/without distress from road surface video clips. *Journal of Transportation Engineering*, 140(1):31–41, 2013.

[46] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

[47] David G Lowe. Object recognition from local scale-invariant features. In *The Seventh IEEE international conference on Computer Vision*, volume 2, pages 1150–1157. Ieee, 1999.

[48] AGCS Marques and Paulo Lobato Correia. Automatic road pavement crack detection using svm. *Lisbon, Portugal: Dissertation for the Master of Science Degree in Electrical and Computer Engineering at Instituto Superior Técnico*, 2012.

[49] Joel C McCall and Mohan M Trivedi. Video-based lane estimation and tracking for driver assistance: survey, system, and evaluation. *IEEE transactions on Intelligent Transportation Systems*, 7(1):20–37, 2006.

[50] Kenneth H McGhee. *Automated pavement distress collection techniques*, volume 334. Transportation Research Board, 2004.

[51] Kevin McGuinness and Noel E Oconnor. A comparative evaluation of interactive segmentation algorithms. *Pattern Recognition*, 43(2):434–444, 2010.

[52] Caio César Teodoro Mendes, Vincent Frémont, and Denis Fernando Wolf. Exploiting fully convolutional neural networks for fast road detection. In *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016.

[53] Christoph Mertz. Continuous road damage detection using regular service vehicles. In *Proceedings of the ITS World Congress*, 2011.

[54] Christoph Mertz, Srivatsan Varadharajan, Sobhagya Jose, Karan Sharma, Lars Wander, and Jina Wang. City-wide road distress monitoring with smartphones. In *Proceedings of ITS World Congress*, September 2014.

[55] Rahul Mohan. Deep deconvolutional networks for scene parsing, 2014.

[56] Yair Movshovitz-Attias, Qian Yu, Martin C Stumpe, Vinay Shet, Sacha Arnoud, and Liron Yatziv. Ontological supervision for fine grained classification of street view storefronts. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1693–1702. IEEE, 2015.

[57] Daniel Munoz, J. Andrew Bagnell, and Martial Hebert. Stacked hierarchical labeling. In *European Conference on Computer Vision (ECCV)*, 2010.

[58] Sebastian Nowozin and Christoph H Lampert. Structured learning and prediction in computer vision. *Foundations and Trends® in Computer Graphics and Vision*, 6(3–4):185–365, 2011.

[59] American Society of Civil Engineers. Report card for america's infrastructure. "http://www.infrastructurereportcard.org/a/documents/Roads.pdf", 2013.

[60] Henrique Oliveira and Paulo Lobato Correia. Automatic road crack detection and characterization. *IEEE Transactions on Intelligent Transportation Systems*, 14(1):155–168, 2013.

[61] Henrique Jos Monteiro Oliveira. *Crack Detection and Characterization in Flexible Road Pavements using Digital Image Processing*. PhD thesis, Universidade de Lisboa - Instituto Superior Tcnico, 2013.

[62] Pavemetrics. Our products. http://www.pavemetrics.com/applications/road-inspection/laser-crack-measurement-system/.

[63] Florent Perronnin and Christopher Dance. Fisher kernels on visual vocabularies for image categorization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8. IEEE, 2007.

[64] Florent Perronnin, Jorge Sánchez, and Thomas Mensink. Improving the fisher kernel for large-scale image classification. In *Computer Vision–ECCV 2010*, pages 143–156. Springer, 2010.

[65] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. Grabcut: Interactive foreground extraction using iterated graph cuts. In *ACM transactions on graphics (TOG)*, volume 23, pages 309–314. ACM, 2004.

[66] M Salman, S Mathavan, K Kamal, and M Rahman. Pavement crack detection using the gabor filter. In *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*, pages 2039–2044. IEEE, 2013.

[67] Jan Salmen, Sebastian Houben, and Marc Schlipsing. Google street view images support the development of vision-based driver assistance systems. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 891–895. IEEE, 2012.

[68] Marcos Santos, Marcelo Linder, Leizer Schnitman, Urbano Nunes, and Lara Oliveira. Learning to segment roads for traffic analysis in urban images. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 527–532. IEEE, 2013.

[69] Colette Caruso Scott Butler. Pavement condition index 101, December 2009.

[70] Aparna Taneja, Luca Ballan, and Marc Pollefeys. Never get lost again: Vision based navigation using street view images. In *Computer Vision–ACCV 2014*, pages 99–114. Springer, 2014.

[71] Tsung-Hung Tsai, Wen-Huang Cheng, Chuang-Wen You, Min-Chun Hu, Arvin Wen Tsui, and Heng-Yu Chi. Learning and recognition of on-premise signs from weakly labeled street view images. *IEEE Transactions on Image Processing*, 23(3):1047–1059, 2014.

[72] Yi-Chang Tsai, Chenglong Jiang, and Yuchun Huang. Multiscale crack fundamental element model for real-world pavement crack classification. *Journal of Computing in Civil Engineering*, 28(4):04014012, 2012.

[73] Yi-Chang Tsai, Vivek Kaul, and Russell M Mersereau. Critical assessment of pavement distress segmentation methods. *Journal of transportation engineering*, 136(1):11–19, 2009.

[74] Michael Van den Bergh, Xavier Boix, Gemma Roig, and Luc Van Gool. Seeds: Superpixels extracted via energy-driven sampling. *International Journal of Computer Vision*, 111(3):298–314, 2015.

[75] Srivatsan Varadharajan, Sobhagya Jose, Karan Sharma, Lars Wander, and Christoph Mertz. Vision for road inspection. In *IEEE Winter Conference on Applications of Computer Vision*, pages 115–122. IEEE, 2014.

[76] A. Vedaldi and B. Fulkerson. Vlfeat: An open and portable library of computer vision algorithms. http://www.vlfeat.org/, 2008.

[77] Kelvin CP Wang and Omar Smadi. Automated imaging technologies for pavement distress surveys. *Transportation Research E-Circular*, 2011.

[78] Qi Wang, Jianwu Fang, and Yuan Yuan. Adaptive road detection via context-aware label transfer. *Neurocomputing*, 158:174–183, 2015.

[79] Jianxiong Xiao and Long Quan. Multiple view semantic segmentation for street view images. In *IEEE 12th International Conference on Computer Vision*, pages 686–693. IEEE, 2009.

[80] Liang Xiao, Bin Dai, Daxue Liu, Tingbo Hu, and Tao Wu. Crf based road detection with multi-sensor fusion. In *Intelligent Vehicles Symposium (IV)*, 2015.

[81] Liang Xiao, Bin Dai, Daxue Liu, Dawei Zhao, and Tao Wu. Monocular road detection using structured random forest. *International Journal of Advanced Robot Systems*, 13:101, 2016.

[82] Amir Roshan Zamir and Mubarak Shah. Image geo-localization based on multiple-nearest neighbor feature matching using generalized graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(8):1546–1558, 2014.

[83] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision*, pages 818–833. Springer, 2014.

[84] Lawrence C Zitnick and Piotr Dollár. Edge boxes: Locating object proposals from edges. In *Computer Vision–ECCV 2014*, pages 391–405. Springer, 2014.

APPENDICES

Appendix A

Pavemetrics Datasheets

Pavemetrics

www.pavemetrics.com

Laser Road Imaging System

PAVEMETRICS's laser road imaging system (LRIS) is composed of two high resolution linescan cameras and lasers that are configured to image 4m transverse road sections with 1 mm resolution at speeds that can reach 100 km/h. This patented imaging system was designed to increase the contrast and visibility of both small longitudinal and lateral road cracks.

The LRIS optical configuration increases the visibility of even the smallest cracks by using the incident illumination angle of the laser to cause the cracks to project shadows.

Using high power laser line projectors and advanced optics, the LRIS system is very robust to variations in outside lighting conditions and shadows cast by roadside objects, viaducts and the inspection vehicle itself.

KEY FEATURES

- 1 mm imaging at 100 km/h
- Day or night operation
- Crack image contrast enhancement
- Low power consumption
- Compact system
- Water resistant (IP65) housing

INO

WWW.INO.CA

Pavemetrics

www.pavemetrics.com

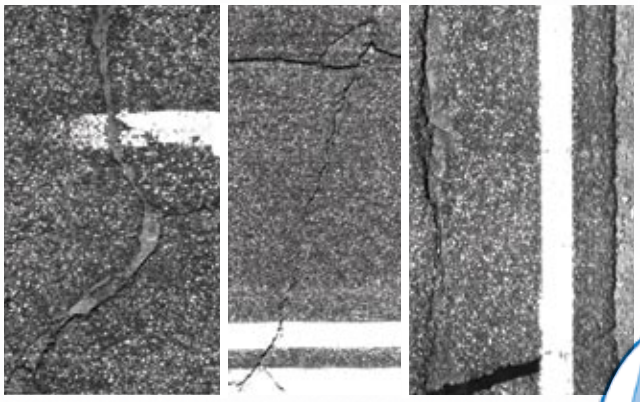
Laser Road Imaging System

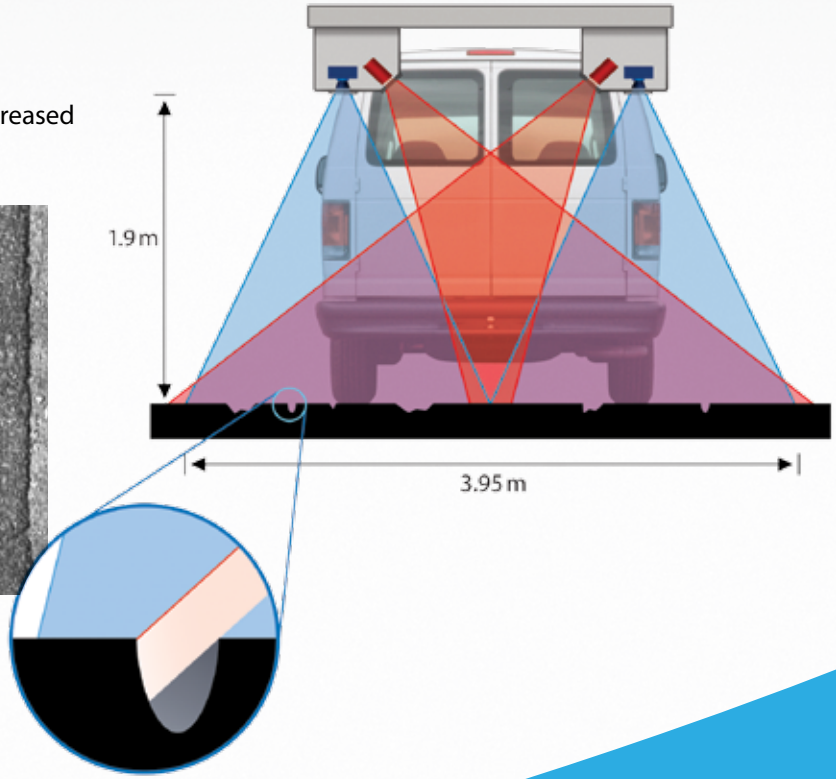
SPECIFICATIONS

Image size:	4096 pixels/line
Line rate:	28000 lines/s
Image width:	4 m (3950 mm nominal)
Laser class:	3B
Power:	250W at 120/240 VAC
Sensor size (approx.):	300 mm(H) x 375 mm(L) x 200 mm(D)
Sensor weight (approx.):	20 kg

SYSTEM CONFIGURATION

Incident angle of the illumination system allows increased visibility of small cracks by projecting shadows.





All specifications subject to change without notice

INO

WWW.INO.CA

Pavemetrics

www.pavemetrics.com

Laser Crack Measurement System

The laser crack measurement system (LCMS) uses laser line projectors, high speed cameras and advanced optics to acquire high resolution 3D profiles of the road. This unique 3D vision technology allows for automatic pavement condition assessment of asphalt, porous asphalt, chipseal and concrete surfaces. The LCMS acquires both 3D and 2D image data of the road surface with 1 mm resolution over a 4 m lane width at survey speeds up to 100 km/h.

LCMS data is acquired and compressed in real time in the survey vehicle so as to minimize storage needs (<1Gb per km). The collected data can then be analyzed using Pavemetric's data processing toolbox (DLL library of C/C++ functions). This library has functions to detect and analyze cracks, lane markings, potholes, ravelling, and macro-texture. Rutting is also measured and characterized using more than 4 000 points and rut depth and type (short, multiple, long radius) is evaluated. Concrete road surfaces can be scanned to evaluate joints, tinning and faulting between the concrete slabs. IMUs can be added to the sensors in order to measure longitudinal profiles, IRI, slope and crossfall.

KEY FEATURES

- Automatic crack detection and severity
- 4 160 point rutting (rut depth, rut type)
- Macro-texture measurements over 100 % of the lane width.
- 3D and 2D data to characterise: cracks, pot holes, ravelling, sealed cracks, joints in concrete, tinning, etc.
- Day and night operation
- Low power consumption
- High resolution (1mm) downward images
- IRI and longitudinal profile
- Slope and crossfall

***Vision Systems** for the Automated Inspection of Transportation Infrastructures*

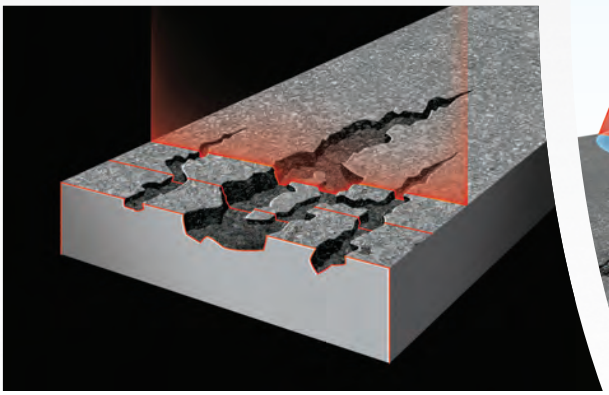
Pavemetrics

www.pavemetrics.com

Laser Crack Measurement System

SYSTEM SPECIFICATIONS

- Number of laser profiles: 2
- Sampling rate: 5 600 profiles/s or 11 200 profiles/s
- Vehicle speed: 0 to 100 km/h
- Profile spacing: 1 to 5 mm (adjustable)
- Transversal field of view: 4 m
- Transversal accuracy: 1 mm
- Transversal resolution: 4 096 points/profile
- Depth range of operation: 250 mm (adjustable)
- Depth accuracy: 0.5 mm
- Laser profiler dimensions: 428 mm (h) x 265 mm (l) x 139 mm (w)
- Weight: 10 kg
- Power consumption (max): 150W at 120/240 VAC



All specifications subject to change without notice

***Vision Systems** for the Automated Inspection of Transportation Infrastructures*

Pavemetrics

www.pavemetrics.com

Laser Rut Measurement System

Pavemetrics's laser rut measurement system (LRMS) is a transverse profiling device that detects and characterizes pavement rutting. The LRMS can acquire full 4-meter width profiles of a highway lane at normal traffic speeds, with 2 options of maximum sampling rate: 30 or 150 Hz. The system uses two 3D laser profilers that digitize transverse sections of the pavement with 1280 points. Custom optics and high-power pulsed laser line projectors allow the system to operate in full daylight or in night-time conditions. Road transverse profile data can be collected and processed in real time on board the vehicle.
Rut extraction algorithms have been developed to automatically measure rut depth and width.

The system is delivered with a complete DLL library of C/C++ functions allowing the user to easily configure the sensors, acquire transverse profiles, extract road rut data, classify rut type (short, multiple, long radius) and validate the laser profiler calibration. Over the past years, this system has been used on a continuous basis by dozens of governments and private agencies. Hundreds of thousands of road kilometers have been surveyed using this technology.

KEY FEATURES

- 1280 point 3D transverse profiles
- Daylight or nighttime operation
- Short integration times for minimal image blur at maximum inspection speeds
- A library of C/C++ functions for easy use and integration
- Proven performance
- Inspection speeds up to 100 km/h
- Rut depth and type (short, multiple and long radius) is evaluated.

INO

WWW.INO.CA

Pavemetrics

www.pavemetrics.com

Laser Rut Measurement System

BENEFITS

- Immediate and precise detection and characterization of rutting conditions
- Optimization of road maintenance funds
- Improvement of safety due to better road pavement maintenance

SYSTEM SPECIFICATIONS

- Number of laser profiles: 2
- Number of 3D points per profile (max): 1280 points
- Sampling rate: 30 or 150 profiles/s
- Vehicle speed: 0 to 100 km/h
- Profile spacing: adjustable
- Transversal field-of-view (nominal): 4 m
- Transversal resolution ±2 mm
- Depth range of operation: 500 mm (30 Hz) or 450 mm (150 Hz)
- Depth accuracy (nominal) ±1 mm
- Laser profiler dimensions (approx.): 108 mm(W) x 692 mm(H) x 220 mm(D)
- Laser profiler weight (approx.): 12 kg
- Power consumption (max): 150 W at 120/240 VAC

All specifications subject to change without notice

INO

WWW.INO.CA

Pavemetrics Products Comparison

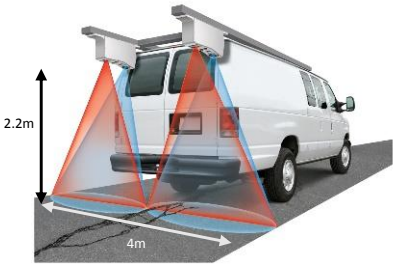
Products

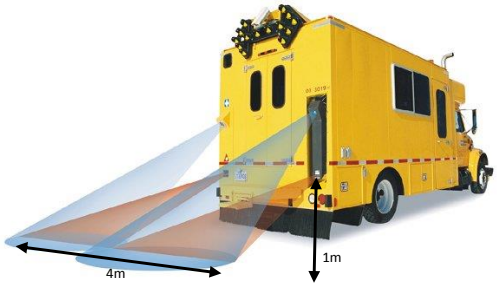
Laser Crack Measurement System (LCMS)

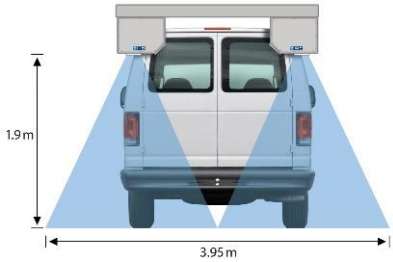
Laser Rut Measurement System (LRMS)

Laser Road Imaging System (LRIS)

Installation







What Does It Do?

	LCMS	LRMS	LRIS
2D Imaging	✓		✓
3D Profiling	✓	✓	
3D Imaging	✓		
Cracking (automatic detection)	✓		
Rutting (automatic detection)	✓	✓	
Ravelling (automatic detection)	✓		
Macro-Texture (evalutation)	✓		
Longitudinal Profile and IRI Measurement	✓		
Slope, Cross Fall and Super Elevation	✓		

Laser Crack Measurement System (LCMS)

Laser Rut Measurement System (LRMS)

Laser Road Imaging System (LRIS)

What Is In the Box?

- Two 3D laser profiling sensors
- Rackmount controller
- Two PCIe camera link frame grabber boards
- All necessary cables
- User manuals

- Two 3D laser profiling sensors
- Rackmount controller
- One PCIe frame grabber board
- All necessary cables
- User manuals

- Two high power laser and two linescan (2048 pixel) imaging units
- Rackmount controller
- 1 beam for mounting the laser units
- One PCIe frame grabber board
- All necessary cables
- User manuals

What Type of Software Comes With It?

LCMS C/C++ acquisition and processing DLL library for custom user applications

AND

LCMS acquisition and processing applications with graphical user interface (GUI)

LRMS C/C++ acquisition and processing DLL library for custom user applications

LRIS C/C++ acquisition DLL library for custom user applications

Key Specifications

- Day and night operation
- 100 km/h operating speed
- Up to 11,200 Hz scanning frequency
- 0.5 mm vertical accuracy
- 1 mm transverse resolution (4096 points)
- 2.5 mm longitudinal scanning interval (configurable)

- Day and night operation
- 100 km/h operating speed
- 30 or 250 Hz scanning frequency
- 1 mm vertical accuracy
- 1,280 point transverse profiles

- Day and night operation
- 100 km/h operating speed
- 32,000 Hz scanning frequency
- 1 mm 2D imaging
- 1 mm transverse resolution (optional 0.5 mm resolution)
- 1 mm longitudinal scanning interval

Outputs

- 3D transversal profiles
- Pavement Images
- Distresses Identification

- 3D Transverse profiles
- Rutting values

- Downward Pavement Images