Vision for Road Inspection

Srivatsan Varadharajan, Sobhagya Jose, Karan Sharma, Lars Wander and Christoph Mertz
{svaradha, sjose, karans, lwander, cmertz}@andrew.cmu.edu
Robotics Institute, School of Computer Science
Carnegie Mellon University

Abstract

Road surface inspection in cities is for the most part, a task performed manually. Being a subjective and labor intensive process, it is an ideal candidate for automation. We propose a solution based on computer vision and data-driven methods to detect distress on the road surface. Our method works on images collected from a camera mounted on the windshield of a vehicle. We use an automatic procedure to select images suitable for inspection based on lighting and weather conditions. From the selected data we segment the ground plane and use texture, color and location information to detect the presence of pavement distress. We describe an over-segmentation algorithm that identifies coherent image regions not just in terms of color, but also texture. We also discuss the problem of learning from unreliable human-annotations and propose using a weakly supervised learning algorithm (Multiple Instance Learning) to train a classifier. We present results from experiments comparing the performance of this approach against multiple individual human labelers, with the ground-truth labels obtained from an ensemble of other human labelers. Finally, we show results of pavement distress scores computed using our method over a subset of a citywide road network.

1. Introduction

State and local maintenance departments are tasked with keeping roads in good repair. One part of this task is to monitor the degradation of road condition, which manifests itself with the presence of cracks, potholes, and other distress. Currently, this is done by: (i) inspectors who visually judge the road condition, (ii) specialized vehicles which measure the distress with cameras or laser devices, (iii) citizens who call in their observations. The first method is tedious and often inconsistent if several inspectors do the inspections. The second method does not have these problems, but it is generally expensive. Usually one stretch of road is only traversed by the specialized vehicle once every two years. This method is effective for interstates and highways, but it is not practical for the inner roads in a city. As for the third method, reports by citizens are generally only about severe problems, like large potholes on main roads. It is therefore desirable to have a low-cost system that can monitor the roads on a continuous basis with minimal human intervention.

We propose a road inspection system that works on images or videos collected by commodity devices such as smartphones. In this work, we have focused on detecting cracks in particular because they have strong texture cues and are the most prevalent type of road damage. No dedicated drivers are needed for the data collection if the cameras are mounted on service vehicles like garbage trucks and police cars that already drive through all the neighborhoods for other purposes. The collected data is then analyzed to score the distress level of the roads.

Examining images for road damage is a challenging computer vision task for many reasons. Firstly, since the camera is positioned such that it captures images from the passenger's perspective, most of the pixels in the image are not from the ground (figure 1). Secondly, damaged road sections neither differ much in terms of color from their background nor do they have well defined borders when compared to objects like vehicles and pedestrians on the road and therefore cannot be easily segmented out. Finally, there is significant variability in annotations of road damage produced by humans because of the highly subjective nature of the task. In addition, annotating the boundaries of cracks precisely takes a lot of diligence on the part of the human labeler, which cannot be always counted on. Therefore one of the key ideas in our approach is using supervised learning methods that can deal with poorly or weakly labeled data. In the following sections we will provide an overview of some background work on this application and then describe the data collection system, followed by a description of the algorithms used and finally we present results.

2. Related Work

Camera based systems for road inspection are not a novel idea by themselves. There are specialized vehicles that use cameras [1] or lasers [2] to get accurate distress data [10] for the road. However, the cost of these vehicles can be several hundred-thousands of dollars and they use dedicated drivers to cover all the roads that need to be inspected. This process is accurate, but is too expensive for many maintenance agencies and is therefore performed only once every two years or so. It is also more useful on highways and interstates where one can take advantage of the full vehicle speed and where the streets are uniform. In cities the streets have various shapes, have many objects (e.g manholes) that might be confused with damage, and often have obstructions (e.g. parked cars).

A data collection approach similar to ours has been proposed before . In an earlier work [9], we described a system based on a structured light sensor that could be installed on buses and other vehicles that drive through the city regularly. The cost of installation and maintenance of such a system is much higher than using a smartphone. [6] makes use of smartphones, but in contrast to our approach they only use the accelerometer readings to find potholes.

2.1. Image-based Distress Detection

A survey of image based road distress detection is presented in [16]. Six crack segmentation methods are tested and compared in [15]. They found that a dynamic optimization-based method performs the best, but the computational requirements are high. In [13] cracks are detected by using image and entropy dynamic thresholding. [11] takes into account simultaneously brightness and connectivity for crack detection. Features which are calculated along every free-form path provide detection of cracks with any form and any orientation. A learning-from-samples paradigm is used in [14] to detect cracks. Those cracks are classified into different categories and finally the severity is determined by measuring the crack widths. [3] reviews many previous works that can be applied to crack detection and categorizes them into histogram, learning, morphology, filtering, and model-based methods.

A key limitation in all the above mentioned works is that the camera plane is required to be more or less parallel to the road, which would mean that the camera needs to be mounted outside the vehicle looking straight down. This contributes significantly to the cost and complexity of the hardware, because the camera requires weather-proofing, damage protection, installation and regular maintenance, calibration and a separate user interface within the vehicle. In contrast it is easy and inexpensive to mount a camera inside a vehicle, behind the windshield, as in our case (figure 1). This allows the data collection process to be performed much more frequently. Our method moves the system complexity to the computer vision software component, enabling it to deal with the near-perpendicular angle of the camera plane to the road plane. Another difference is that all the previous methods were designed just to detect cracks (or other thin structures) specifically, and show results only on images containing cracks against an uncluttered background. Our framework is more general in that it does not assume an uncluttered background (see figure 2 for example) and can potentially be extended to detect other kinds of pavement damage such as potholes, patches, etc.

[1] http://www.pavemetrics.com/

[2] http://www.greenwood.dk/profiler.php, http://www.roadware.com/products/ARAN-Subsystems/

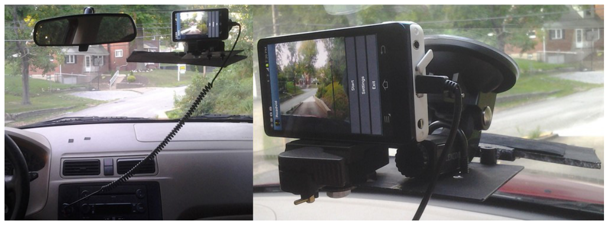
Figure 1: Mounting of the smart camera. On the left one can see its position within the windshield and the power cord from the cigarette lighter. On the right is a closeup of the suction cup, the shield, the snap-mount, and the display after startup.

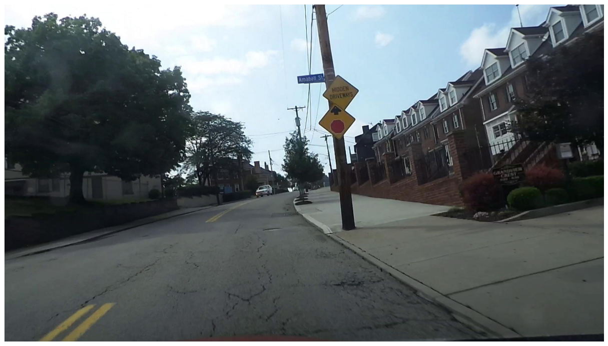
Figure 2: Typical input image, captured from the passenger's point of view.

3. Data Collection

The main device for the data collection was a Samsung Galaxy Camera. It is a compact camera that runs Android, has all the sensors and almost all functionalities of a smartphone. It was mounted on the windshield of a vehicle (figure 1 left). A snap mount was used to enable easy removal of the camera. Between this mount and a metal camera base, a rubber layer was installed to absorb vibrations. The plate was held in place by a suction cup with a short arm. We wanted to keep the arm short to further minimize vibrations. A black shield was positioned extending from the metal base to the windshield to prevent artifacts in the image caused by reflections from the windshield.

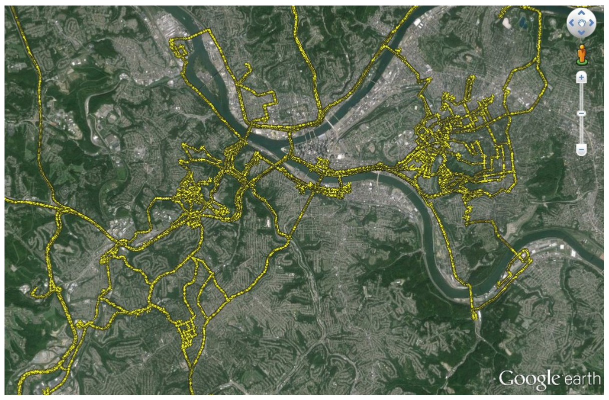

Figure 3: The yellow dots represent the locations of data collection in the Pittsburgh area.

Figure 1 (right) also shows the user interface seen once the data-collection app is launched. The user simply has to press the "start" button to start the collection. The following data was collected by the application: 1080p 10Hz videos, GPS, time, acceleration, relative orientation and the derived quantities gravity, linear acceleration, and absolute orientation. We used a private vehicle and collected more than 100 hours of video (which is about 4 million images), over a period of one year. Figure 3 shows the coverage of the Pittsburgh area by these images.

Since the data was collected all year long, we have multiple runs for each stretch of roads. This redundancy allows us to automatically select the images that are most suitable for analysis at each location, based on certain heuristics. Rain or snow would create artifacts in the images and sunshine would create shadows that can mimic road damage. Such conditions heavily affect the functioning of most vision algorithms. Having the GPS and time information for each image allows us to obtain the weather data. We selected images that were taken during the daytime and when the weather was overcast or mostly cloudy, which offers good lighting conditions. In addition, there was motion blur because the camera often underwent strong and sudden motions when the vehicle was moving (especially on roads with damage) and also because of residual vibration from the engine when the vehicle was not moving. To overcome this, we measured the magnitude of blur [5] in each image and only selected images for which the blur-score was below a threshold of 0.35. Also to avoid redundancy, we selected only one image for every stretch of 10m for processing.

4. Segmentation for Crack Detection

Unlike physical objects, cracks are not easily separable from the background road by color or strong boundaries. However, regions with cracks usually have unique texture

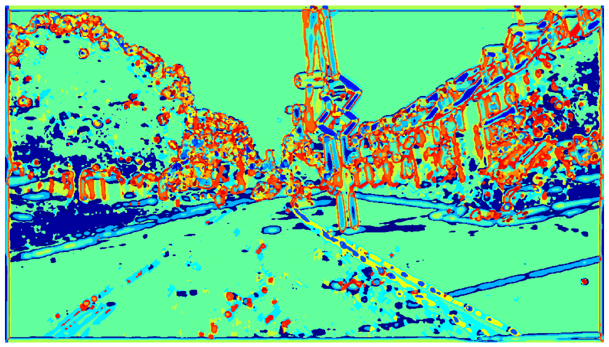

Figure 4: The raw image converted into a texton map. Distributions of textons within small regions are used to represent the texture within that region.

and therefore texture-based features can be used to identify them from the background. As a first step, we group pixels into regions of coherent texture by over-segmenting the image. The *texton* representation [8] (figure 4) is commonly used for analyzing texture at different regions within the image. The superpixels thus obtained are classified as either cracked (positive) or not cracked (negative) by a classifier trained on texture-based features. The pipeline is summarized in figure 5.

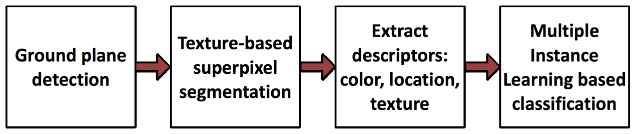
Ground plane detection
Texture-based superpixel segmentation
Extract descriptors: color, location, texture
Multiple Instance Learning based classification

Figure 5: Flowchart of crack detection pipeline.

Ground Segmentation Since we are interested only in detecting cracks on the pavement, the first stage of the pipeline is to segment out the ground plane from the rest of the image. For this purpose, we use the method described by Hoiem et al. in [7], the software for which is publicly available. The result of ground plane detection is shown in figure 6. Occasionally other non-ground objects like cars and pedestrians were included in the ground plane segmentation, but we found the results produced by this method to be quite accurate in almost all cases.

4.1. Obtaining Superpixels with Consistent Texture

Once the segmentation mask for the ground plane has been obtained, our next step is to produce an oversegmentation of the region enclosed by the mask. A desired characteristic in our case is to produce salient regions (or superpixels) that have consistent texture and color, as opposed to just color. To this end, we extend the SLIC superpixel algorithm [1]. SLIC works by performing iterative clustering of pixels over a five dimensional feature-space, given

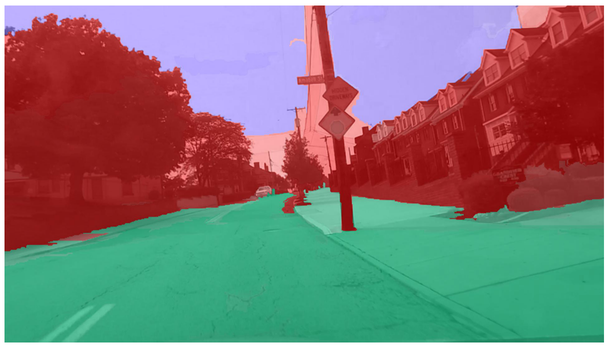

Figure 6: Ground plane detection using the geometric context algorithm: blue indicates sky, red indicates vertical structures and green indicates ground.

by the three CIELAB color values (L, A, B) for each pixel and its location (x, y) in the image. There are two parameters to tune: k, the number of superpixels required and m, a quantity that specifies the relative importance of spatial proximity over proximity in the color space. Larger m values result in more compact superpixels and smaller m results in superpixels that are more likely to adhere to image colour boundaries. In our experiments, we chose m and k to be 40 and 2000 respectively. These values were chosen to keep the superpixels sufficiently small, because in the case of larger superpixels, the texture information within them gets 'diluted' and it becomes difficult to distinguish cracked regions from the background. The distance measure used in SLIC for clustering pixels is:

$$D = \sqrt{d_c^2 + \left(\frac{d_s}{S}\right)^2 m^2} \tag{1}$$

Here S is the expected maximum euclidean distance in the image between pixels belonging to the same superpixel. For an image with N pixels, $S = \sqrt{N}/k$. To incorporate texture similarity in the clustering process, we define a texture distance d_t between any pair of pixels, by considering their local texton distributions. Given two normalized texton histograms a and b, computed within a 10×10 square window around two pixels, d_t is obtained as

$$d_t = 1 - \text{HIK}(a, b) \tag{2}$$

where HIK is the similarity measure obtained using the histogram intersection kernel.

$$\text{HIK}(a, b) = \sum_i min(a_i, b_i) \tag{3}$$

This is integrated in to the clustering distance measure with a weight w (set to 60, in our implementation) as

$$D = \sqrt{d_c^2 + \left(\frac{d_s}{S}\right)^2 m^2 + w\,(d_t^2)} \tag{4}$$

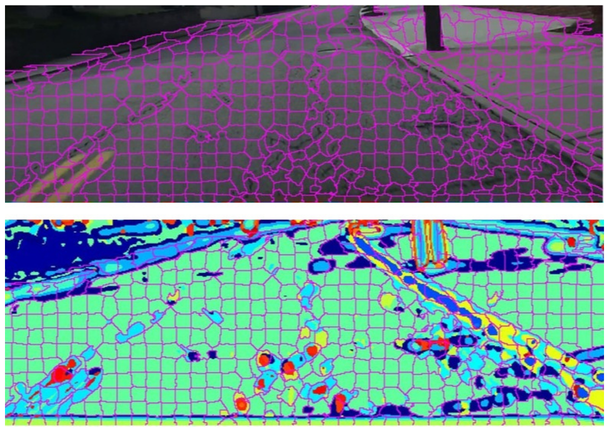

Figure 7: Superpixels overlaid on the color image (top) and the texton image (bottom). Note how the superpixels naturally separate textured areas from non-textured areas. (Best viewed on a screen, with magnification)

Our variation of SLIC produces superpixels on the ground plane as shown in figure 7. We have observed in practice that this kind of clustering does indeed tend to cluster pixels belonging to cracked regions into their own superpixels, separate from the background undamaged regions of the road which are characteristically textureless.

5. Classifier for Distress Detection

Human annotators tend to differ a lot in their perception of the boundaries and extent of cracks, which have widely varying and intricate structures. The tediousness of labeling such complicated patterns leads to negligence and unreliability in the annotation process. In our experiments we observed that on the same images, different annotators exhibited a wide range of biases and attention to detail, giving rise to very different annotations (see figure 8). To address this, we aggregated labels from multiple annotators in the learning stage. Additionally, we made weaker assumptions regarding the positive labels which allowed us to naturally express the learning problem using multiple instance learning.

Feature Descriptor We compute descriptors based on color and texture for each superpixel, associated with a binary label indicating the presence or absence of cracked pavement. The descriptor is 138-dimensional, and is composed of color elements, location in the image, histograms of textons and Local Binary Patterns (LBP) [12] within the superpixel.

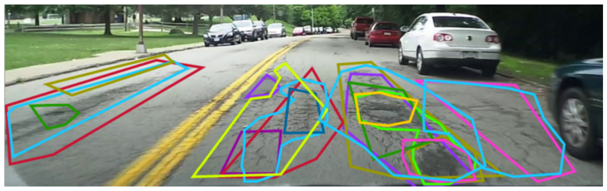

Figure 8: Each color represents an individual annotator, working independently. Different skill levels and biases of annotators lead to different perceptions about the boundaries and extent of cracks.

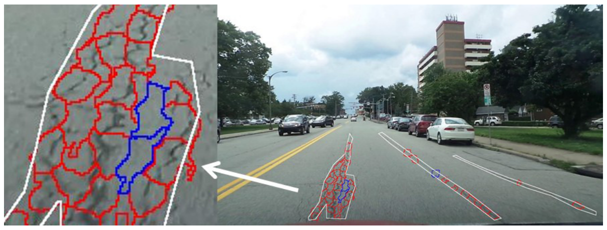

Figure 9: Regions of the image are labeled as containing cracks (white border) by annotators. Most of the superpixels inside the region are rich in texture (red), but some superpixels resemble undamaged road and have very little texture (blue). (Best viewed on a screen, with magnification)

Features	Number
Mean RGB	1-3
HSV for mean RGB	4-6
Hue histogram	7-11
Saturation histogram	12-14
Centroid (x,y)	15-16
10th, 90th percentile x	17-18
10th, 90th percentile y	19-20
Texton histogram	21-70
Multi-scale LBP histogram	71-138

Some of the features we use were adapted from [7]. The bulk of the descriptor is made up of the texton and LBP histograms, which is in line with our premise of distinguishing cracked regions using texture. Color elements as features facilitate distinguishing lane markings and grass from the road and location-based features help in supressing sidewalks and other regions on the side of the road.

5.1. Multiple Instance Learning for Classifier Training

The final classification stage involves training a machine learning classifier with data obtained from human annotators. A group of annotators were given instructions to label regions on the ground as positive (cracked road) or negative (includes undamaged road, lane markings, sidewalks, grass,

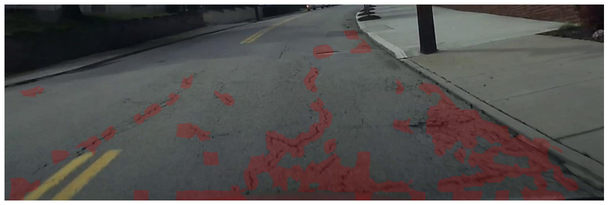

Figure 10: Classifier result: detected crack regions are shown in red.

vegetation etc.) However, a complication arises at this stage due to the way in which annotators typically label regions. Certain kinds of common cracking patterns like alligator cracking are spread out over an area and tend to enclose patches that resemble the undamaged background sections of the road, as shown in Figure 9. These trapped patches get segmented into their own superpixels since their texture is quite different from the cracked region that encloses them.

Human annotators often do not make this distinction and they label a solid polygonal area in the image around the cracked region. As a result, these trapped patches that are locally similar to the background region, get assigned the same training labels as the cracked regions (positive), while the other background superpixels get assigned different training labels (negative). We found that up to a third of superpixels labeled as positive were actually closer to the background in texture. When a classifier is trained on such data, it produces a lot of false positives on testing.

This situation is an typical application for algorithms that make weaker assumptions on the training labels- in particular, multiple instance learning (MIL). In MIL, we assume that the training data are in *bags*, which are labeled positive or negative. Each bag contains multiple *instances* of data, with the constraint that a bag labeled 'positive' must contain at least one positive instance and a negative bag must have only negative instances. In our application, each bag is a single, user annotated connected component, and its instances are the superpixels contained within that region. Since the objective in our problem is to classify superpixels as positive or negative, we require an instance level classifier that learns from data labeled at the bag level. The mi-SVM approach described in [2] solves exactly this problem, using a modified version of the standard SVM optimization function with the additional multiple instance learning constraint described above. The SVM trained using the MIL cost function gives significantly better results than a SVM trained in the standard manner.

6. Experiments

A total of 220 images collected around the city were used for training. Each image was labeled by multiple annotators. As shown in figure 9, most of the annotations included some textureless regions inside the labeled regions. To train the mi-SVM classifier, we used the Multiple Instance Learning Library [17] along with the LIBSVM package [4]. We used an RBF kernel with C and γ parameters determined using cross-validation and grid-search. For the test dataset, we had a total of 12 annotators work on a set of 140 images. Not all the annotators were required to label each image; on average 5 people labeled each image and each image had at least 2 annotators. Each annotator's work was of a different quality judging by the detail of the boundary they drew around cracks and the time spent on each label. The time spent by each annotator per image varied from 30 seconds to 5 minutes approximately, as reported in their feedback.

To the best of our knowledge, none of the existing methods to detect pavement distress from images can work on the kind of images we use. Therefore, we compare our classifier's performance against individual human subjects. While evaluating, we followed a leave-one-out strategy with the human annotators. That is, given a set of annotations in the form of binary masks $A = \{a_1, a_2, \ldots, a_N\}$ for an image, N different versions of the ground truth ($G = \{g_1, g_2, \ldots, g_N\}$) were generated by combining the annotations of $N-1$ annotators.

$$g_i = \text{round}\left(\frac{1}{N-1}\sum_{\substack{j=1\\ j\neq i}}^{N} a_j\right) \tag{5}$$

The summation and rounding was done at the pixel level. For each version of the ground truth g_i, we compare the performance of the classifier with the annotator i, whose annotations a_i were excluded while computing g_i. For each image, the classifier prediction was evaluated against all versions of the ground truth.

The precision-recall curve of the classifier obtained from this evaluation is shown in figure 11. As the plot shows, the combined precision (0.5) and recall (0.7) of human annotators is itself low. This validates our earlier observation that human performance on this task is very subjective and has high variability. This can be further verified from figure 12 which shows the performance of the classifier compared to individual annotators, on the subset of images labeled by that particular annotator. There is a wide variation in precision and recall even among the best annotators, and in each case the classifier's performance comes close to that of the annotator.

Qualitative results of the classifier prediction are shown in figures 13 and 14. Regions where the classifier returned a positive result for cracks are shaded red in the images. The ground truth positive labels are indicated by the white outlines. Figure 13 shows instances where the classifier results matched well with the annotator-drawn ground truth labels. Most of the cracked regions are detected correctly

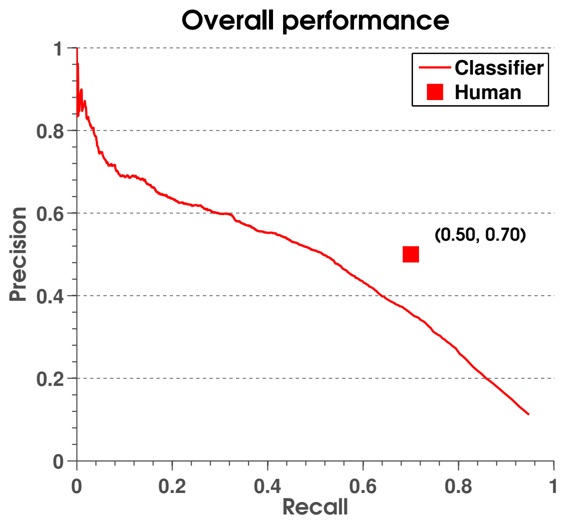

Figure 11: Precision - Recall curve for classifier.

and there are very few false positives. Figure 14 shows instances where the classifier performed poorly. In figure 14a, the algorithm was able to detect only a few sections of a longitudinal crack. In figure 14b, a lot of false positives occurred on the edges of the white crossing. In figure 14c), a large network of cracks can be identified by human annotators, but is too faint for the algorithm to recognize the texture. Despite these issues, the overall performance of the algorithm is not too far from that of the human annotators.

We used the described algorithm to process approximately 25,000 images captured citywide. On a computer with a quad-core Intel-i7 CPU and 8GB of memory, our unoptimized code takes around 60 seconds to process each image, from scratch. For each image we computed a score, indicating the approximate level of road-damage (due to cracks) at the location where the image was captured. To obtain the score, we first compute the weighted sum of the areas of the superpixels classified as 'positive' by our algorithm, with the classifier confidence in each superpixel acting as its weight. The ratio of this sum to the visible road area within a fixed region was taken as the score. All area computations were done by projecting the segmentation to the road plane (assuming that the road plane normal is perpendicular to the principal axis of the camera). The resulting scores were used to generate a map showing the approximate road conditions on a citywide scale (figure 15).

7. Conclusion and outlook

We note that in this application, it is not critical for an algorithm to classify all the superpixels correctly to be useful. It only needs to be reasonably close to the performance of human annotators. The scoring produced by the algorithm (area of the segmented regions projected on to the

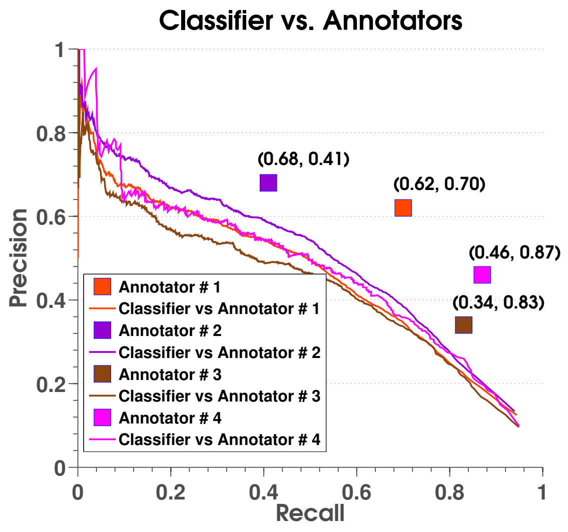

Figure 12: Precision - Recall curves for the classifier, compared to individual annotators.

road plane) will be consistent whereas there will always be variations between human annotators. Once the superpixels are classified it is straightforward to calculate ratios of damaged and undamaged road surface. In contrast, human inspectors only provide subjective estimates of the areas or scores for the severity of damage. In addition, our framework is very general and can be trained for other road objects like potholes, patches, lane markings, manholes, road signs, etc. This can become the basis of a comprehensive inspection and inventory system for public transport infrastructure.

We have shown that it is possible to build a road distress monitoring system that can evaluate the surface quality of roads on a regular basis, close to human performance and at very low cost . The equipment can be purchased for less than $1000, the labor required is small, and the analysis can be done on a standard computer. Having completed the development of a prototype system, we are in the process of testing it in a pilot project with the City of Pittsburgh. We will mount the data collection system on their vehicles, analyze the collected data and load the results into their asset management system. First indications are that the system will be very useful for them.

References

[1] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Susstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(11):2274–2282, 2012.

[2] S. Andrews, I. Tsochantaridis, and T. Hofmann. Support vector machines for multiple-instance learning. In *Advances in neural information processing systems*, pages 561–568, 2002.

[3] S. Chambon and J.-M. Moliard. Automatic road pavement assessment with image processing: Review and comparison. *International Journal of Geophysics*, 2011, 2011.

[4] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[5] F. Crete, T. Dolmiere, P. Ladret, and M. Nicolas. The blur effect: perception and estimation with a new no-reference perceptual blur metric. *Human vision and electronic imaging XII*, 6492:64920I, 2007.

[6] J. Eriksson, L. Girod, B. Hull, R. Newton, S. Madden, and H. Balakrishnan. Pothole patrol: Using a mobile sensor network for road surface monitoring. In *Proceeding of the 6th international conference on Mobile systems, applications, and services*, June 2008.

[7] D. Hoiem, A. A. Efros, and M. Hebert. Geometric context from a single image. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 1, pages 654–661. IEEE, 2005.

[8] J. Malik, S. Belongie, T. Leung, and J. Shi. Contour and texture analysis for image segmentation. *International journal of computer vision*, 43(1):7–27, 2001.

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

[10] J. Miller and Y. Bellinger. *Distress Identification Manual for the Long-Term Pavement Performance Program*, fourth revised edition, June 2003. Report No. FHW-RD-03-031.

[11] T. S. Nguyen, S. Begot, F. Duculty, and M. Avila. Free-form anisotropy: A new method for crack detection on pavement surface images. In *Image Processing (ICIP), 2011 18th IEEE International Conference on*, pages 1069–1072, 2011.

[12] T. Ojala, M. Pietikainen, and T. Maenpaa. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(7):971–987, 2002.

[13] H. Oliveira and P. L. Correia. Automatic road crack segmentation using entropy and image dynamic thresholding. In *Proc. European Signal Process. Conf.(EUSIPCO'09)*, pages 622–626, 2009.

[14] H. Oliveira and P. L. Correia. Automatic road crack detection and characterization. *Intelligent Transportation Systems, IEEE Transactions on*, 14(1):155 –168, march 2013.

[15] Y.-C. Tsai, V. Kaul, and R. M. Mersereau. Critical assessment of pavement distress segmentation methods. *Journal of Transportation Engineering*, 136(1):11–19, Jan. 2010.

[16] K. Wang and O. Smadi. Automated imaging technologies for pavment distress surveys. Transportation Research Circular E-C156, Transportation Research Board, 2011.

[17] J. Yang. Mill: A multiple instance learning library. *URL http://www. cs. cmu. edu/˜ juny/MILL*, 2008.

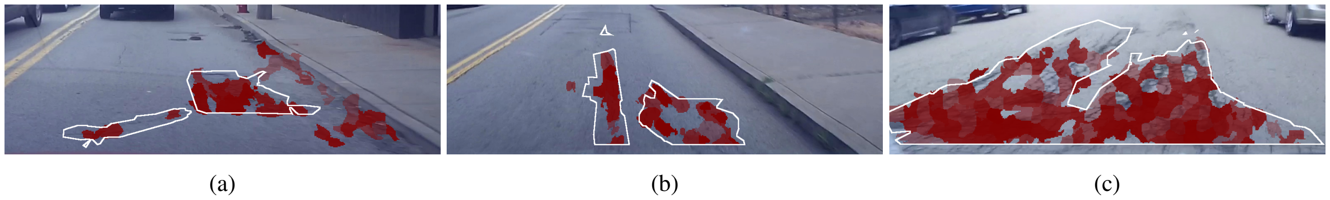

Figure 13: Instances with good agreement between ground truth labels and classifier output: most of the human-annotated cracks are also detected by the classifier.

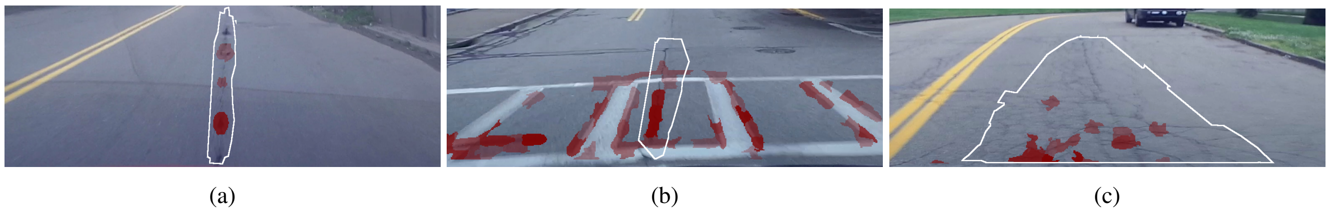

Figure 14: Examples of classifier failure: In (a), most sections of the long crack are missed by the classifier; in (b), the classifier produces a lot of false positives along the edges of the pedestrian crossing and in (c), there is a big network of cracks that is just visible, but too faint for the algorithm to detect the texture.

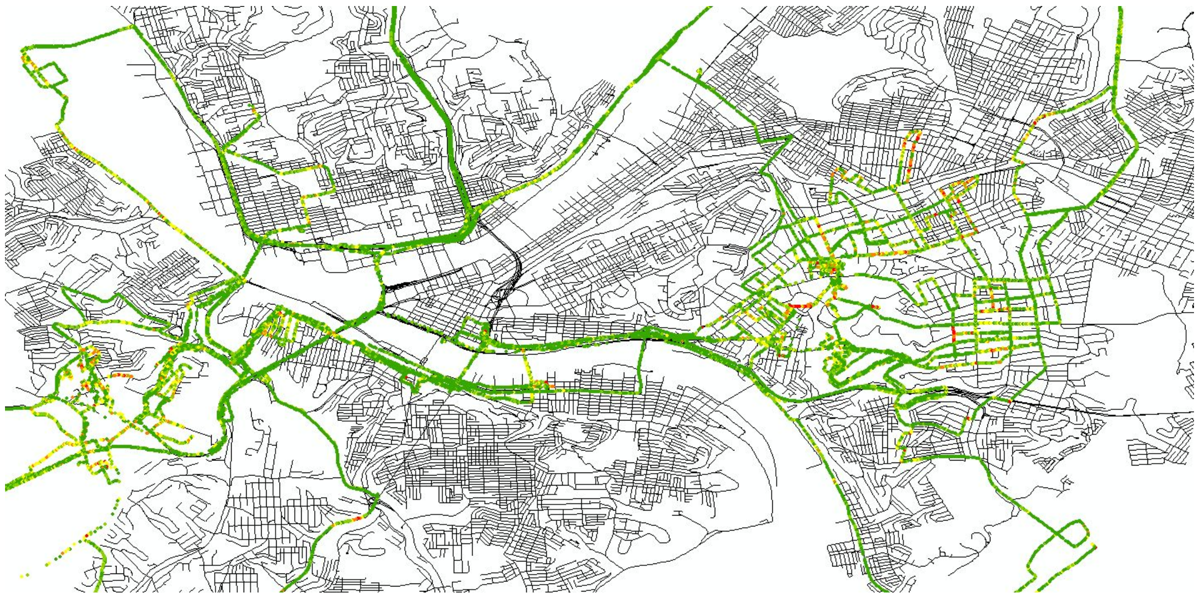

Figure 15: Citywide road damage severity due to cracks: green - low to none, yellow - medium, red - severe. Note that only a subset of the city's road network has been covered.