Advanced Engineering Informatics 32 (2017) 11–25

Contents lists available at ScienceDirect

Advanced Engineering Informatics

journal homepage: www.elsevier.com/locate/aei

A real-time automatic pavement crack and pothole recognition system for mobile Android-based devices

A. Tedeschi, F. Benedetto *

Signal Processing for Telecommunication and Economics Lab, University of Roma TRE, Via Vito Volterra 62, Roma, Italy

ARTICLE INFO

Article history:
Received 19 May 2016
Received in revised form 12 December 2016
Accepted 14 December 2016
Available online 24 December 2016

Keywords:
Computer vision
Distress recognition
Android
Pavement distress
Infrastructure monitoring

ABSTRACT

Due to the rapid growth of vehicles and traffic accidents caused by road pavement defects, road safety has become a pressing concern worldwide. For this reason, Countries and Federal States have started focusing their resources on the analysis of civil infrastructures to assess their safety and serviceability. The analyses are performed by specialized teams of inspectors and structural engineers who manually inspect road infrastructures and provide detailed reports about the detected pavement distresses and their magnitudes. This work aims at providing a new system able to detect the framed distress using solely the computational resources provided by a mobile device To reach this goal, an automatic pavement distress recognition system based on the OpenCV library is developed and embedded in a mobile application, enabling the recognition of three common pavement distresses: Pothole, Longitudinal-Transversal Cracks, and Fatigue Cracks. Our method, tested on several Android mobile platforms, is able recognize the pavement distresses of interest reaching more than 0.7 of Precision, Recall, Accuracy, and F-Measure. This application promises to improve the on-site work of inspectors by decreasing the time required to perform inspections while ensuring, at the same time, a higher level of accuracy.

1. Introduction

The ever-increasing number of vehicles and traffic accidents has made road safety a major reason for concern worldwide. The factors playing a role in road fatalities can vary from drivers' errors to vehicles' malfunctions and issues related to roads environment.

The U.S. Department of Transportation has conducted a statistical analysis about the number of traffic fatalities occurred during 2014, estimating that, in that year alone, 32.675 people died in motor vehicle traffic crashes [1]. In the same year, more than 25.700 people died on the road in Europe (i.e. the equivalent of a medium town) and more than 200.000 people suffered serious life-changing injuries [2]; it is estimated that for every death on the road in Europe, there are 4 permanently disabling injuries, 8 serious injuries, and 50 minor injuries.

Even though the data depict a dramatic scenario, it is important to underline that the trend seems to be changing for the better. Several researches to enhance road safety have been carried out, defining new strategies and approaches and exploiting new and advanced methodologies. In particular, the technology provided by the Information and Communication Technologies (ICT) can prove valuable in finding new and innovative solutions to road safety issues, developing new systems and tools which can improve the investigation and analysis of civil infrastructure issues, such as the well-known pavement distress.

In the past years, a great deal of research efforts have focused on this field, developing innovative methods and algorithms to improve the manual visual inspection of the road pavement. Manual visual inspection is the most common approach to evaluate the physical and functional condition of civil infrastructures (i.e. roads, bridges, etc.) [3], while giving an indication of their serviceability. These evaluations, performed by certified inspectors and structural engineers, are based on manual measurements of pavement roughness and of the magnitude (i.e. size and gravity) of the surface distress deflection. The inspections are done at regular time intervals or even after disasters causing changes in the pavement surface, and are then followed by ad-hoc and detailed reports and evaluations. The analysis of these evaluations plays an important role in urban road maintenance and rehabilitation (M&R) projects. Through their analysis and the correct classification of pavement distresses, it is possible to improve the decision making procedure in pavement management systems (PMS) and to plan ad-hoc repair interventions. To reach these goals, several methodological and fuzzy approaches, such as the Analytic Hierarchy Process (AHP) and the Technique for Order Preference by Ideal Situation (TOPSIS)

* Corresponding author.
E-mail address: francesco.benedetto@uniroma3.it (F. Benedetto).

http://dx.doi.org/10.1016/j.aei.2016.12.004
1474-0346/

depicted in [4], are designed. As a consequence, road agencies are able to plan repair interventions, minimizing the life-cycle costs of the deteriorating structures [5]. However, such approaches require automated distress surveys and measurements to decrease the time required to make road condition reports and to improve their accuracy and reliability.

Recently, several road agencies have started to involve automated and semi-automated systems based on Computer Vision (CV) techniques to improve the collection and analysis of the pavement data. Their choice is well sustained by the availability of low cost and high quality visual sensing devices, such as digital cameras. This has led to a new research trend in the field of CV applied to infrastructure monitoring systems. As a matter of fact, one of the goals of CV is the object detection and recognition task which allows to automatically detect instances of a specific object in digital images and video captured in real-time [6]. Hence, it is possible to create ad-hoc automatic classification systems based on CV approaches that are able to process the acquired frames and automatically detect and recognize a specific pavement distress, if it is present in the analyzed frame. Most of these approaches do not allow to perform a real-time detection of the road pavement distress for two reasons: (i) the data collection and analysis are two separate stages; (ii) they require sophisticated preprocessing and recognition methods that exploit a large amount of computational resources for a correct detection. For these reasons, mobile devices have so far only been used to collect data, but never as means for a real-time detection of pavement distresses.

This work proposes a new approach to automatically recognize three common categories of pavement distress (i.e. Potholes, Longitudinal and Transversal Cracks, and Fatigue Cracks). Based on CV techniques, such approach exploits the computational resources provided by an Android-based mobile device.

Following the same methodological approach of [4] to the analysis of pavement distresses, our application aims at improving the manual visual inspection process performed by structural engineers and specialized inspectors, by creating automatic reports of pavement inspections and providing the following information: (i) the identified pavement distress categories; (ii) their numbers; (iii) the images of the distress detected; (iv) the date and time of the distresses' acquisition process; (v) the location of each distress through the GPS module of the device.

This information allows to create a database of pavement distresses that can be exploited to trace the degradation of a road in a specific timeframe by querying the database with the date, time and GPS locations of interest. As a matter of fact, exploiting the information about the date and the GPS location provided in a report for each pavement distress, it is possible to: (i) assess which are the main categories of pavement distresses that affect the considered road and to plan ad-hoc road maintenance; (ii) trace the numbers of inspections and when they were performed. To reach this goal, we exploit the well-known Software Development Kit (SDK) of the Android platform [7] involving the OpenCV library [8], which provides optimized methods for the image processing and the classification tasks.

Through OpenCV, we have designed three cascade classifiers, one for each distress of interest, based on Local Binary Pattern (LBP) features [9], creating a new Automatic Pavement Distress Recognition (APDR) system. The three proposed classifiers are able to detect a specific pavement distress category (i.e. Fatigue Cracks, Longitudinal-Transversal Cracks, and Potholes) and, once embedded into the mobile application, they allow to process the frame acquired by the device's camera, displaying the detection output to the end-user in real-time.

The Android SDK provided us with all the methods and mechanisms to create the mobile application and the access to the devices' resources such as the permissions to access to: (i) the device camera in order to allow the end-user to start the semi-automatic visual inspection; (ii) the GPS sensor embedded in the mobile device to get the current position of the detected distress; (iii) the local device memory to store the acquired information like the last frame of the acquired distress and a text file. In particular, the text file will contain a table where each line represents a detected pavement distress composed of the distress category name, its numbers in the analyzed frame, the date and time of the acquisition, and its location.

Our application allows a real-time identification of the distress through the video stream acquired by the device's camera, and shows the output of the recognition process, highlighting the section of the frame that contains the distress. This application promises to provide innovative potentialities for all the agencies involved in roads management, improving the on-site work of structural engineers and specialized inspectors.

The remainder of this work is organized as follows. Section 2 presents a categorization of the main pavement distress categories, while Section 3 provides a deep review of the state-of-art approaches for automatic detection and recognition of pavement damages. In Section 4, we discuss the rationale and the design behind our Automatic Pavement Distress Recognition system embedded in an Android application. Lastly, we present the results of our evaluation in Section 5, while Section 6 briefly depicts our conclusions.

2. Manual pavement distress classification

The manual visual inspection of the physical and functional conditions of roads is the most important task to assess the quality and availability of this type of civil infrastructure. Inspections are performed only by opportunely trained teams of structural engineers and specialized inspectors and they are done at regular intervals, aiming to provide detailed reports about the conditions of the road's infrastructures. To aid inspectors in performing this task, each Country provides its own distress identification manual containing information such as: (i) the distress name; (ii) a description of essential features and notes to aid the identification; (iii) the severity level (i.e. low, moderate or high); (iv) how to measure the distress; (v) a list of attributes; (vi) examples of the distress; (vii) a set of possible causes; and (vii) the recommended remedies. Two well-known examples of manuals are the Catalog of Road Defects (CORD) [10] and the Distress Identification Manual [11] that provide a deep categorization of the most common pavement distresses. Such distresses can be grouped in the following five macro-categories: (i) Cracks are among the most common pavement defects and they can be of different kinds, such as longitudinal, transversal or fatigue cracks. The longitudinal and transversal cracks are parallel and perpendicular to the pavement centerline, respectively, while the fatigue crack, namely Alligator crack, is composed of interconnected cracks; (ii) Potholes are bowl-shaped holes of various size involving one or more layers of the asphalt pavement structure and their size and depth can increase whenever water accumulates in the hole; (iii) Patches are portions of road pavements that were replaced due to a deterioration of pavement surfaces; (iv) the Surface Deformations category groups distresses that cause changes in pavement surfaces. A typical distress of this category is rutting, a longitudinal depression worn into the wheel-paths; (v) Surface Defects is a category that classifies distresses generating defects in pavement surfaces, such as bleeding. This distress is related to the accumulation of bituminous binder on the pavement surface and it can be found in the wheel paths. Even though manual visual inspection can achieve high levels of accuracy and precision in the classification of pavement distresses, its main disadvantages are the time required to perform

it and the fact that it lacks objectiveness, since it depends on the use of manuals and on the experience of the team.

To overcome these issues, in this work we provide an automatic pavement distress recognition system able to detect the following three categories of pavement distress: Potholes, Longitudinal and Transversal Cracks, Fatigue Cracks. Fig. 1 shows an example of these distresses. In particular, the pictures were taken in Rome (Italy) during road inspections performed for this work.

3. Related works

As discussed by Koch et al. [3], the number of algorithms and systems based on CV approaches for civil engineering applications has exponentially increased thanks to the availability of high quality and low cost visual sensing technologies. Most of these methods are built on different image processing techniques to highlight specific aspects of the images and improve the probability of correct detection. Koch et al. summarize in [3] the common CV pipeline exploited in the detection of defects on civil infrastructures.

The pipeline can be divided in six macro-stages as shown in Fig. 2, where each stage is related to a set of general CV methods as follows: (i) *image acquisition* is the basic stage of the CV pipeline. It allows the system to acquire the data and process them through a digital camera exploiting the available APIs of the chosen CV library; (ii) the *pre-processing* stage is composed of basic methods to enhance the quality of the image, such as restoration and noise reduction; (iii) the *segmentation* stage allows to identify the region of interest in the image through the application of the edge detection and clustering approaches; (iv) through the *features extraction* stage, it is possible to retrieve specific textures or shapes that can be exploited during the recognition process; (v) the *object recognition* is the main stage of the CV pipeline since it performs the detection of the distress exploiting the preprocessed image. Typical approaches are clustering, Support Vector Machine (SVM) and Artificial Neural Networks (ANN); (vi) the *structural analysis* stage provides a deep understanding of the scene through a 3D reconstruction (e.g. Optical flow and stereo reconstruction). Exploiting the proposed CV pipeline or part of it, several state-of-art applications focus on the detection of the three most common categories of pavement distress: Cracks, Potholes and Patches.

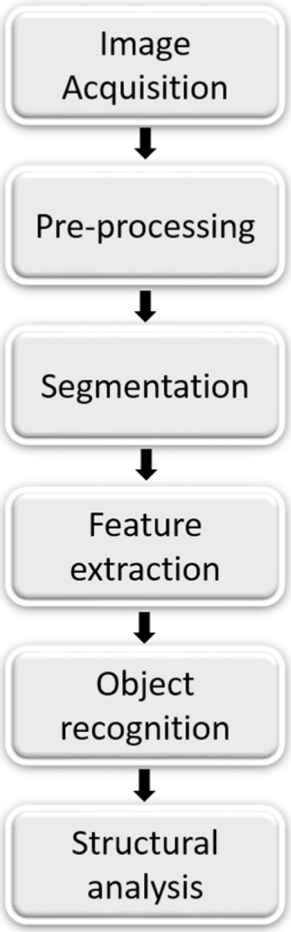

Fig. 2. Pipeline of the computer vision stages to recognize defects in civil infrastructure [3].

3.1. Cracks detection

In recent years, a large number of methods have been used for crack detection. For instance, an automatic recognition of cracks

Fig. 1. Examples of (a) Pothole; (b) fatigue crack; (c) longitudinal crack; (d) transversal crack.

based on an ANN is designed in [12,13]. Xu et al. in [12] have proposed a back propagation (BP) ANN based on three hidden layers to detect a pavement crack in an image. The approach relies on the concept of pattern matching, which is applied to the binary version of the image divided in sub-images. The detection is then applied to each sub-image, differentiating them between crack and crack-free image. To perform this task, the authors selected four parameters that are extracted from the binary sub-image and provided as input for the BP-ANN. The ANN is composed of three hidden layers and trained by using pavement images captured from highways. Bray et al. in [13] use two hierarchical ANNs to obtain a true detection of cracks. The first ANN gets the image's density and histogram features as input to assess the presence or the absence of defects. The provided output is a set of images with cracks used as input for the second ANN that recognizes the type of defect, distinguishing among Fatigue, block, horizontal and vertical cracks. Li et al. [14] developed an algorithm to detect cracks using a genetic algorithm and SVM. Through the genetic algorithm, the authors find the optimized input parameters for the SVM, which is trained through a set of training samples. Once the cracks' features are extracted from the image, the SVM performs the classification and provides the results. Oliveira et al. in [15] designed a two-stage unsupervised approach to detect pavement cracks in an image involving the Unsupervised Information-theoretic Adaptive Image (UINTA) filtering method [16] used to reduce images' entropy. Through the pre-processing stage, the system applies a set of filters to reduce the pixel intensity variance and saturation, and then a one-class Parzen classifier is exploited to create a set of non-overlapping sub-images. Finally, each sub-image is classified as cracks or cracks-free image by comparing the mean and the standard deviation of the sub-image with their neighbors or by UINTA filter. Despite the good performance provided by this approach, the UINTA filtering method has a high computational time and cost. An improvement is proposed in [17], where Oliveira et al. propose a parallel implementation of UINTA filtering method decreasing the computational time required to apply the algorithm and achieving the same performances of the previous work. Then, Medina et al. described in [18] an automatic detection system for road cracks combining both visual appearance and geometrical information. To acquire and store data, a multi-device system is designed and implemented on a vehicle performing the inspection survey. Once the visual appearance and geometrical information are obtained, the AdaBoost algorithm [19] is applied to combine the results and improve the detection.

Wu et al. design in [20] a crack-defragmentation method (i.e. MorphoLink-C) to connect the fragmented cracks to improve the accuracy of a crack detector based on an ANN. To reach this goal, they exploit the fragment grouping process, which is created by using the dilation transform and fragment connection obtained through the thinning transform. Exploiting the MorphoLink-C, they trained an ANN to detect the cracks in an image.

An image processing approach based on minimal paths to detect crack distresses in road pavement is proposed in [21]. The proposed approach exploits the geometry and intensity properties of cracks using an ad-hoc algorithm to find the minimal path from a considered set of pixels, i.e. cracks pixels. Oliveira et al. propose in [22] a toolbox for Matlab environment to preprocess the input images and use them to detect and categorize the type of crack. The proposed preprocessing methods include image smoothing, pixel intensity normalization and saturation, while the algorithm for the detection of a crack in an analyzed image is based on pattern classifications using the block-based or pixel-based approaches. Finally, the crack characterization allows to classify the detected crack as longitudinal, transversal or miscellaneous, assigning a severity level. Wu et al. propose in [23] a method for the extraction of cracks that affect road pavements. The method exploits a new preprocessing approach based not only on the typical image enhancement, but also on the combination of Canny edge detection and hyperbolic tangent (HBT) filter. Then, the Curvelet transform is applied, obtaining three levels composed of different numbers of images. Such images are elaborated by maximum-approximation and mean-detail (MaxMean) fusion algorithms, which allow to detect the cracks in the analyzed frame. A noninvasive sensing technique for the automatic detection of cracks on asphalt-surfaced pavement is provided in [24]. Kapela et al. based this technique on optical images of the pavement surface analysis defining two stages: (i) the data collection stage, exploiting a vehicle composed of three cameras and a lighting system; (ii) the data analysis process, exploiting a new algorithm. In particular, the new algorithm is based on six blocks that preprocess the image to provide the histograms of oriented gradients (HOGs) through the parallel computing provided by Nvida CUDA. Finally, an SVM classifier is used to detect the cracks in the analyzed images. Baltazart et al. in [25] defines a new crack-detection approach, namely Minimal Path Selection (MPS), assuming that a crack contained in an image is characterized by dark pixels embedded within the background. Through the MPS, the authors are able to obtain an accurate segmentation of the crack pattern and to estimate its width.

3.2. Potholes and patch detection

The detection of potholes could be based on different methods: the two-dimensional (2D) vision-based approach, the three-dimensional (3D) reconstruction-based approach and the vibration method.

Ryu et al. in [26] designed a 2D vision-based method to detect potholes. The approach exploits an optical device mounted on a vehicle that collects images of the road and sends them to the pothole detection algorithm. The algorithm exploits a segmentation stage to distinguish the pothole region from its surroundings, using the histogram shape-based thresholding and maximum entropy methods. Then, the candidate regions are filtered through the median filter to remove noise such as cracks. Finally, the decision is made by comparing the candidate regions with background regions through histogram and standard deviation features. Koch and Brilakis in [27] designed an approach based on the segmentation of the acquired frame in defect and non-defect areas using histogram shape. These areas are preprocessed to extract features (i.e. shape and texture) and find the candidate areas and the non-defect texture. Finally, using spot filter responses, the texture is extracted from the candidate areas and compared to the non-defect texture to find the pothole region. Koch et al. provide an enhancement of such approach in [28]. In particular, the advantage of proposed method is twofold: (i) the proposed algorithm is able to incrementally update the texture signature for no-patches areas, in order to progressively create a representative texture template by using a sequence of several frames; (ii) a vision tracking method is involved to track the detected potholes over a sequence of pavement video frames. Through such improvements, the method proves to be robust to abrupt surface and illumination changes.

Another important improvement of the detection results provided by [27,28] is proposed in [29]. Jog et al. developed an approach that detects and assesses the severity of potholes combining both 2D recognition and 3D reconstruction. The 2D reconstruction method relies on the texture extraction process to divide the image in segments of distress and non-distress areas. The texture is then compared with a dataset of healthy pavement textures. The 3D recognition method performs a detection of the features and applies a pairwise feature matching approach to detect the pothole areas. Finally, the combination of the outputs

allows to correctly identify and assess the potholes in the analyzed video.

Recently, works such as [30,31] have proposed new approaches to the detection of potholes in road pavement. Zhang et al. introduce in [30] an algorithm based on stereo vision to detect a pothole, providing the size, volume and position of the potholes. As a matter of fact, the stereo vision based approach provides 3D measurements allowing the system to gather information about the size of detected potholes without using additional sensors. The proposed approach provides a real-time disparity calculation algorithm to generate a disparity map used to allow the system to detect a pothole. Finally, to apply the proposed algorithm the authors exploit two cameras mounted in parallel onto the optical rail and connected to an Arduino synchronization board.

Li et al. propose in [31] a novel method to combine the processing of 2D images with the Ground Penetrating Radar (GPR) data collected through a GPR system. The approach is composed of three stages. In the first one, the image and the GPR data are preprocessed to reduce the noise, enhancing the quality of the data. Then, the pothole classifier is applied to the GPR signals reflected by potholes. Finally, exploiting the information obtained from the classifier, it is possible to estimate the position and the dimension of the detected pothole, mapping it to the related image.

Conversely, Tai et al. in [32] propose an automatic road anomaly detection based on the analysis of the vibrations captured through a mobile device and a remote server. The idea is to collect the abnormal vibration of vehicles caused by roads' anomalies, such as potholes or missing pavements, through the accelerometer module. In particular, a mobile device was installed inside the storage box of a motorcycle tracking the location of anomalies through the device's GPS module. To perform the detection, they created a client-server application where the client is the mobile application that acquires and processes the data and then sends them to the server program. The server makes the final decision exploiting an ad-hoc SVM.

Furthermore, Mednis et al. propose in [33] a real time pothole detection system exploiting accelerometer information provided by an Android-based device. Following the same idea proposed by Tai et al., they apply the Z-thresh algorithm to measure the acceleration amplitude at Z-axis, while the Z-diff and the standard deviation of Z are used to compute the difference between the two amplitude values and the standard deviation of vertical axis acceleration and G-Zero. However, such approach requires to drive the vehicle into the pothole, which could damage the vehicle and the driver.

Even though these CV approaches offer high quality results in the detection of road infrastructures' defects, most of them cannot directly exploit the computational resources of a mobile device. For instance, approaches based on noise reduction, UINTA filtering and ANN, require a lot of computational resources which rapidly drain the device's battery, failing to provide real-time results. Currently, very few approaches like [32,33] exploit mobile devices' resources to collect data and then process them through other systems.

An automatic pothole and hump detection system is proposed by Madli et al. in [34]. They have created an Internet of Thing (IoT) application based on an ultrasonic sensor and a GPS receiver connected to a PIC Microcontroller. The ultrasonic sensor measures the distance between the body of the vehicle and the road surface. Such distance, on a smooth road surface, is used as the threshold's system and if the distance measured by the ultrasonic sensor is greater than the threshold, a pothole is detected. Otherwise, if the distance is smaller, a hump is detected. The collected information is forwarded from the PIC microcontroller to the Android server module, which stores the collected detections and their locations. Finally, the Android application compares the local position of the vehicle with the ones stored in the server's database and if the distance is lower than 100 m, an alert message is displayed to the end-user.

Finally, another common pavement distress is the patch, an area of the road pavement repaired using new materials. Even if a patch is a repair by definition, it is still considered as a distress because it can create height differences (i.e. humps). In addition, it is characterized by a different shade of gray, if compared with the surrounding pavement areas. The authors in [35] exploit a clustering method to analyze the frames acquired during the collection process in order to detect the possible patches or potholes. However, this approach is not able to distinguish the former from the latter. An approach based on multiscale second order moment is developed in [36] to detect the patches in a concrete pavement image. In particular, the approach is composed of three stages: the application of the image pyramid transform on the original frame as preprocessing; the computing of the second order moment exploiting the preprocessed image; the detection of patches by exploiting the extracted features. The detection of patches in asphalt pavement is proposed by Radopoulou et al. in [37] by applying morphological operations that segment images through the different colors of the pixels. In particular, the algorithm applies the texture information and four filters creating the corresponding feature vectors through the computing of the standard deviation of the regions' gray intensities. Finally, the approach is able to identify a patch if the length of its texture feature vector is equal to the texture feature vector of the surrounding area. Radopoulou et al. proposed an improvement of their algorithm in [38] presenting a technique based on five different preprocessing steps (i.e. gray-scale transformation; median filtering; histogram equalization; image binarization; morphological operations) applied to the pavement video frames obtained during the data collection process. Through this preprocessing operations, they obtain a set of frames containing objects with enhanced contours that are used in the detection process based on the texture extraction and comparison in order to detect the frame area that contains the patch. A classifier based on SVM is proposed by Hadjidemetriou et al. in [39]. The SVM classifier is trained to detect changes in color and texture of the patch and non-patch areas by extracting a set of features generated from the histogram and two texture descriptors. For each frame of an analyzed pavement video, the method extracts the feature and create its binary version in order to classify each region as patch or no-patch area.

Mednis et al. in [40] exploit the accelerometer sensor embedded in an Android-based device processing the acquired data to detect a pavement pothole in real-time. The proposed system uses a limited amount of computational resources, detecting potholes by exploiting different kinds of algorithms on a set of collected data. However, the proposed method requires the end-user to calibrate the device in order to set a start-point for the accelerometer and set the observation window size for collecting and processing the data. Conversely, Douangphachanh and Oneyama in [41] exploit the data acquired through the accelerometer of a mobile device to analyze the road's roughness. In particular, the authors use a system composed by the smartphone accelerometer, a GPS and video recorder, and a laptop as the core of the system which performs the data processing and analysis.

3.3. Detection of different pavement distress

Hu et al. in [42] design a pavement distress detection system based on texture analysis and shape descriptors. In particular, they exploit the gray-level co-occurrence matrix to extract the feature used for texture recognition and classification, requiring a large amount of computation resources.

Banharnsakun et al. in [43] develop a pavement distress recognition system based on the combination of artificial bee colony

(ABC) algorithm and ANN. The proposed system, namely ABC-ANN, captures the pavement image and segments it into distress and non-distressed regions by using a threshold method. The ABC algorithm is employed to identify the optimal threshold. Then, three features are extracted from the distress regions and given as input to the ANN, which will classify the kind of distress. The system provides reliable classification with a high level of accuracy. However, the method's performance depends on the quality of the results provided by the segmentation stage.

Salari et al. in [44] propose a segmentation approach based on Genetic Algorithm (GA) to classify pavement potholes and cracks. In particular, the method applies an objective function to choose the optimal threshold for the segmentation function. Then, the features related to the distress regions are forwarded to a three-layer feed-forward ANN to classify the detected distress.

Seraj et al. in [45] define a SVM classifier based on the LIBSVM library [46] in order to detect pavement anomalies such as potholes by using the inertial sensors embedded in the device. The system is able to accurately detect three different event classes of anomalies (i.e. Severe, Mild, and Span) displaying the detected class to the end-user. Despite the good performance, the system is not able to highlight the distress in the frame acquired through the camera and to provide a fine-grained classification of the detected anomalies.

3.4. Drawbacks

From the analysis of the literature review, several issues can be highlighted. Most of the considered approaches are off-line methods, based on sophisticated preprocessing techniques that require a large amount of computational resources and time. Moreover, they can only detect the presence of a pavement distress, without being able to discriminate between one pavement distress and another. In such off-line approaches, the collection and analysis of the acquired pavement data are two different phases, and therefore a real-time detection of the pavement distresses is not possible, making these approaches not suitable to improve the visual inspection. Even though the off-line approaches provide good performance and a high-level of accuracy, they are not suitable for a real-time detection that requires a fast analysis of the current pavement condition. The real-time detection of pavement distresses provides several benefits, such as the possibility to send real-time reports about pavement conditions of the currently inspected site to a coordination center that can rapidly plan ad-hoc repair interventions, reducing the risk of traffic fatalities. In addition, to improve the on-site analysis it is becoming mandatory to use compact and inexpensive devices such as mobile devices.

Some recent approaches employ mobile devices to provide a real-time detection of pavement distresses using the information collected by the device's inertial sensors. In particular, some of these works, such as [32,33,40,41], only exploit the mobile device as a mean to collect the data provided by the accelerometer module, which will then be processed by a third party system, or to alert drivers about imminent road defects. In addition, these works are able to detect only potholes and patches. Other approaches, such as [45], exploit the inertial sensors of the mobile device and a SVM classifier to detect the distress' severity without providing a fine-grained classification of the detected anomalies. Recently, some works (e.g. see [47]) have exploited mobile device's resources for object detection and recognition in different fields not related to pavement distress detection by applying the CV techniques. Even though these approaches are based on mobile devices, they do not exploit all the features of these devices, allowing end-users to recognize only a few pavement distress categories, such as potholes.

To overcome these issues, the rationale of the off-line approaches should be first adopted, and then optimized according to the limited amount of computational resources provided by a mobile device. Following this rationale, we propose a CV-based approach that aims at addressing the highlighted issues. In particular, our system relies solely on the computational resources and modules of mobile devices in order to provide an automatic detection of three different categories of pavement distress, i.e.: potholes, longitudinal-transversal crack, and fatigue crack. To do so, the proposed approach uses three LBP cascade classifiers, which process the frames acquired from the device's camera in real-time and provides a visual output to the end-user, showing the current position of the distress and its category. Additional information, such as the date and the location, of the detected distresses are acquired by the system in order to create an automatic report of the current visual inspection and to improve the on-site work of structural engineers and specialized inspectors.

4. Automatic pavement distress recognition system for mobile devices

After any pavement inspection operation, both structural engineers and specialized inspectors have to provide a report specifying the pavement distress, its severity and size, the location, the description of the repair, and the date of the inspection. In addition, they should also assess how many distresses of the same category are in the area of interest and perform all these operations manually.

Our work aims at creating a new mobile application to simplify the creation of the report through the typical object recognition task of the CV. To reach this goal, we have developed an innovative mobile application for Android-based devices, which exploits the methods defined by OpenCV creating an Automatic Pavement Distress Recognition (APDR) system. In particular, the ADPR system is able to recognize and highlight in real-time three different categories of pavement distress (i.e. Fatigue Cracks, Longitudinal and Transversal Cracks, and Potholes) and their location in the analyzed frame, exploiting three cascade classifiers based on LBP features [9]. Using the LBP cascade classifier defined by OpenCV, we exploited three training sets of positive samples, one for each pavement distress, and one training set of negative samples. The positive and negative samples are images acquired during several road inspections performed in Rome (Italy) and they were labelled and modified in order to maximize the performance of the system. Then, the classifiers are embedded in the mobile application and directly applied to the frames of the streaming video acquired by the device's camera and allow to display the detection results to the end users in real-time (i.e. augmented reality). Additionally, once the detection and recognition processes are completed, the detection data are saved in a specific folder of the mobile device's local memory (i.e. '*\APDR\Date*'). In particular, the application saves the information about the category of the recognized distress, the location information obtained by the GPS module of the mobile device, and a frame which represents the recognized distress.

In the following sections, we present the main structural choices behind the design and development of the proposed application explaining how our recognition algorithm works.

4.1. Application design

We designed our application for the automatic recognition of pavement distress relying on the main principles of software analysis and design. In particular, we exploit the *Layer* and *Model View Presenter* (MVP) design patterns [48,49]. The reason for using the

MVP pattern is that it allows us to separate software responsibilities across three main components: (i) the *Model* component groups the data that should be displayed in the User Interface (UI); (ii) the *View* component renders the User Interface (UI) elements; (iii) the *Presenter* component manages the interaction between the UI and the application logic by retrieving and formatting the data that should be displayed to end-users. Through this first division in three component, we involve the *Layers* pattern which allows us to define four layers, each of them managing a specific application behavior. The four layers are: (i) the *model* layer is used to model the domain of interest (e.g. the distress and their features); (ii) the *business logic* layer is the main layer. It coordinates the application and performs the processing of the acquired frame to detect and classify the distress. In detail, it provides the methods to manage the classifier and its results, and to estimate the size of the detected pavement distress. It also provides two sub-layers: the first one supplies some I/O functions, while the second one provides the methods written in C++ that exploit the OpenCV library; (iii) the *presentation* layer supplies the methods to manage the interaction between the *UI* and *Model* layer, and exploits the methods provided by the business logic layer; (iv) the *UI* layer shows the real-time classification results of the distress of interest, allowing users to interact with the application through the UI elements. Through these design patterns, we define an organized structure that can be replicated in other mobile OS-platforms since it does not require any specific feature of the Android OS. In addition, such organization allows us to easily manage and update the application without modifying its structure. For instance, it will possible to replace the core of the application (i.e. the three LBP cascade classifiers) with an updated version or with new classifiers to improve the overall performance of the ADPR system.

4.2. Real-time implementation by Android and OpenCV

In this section we highlight the main implementation aspect of the procedure used to develop a prototype of our automatic classification system. In particular, we exploit the well-known OpenCV library to create its core and we choose Android as target mobile platform because of its worldwide diffusion and open source nature [50,51]. In addition, Android provides the Android Native Development Kit (NDK), a toolset to develop parts of applications using the C/C++ programming language [52]. We combine the Android NDK framework with the OpenCV library to create our real-time APDR system. We adopt OpenCV in our application because it is an open source library written in C/C++ and it provides several methodologies and functions in the field of computer vision, such as image processing and object recognition.

4.2.1. Proposed training for the LBP cascade classifier

We exploit the OpenCV's cascade classifier based on Local Binary Patter (LBP) features [9] instead of the well-known Haar feature-based cascade classifier [53]. The choice is well sustained by the fact that the LBP features has inherent immunity to a vast variety in lighting conditions making them suitable for the detection of pavement distresses and an LBP classifier provided by OpenCV is optimized to be exploited by a mobile device, it is faster than a Haar one and it perfectly matches the real-time and computational cost required for mobile devices. This kind of cascade classifier is exploited by several face and car detection applications but only few works, such as [54], exploit the LBP features (and not the cascade classifier) in combination with more sophisticated classifiers, like SVM classifier, to improve the detection of a pavement distress. In addition, it could be extremely difficult to exploit one of the approaches analyzed in the literature review without considering a remote server that performs the classification and provides the result to the end-user. In fact, to exploit a remote server, a web service has to be created to provide a communication between the mobile device and the remote server used to send the acquired frame, increasing the time required for the detection of a distress. However, it is important to mention that the performance of a cascade classifier based on LBP features depends on the quality of the training dataset and parameters exploited to train the classifier. Our contribution is twofold: (i) we design three different datasets to train the three LBP classifiers, one for each pavement distress category of interest, to automatically classify pavement distresses; (ii) we exploit the trained classifiers in a new and innovative mobile application for Android-based devices by using OpenCV to manage the recognition process. A cascade classifier is based on two main stages: (i) the training stage is used to train the classifier with a set of samples (positive and negative samples); (ii) the recognition stage allows to detect the object of interest in the analyzed frame. As a matter of fact, a cascade classifier is composed of several simpler classifiers applied in cascade to a region of interest of the analyzed image during the training and the classification stages. In detail, at each stage, the classifiers exploit their built-in classification function based on LBP visual descriptor and mathematics as depicted in [9]. Even though both stages are important, it is crucial that the training of a cascade classifier is performed properly in order to reach high level of accuracy and precision.

To enable our application to automatically detect pavement distresses in frames acquired from streaming videos, we create three cascade classifiers based on LBP features for the recognition of three categories of pavement distress, Fatigue Cracks, Longitudinal and Transversal Cracks, and Potholes, respectively. This choice is well sustained by the fact that each LBP cascade classifier is able to detect a specific pavement distress, allowing the ADPR system to not only detect a pavement distress but also to recognize the category of the distress. Actually, the definition of a unique classifier for the detection of the three considered pavement distresses would allow detecting the presence of a distress but it would not be able to distinguish among them due to the impossibility to label the positive samples by using a LBP cascade classifier.

Following the main guidelines provided by supervised learning [55], the training of each cascade classifier is based on two sets of samples, positive and negative samples. Hence, for each classifier we created a set of positive and negative samples composed of the grayscale images in Portable Gray Map (*.pgm*) format. In particular, for each cascade classifier we defined a positive set of samples composed of images representing the distress of interest from different perspectives and in different light conditions, while we adopted only one set of negative samples. This set is composed of arbitrary grayscale images that should look like the positive ones except that they do not contain the objects that we want to detect. The negative samples should contain typical objects of the urban scenario (i.e. asphalt, cars, manholes, grid pavements, etc.) that might be improperly identified as the objects of interest causing the classifiers to perform a wrong recognition.

Once the sets of positive and negative samples are defined, the creation of three files in *.vec* format is required by OpenCV to train the classifiers. The *.vec* format is a binary format that contains the samples involved in the training of the cascade classifier. Finally, we train the classifiers launching the command opencv_traincascade, as depicted in the OpenCV documentation [56]. As a result, we have three different files in XML format which contain the information to perform the automatic recognition of pavement distresses. The XML file is composed of two main tags: (i) the *stages*, a set of stages that contains a set of features such as thresholds and weights to perform the detection; (ii) the *features* tag is composed of an array of *rect* tags which describe the geometry of the feature as a tuple (x, y, width, height) where the first two represent the

coordinates in pixel of the first point, while the next two are the width and height of the sub-rectangle. In particular, the information about the size of the sub-rectangle allows the ADPR system to highlight the size in pixels of the detected distress. Unfortunately, such information cannot be directly converted to the international system of units because the system does not know the ratio between pixels and meters. To overcome this issue, the system should be trained to recognize a new object (of which the size is known) which, placed in the same shot as the pavement distress, would allow to estimate the real size of the distress. However, such approach has several constraints, such as the definition of an accurate mechanism for the calibration of the camera and a perfect 90-degree view to estimate the size.

Finally, we integrate the XML files in the Android application managing them through the OpenCV library and Android NDK and enabling the recognition process of pavement distresses. In particular, following the main guideline provided by [57] to enable leverage C/C++ code in our Android application, we create a new C/C++ class, namely *ADPRCore*, which provides the methods to: (i) preprocess the acquired frames; (ii) load and release the classifiers; (iii) perform the detection; (iv) create the output frames with the overlaid information. Such methods are then exploited by the Java classes defined in the business layer that allow the *Presenter* to manage the pavement distress recognition process.

4.2.2. Proposed approach

In Fig. 3, we present a simplified flowchart that depicts the main steps of the recognition approach of our application and the report creation. Once the application is running and the camera is initialized, the *Presenter* component loads the three cascade classifiers in memory in order to be responsive during the recognition process. Then, through the methods provided by the business logic layer, the *Presenter* launches the detection algorithm on each acquired frame until the end-user stops the acquisition process. Once the frame is acquired by the device camera, the algorithm processes the frame converting it from RGB to grayscale color domain, discarding the unnecessary color component. The grayscale frame is then processed by the three LBP features-based cascade classifiers at the same time starting the recognition process. When one of the three classifiers detects a distress, it provides a rectangle that matches the area of the frame in which the distress is detected. Such information is combined with the original RGB frame drawing a rectangle which highlights the distress location in the frame. Finally, the output frame with the overlaid sub-rectangles, the name of the detected distress category and their numbers are displayed to the end-user in real-time.

If the classifiers do not detect a distress in the processed frame, the algorithm continues the recognition cycle until the end-user stops the recognition process. When this happens, the *Presenter* gathers the current location through the GPS modules, exploiting the Google Location APIs [58,59]. Then, the *Presenter* saves the last acquired image of the distress in JPG format and the obtained recognition results (i.e. the name of the recognized distresses, their numbers and locations, and the acquisition date and time) in a text file, namely *Report_[DATE].txt*. Such data are stored in the local memory of the mobile device under the folder '*\APDR\Date*' by exploiting the *File* APIs provided by Android SDK [60]and an ad-hoc asynchronous task, namely *SaveAsyncTask*, based on the *AsyncTask* class of Android [61]. Then, a popup window is displayed to the end-user who can choose whether the ADPR system must continue the detection of new distress or not. If the end-user wants to continue the acquisition, the *FrameAcquisition_stop* variable is set to false and the ADPR system starts to acquire new frames through the device's camera. Otherwise, the *Presenter* releases the cascade classifiers and exits the recognition process redirecting the end-user to the main page of the application, where he/she can access to the stored data and launch a new recognition process. Finally, the application exits from the recognition process and redirects the end-user to the main page of the application, where he/she can access to the stored data and launch a new recognition process.

The application enables the real-time recognition process through the multithreaded programming, allowing us to create a new thread that performs the recognition process and updates the main thread when a new distress is detected. This allows us to create a responsive application which does not freeze the main thread to perform the distress recognition, therefore allowing end-users to dynamically change the point of view of the camera.

5. Experimental results

In this section, we report the experimental results carried out to test our application, evaluating the performance of the APDR system embedded in the mobile application for Android-based devices. We tested our application on three Android-based devices based on different hardware and software characteristics to verify the portability of our system on both older versions of Android (i.e. JellyBean) and hardware components which are still diffused. As a matter of fact, it is important not only to create an efficient system but also to provide a global solution that can be exploited regardless of the type of device used. In Table 1, we report the characteristics of the three devices anonymizing their brands.

5.1. Application overview

In Figs. 4 and 5, we present an overview of our application in the presence of pavement distresses. When the application is launched, the *MainPage* is shown allowing the inspector can interact with the application through the two provided buttons.

Through the *Browse Results* button, the inspector can browse the previous detections results, while clicking on the *Start Recognition* button the inspector is redirect to the *RecognitionPage* and the recognition process is launched. As mentioned in the Section 4.2, the APDR system highlights the area of the frame that should contain the detected distresses and count how many distresses of the same category are present in that frame. The name of the detected distresses, its quantity, and the box that highlights the area in the frame are displayed to the end-user using a specific color as shown in the following figures.

Fig. 5 shows the results of the APDR system in the presence of the distresses of interest. For each detected pavement distress, it is possible to know its location in the frame, highlighted by a sub-rectangle of different colors (i.e. red for potholes, yellow for fatigue cracks, and green for longitudinal-transversal cracks), the name of the distress category and its number in the analyzed frame. The other acquired characteristics are saved in the text file, once the end-user interrupts the current acquisition. As the end-user exits the recognition process, he/she can analyze the data saved in the device's local memory.

Additional classification examples are provided in Fig. 6, proving the capability of our algorithm to correctly identify the pavement distresses of interest.

In the proposed examples, it is possible to note that even if the system is able to detect and highlight the area of the frame where the distress is located, the sub-rectangle does not cover the entire framed distress. Due to such behavior, it is not possible to accurately estimate the size of the distress detected solely through the information provided by the sub-rectangle. Finally, once the end-user stops the recognition process, he/she is redirected to the main page of the application to browse the new reports. In particular, the end-user will be redirected to the ADPR folder that will contain all the inspection results organized in folders. Each folder

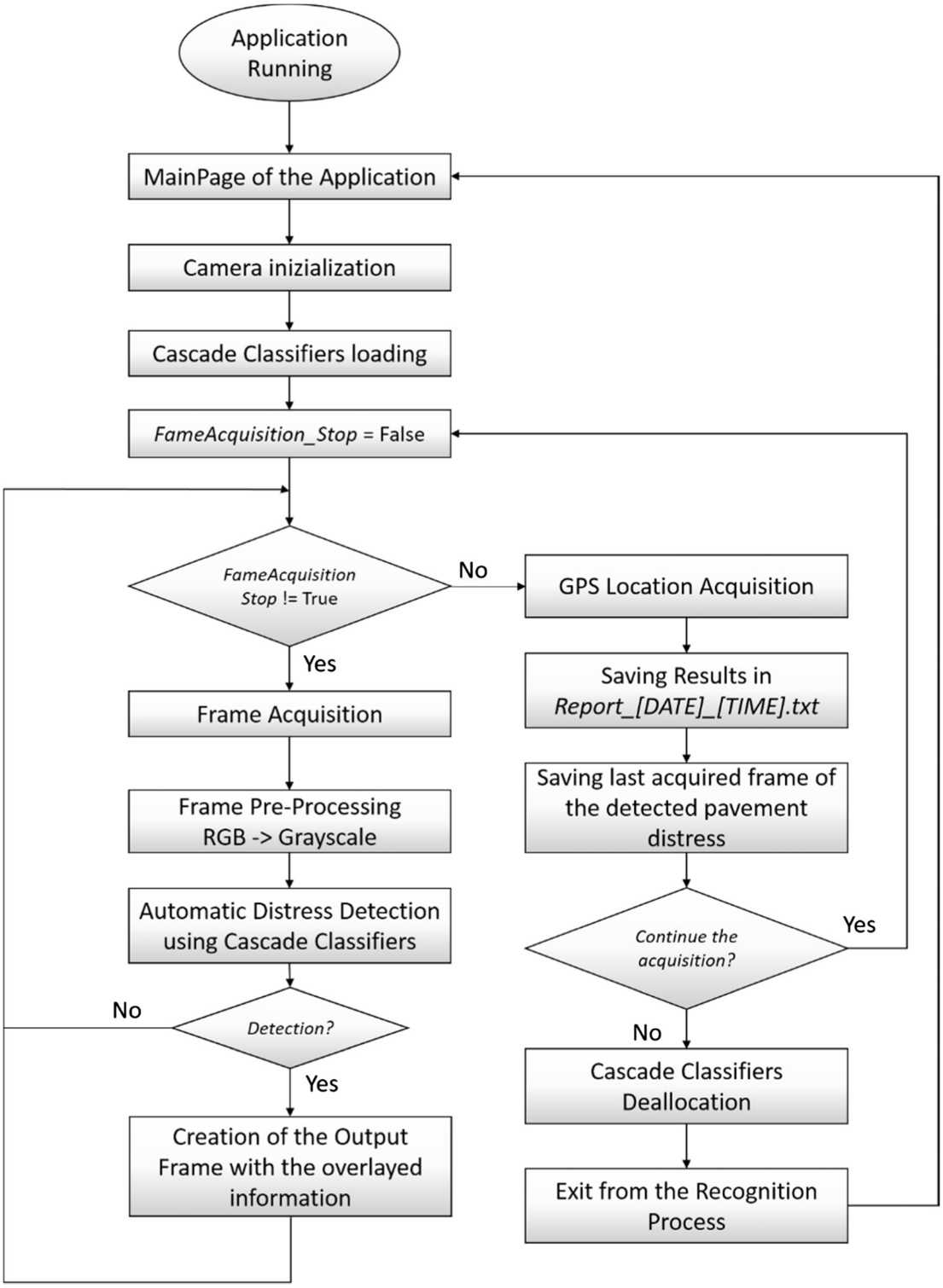

Fig. 3. The simplified flow chart of the automatic classification of pavement distress.

Table 1
Characteristics of the three devices.

Name	Android OS version	CPU	RAM	Camera	
				Megapixels	Resolution
Device1	Jelly Bean (4.3)	Dual Core 1 GHz	1 GB	5 Mp	2592 x 1944 pixel
Device2	Lollipop (5.0)	Quad-Core 2.3 GHz	2 GB	8 Mp	3264 x 2448 pixel
Device3	Marshmallow (6.0)	Quad-core 2.7 GHz	3 GB	13 Mp	4128 x 3096 pixel

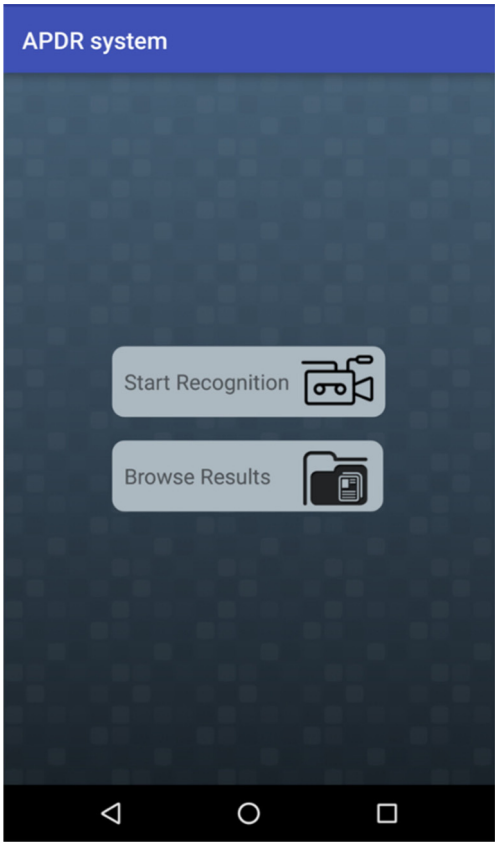

Fig. 4. Main page of the application.

is named with the inspection date and contains the text file and the figures of each recognized distress. Fig. 7 shows an example of the results obtained during the road inspection performed in Rome in July 15th 2016. It is possible to notice that each image related to a distress is named with the string '*Index[VALUE]_[TIME]*', where *[VALUE]* is an integer value that is increased by one at each correct recognition, while *[TIME]* is the time in which the detected distress is saved in the local memory of the device. This allows the system to create a unique identifier for each image and allows the user to easily gather the related information for each distress in the report file. As a matter of fact, the report file is divided in six tab-delimited columns that report the information of each distress identified by the index value.

5.2. Performance evaluation of ADPR system

We evaluate the performance of our method versus the state-of-the-art approaches described before. In particular, we focus our preliminary tests on the performance analyses of each classifier in terms of Precision, Recall, Accuracy and F-Measure, exploiting two different file format for the training sets. These metrics are based on the following values: (i) True Positives (TP) represent correct decisions of the classifier when positive samples are detected; (ii) True Negatives (TN) are the correct decision of the classifier in presence of negative samples; (iii) False Positives (FP) are the negative samples that are labelled as positive; (iv) False Negatives (FN) happen when a positive sample is rejected by the classifier. It is possible to define the above metrics as follows.

Precision is the ability of the classifier to reject the irrelevant samples for a classifier, as given by Eq. (1):

$$Precision = \frac{TP}{TP + FP} \tag{1}$$

The Recall metric allows us to measure the ability of the classifier to recover the highest number of relevant samples:

$$Recall = \frac{TP}{TP + FN} \tag{2}$$

The accuracy metric allows us to assess the correctness of the classifier by computing the ration between the true (positives and negatives) results and the total number of examined samples. It is computed as:

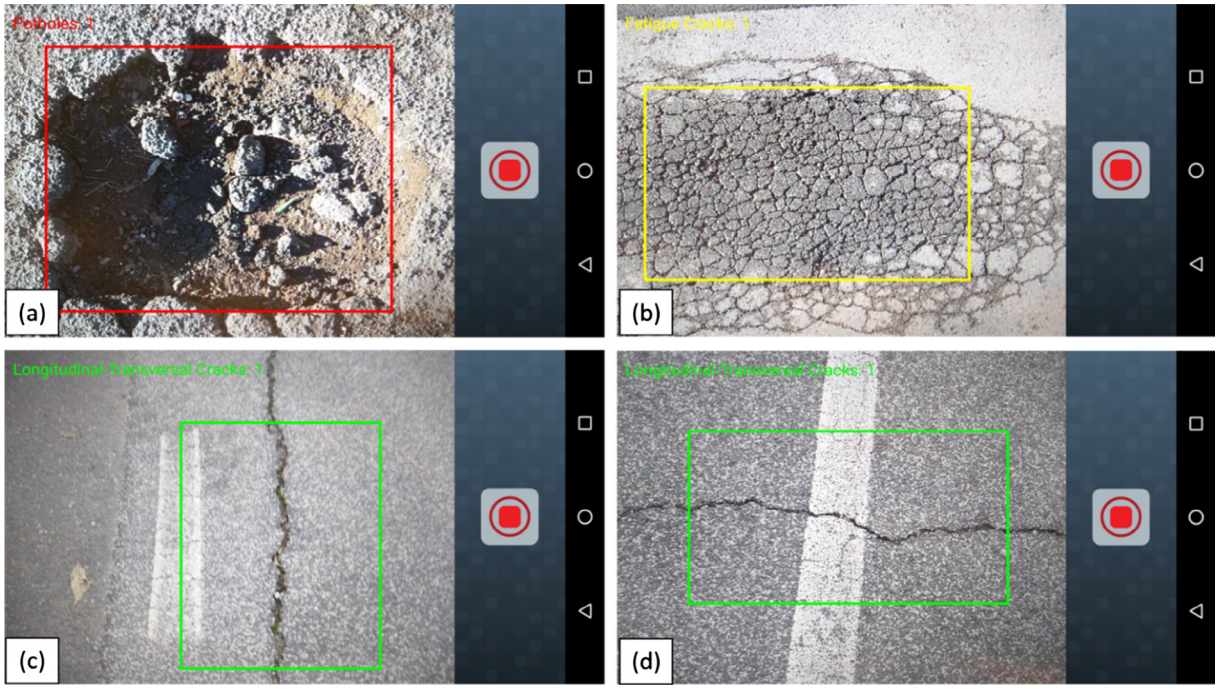

Fig. 5. The detection outputs of the APDR system in presence of (a) Pothole; (b) longitudinal crack; (c) transversal crack; (d) fatigue crack.

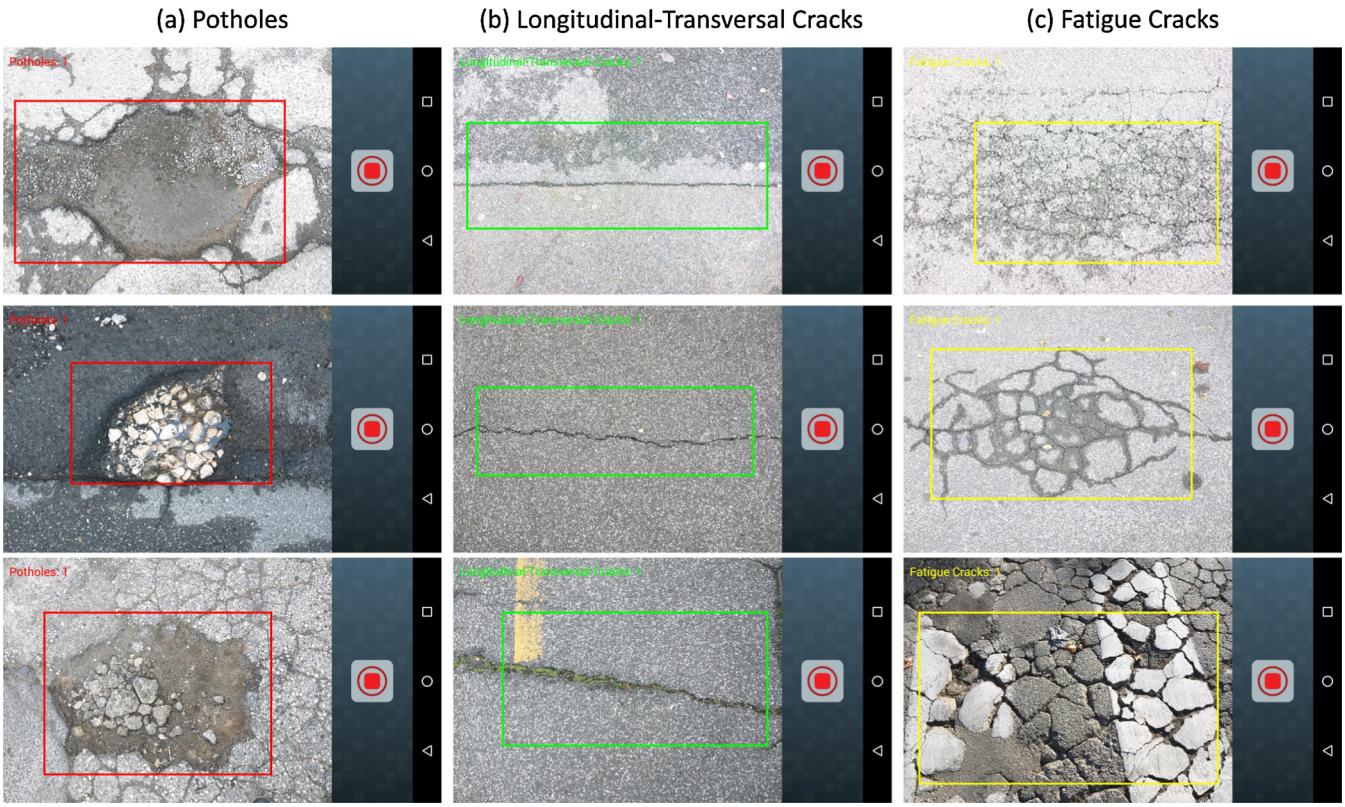

Fig. 6. Examples of correct detections of (a) Potholes; (b) longitudinal-transversal crack; (c) fatigue cracks.

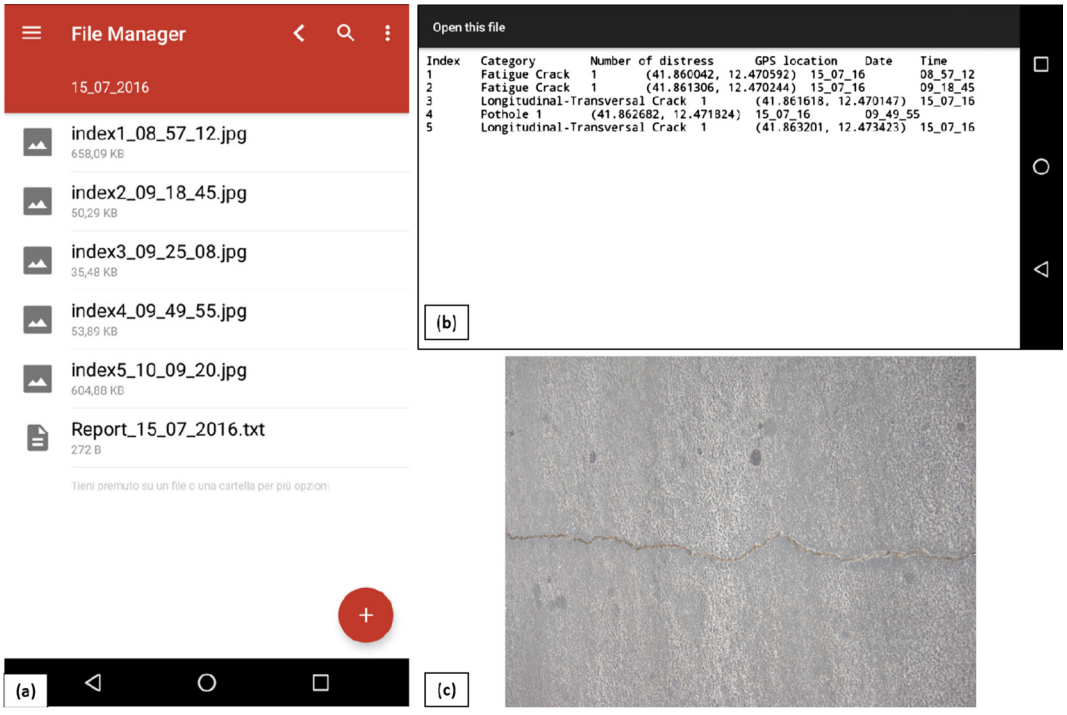

Fig. 7. The outputs contained in the ADPR folder store on the local memory of the device: (a) overview of the folder contents; (b) the text file with the detected pavement distresses and their information; (c) an acquired frame of the first longitudinal-transversal crack.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{3}$$

Finally, the F-Measure is the harmonic mean of precision and recalls values summarizing in a single factor the effectiveness of a classifier and it is given by:

$$\mathrm{F-Measure} = 2 * \frac{Precision * Recall}{Precision + Recall} \quad (4)$$

To perform the tests, we trained the classifiers as follows. For each of them, we created a set of positive samples of real pavement distresses manually collected and one set of negative samples, each set being composed of 1000 samples in TIFF file format. Then, we modified the file format of each training set's samples exploiting two different formats, *grayscale pgm* and *jpg*, to assess how the file formats and the image colors can affects the recognition process. Finally, we defined three test sets composed of 1000 images where: 30% of the images are negative samples; 20% of the images are positive samples used in the training set; 50% of the images are new positive samples. The samples used to train and validate the three proposed classifiers and the ADPR system have been manually acquired in the city of Rome and its province through a professional camera with 18 Megapixels. The samples are saved in RAW format, and then converted in TIFF file format exploiting an ad-hoc script written in Python. The acquired distresses are accurately selected in order to discard the out of focus images and to provide reliable results. These test sets are then used to evaluate the performance of each classifier. The obtained results are shown in Table 2.

As confirmed by the tests, the *grayscale pgm* outperforms the *jpg* format for the Potholes and the Transversal and Longitudinal Cracks classifiers, while they both provide similar performances for the Fatigue Cracks classifier. Such results show that the color component does not improve the classifiers' training, introducing misbehaviors during the recognition process and increasing the mean recognition time. In addition, the results underline the ability of each classifier to recognize the pavement distress of interest reaching more than 0.73 of Precision, Recall, Accuracy and F-Measure, except for the Fatigue Cracks classifier which shows the worst performance. Based on these results, we chose the PGM file format for the training sets of the considered classifiers.

To assess the performance of each classifier, we compared their performance with the ones of two state-of-art methods, one for the detection of cracks and the other for the detection of potholes, since they exploit similar approaches. In particular, they focus on the same kind of pavement distresses that we consider in our work, evaluating the results with different metrics (i.e. Precision, Recall, F-Measure, and Accuracy). In addition, since this work is a proof-of-concept of real-time detecting pavement distresses on mobile devices, we have not applied those methods on our images, but just compared the presented results provided by the authors with the ones obtained by the proposed approach by using the values of Precision, Recall, F-Measure and Accuracy. The comparisons are proposed in Tables 3 and 4.

As shown in Table 3, the method proposed in [15] provides a better performance in terms of Precision, Recall and F-Measure, but it also requires more computational resources and time than our approach. Also, it would not be possible to adopt it for a real-time detection of cracks by exploiting a mobile device.

A different scenario is proposed by Table 4, where both methods achieve similar values of Precision and Accuracy. The proposed Potholes classifier, however, outperforms the approach proposed by [19] in terms of F-Measure and Recall.

Table 3
Performance comparison among the proposed cracks classifiers and cracks detection method of [15].

	Proposed fatigue cracks classifier	Proposed longitudinal-transversal cracks classifier	Cracks detection method of [14]
Precision	0.7523	0.7673	0.93
Recall	0.713	0.736	0.992
F-Measure	0.7321	0.7692	0.958

Table 4
Performance comparison among the proposed potholes classifier and pothole detection method of [26].

	Proposed Potholes classifier	Pothole detection method of [26]
Precision	0.8181	0.8
Recall	0.7674	0.735
Accuracy	0.74	0.735
F-Measure	0.7919	0.766

The comparison between the performance of our classifiers and the state-of-art methods shows the capability of our classifiers to provide good performances using the computational resources of a mobile devices. As a matter of fact, the proposed training dataset used for the three considered classifiers, combined with the computational resources provided by a mobile device, allow the ADPR system to reach more than 0.7 of Precision, Recall, Accuracy, and F-Measure and to provide performances that can be compared with more sophisticated state-of-the-art approaches such as the considered ones. It also underlines, however, that further work is required to improve the performance of our cracks detection classifiers making it similar to that of offline methods.

The second battery of test is performed to evaluate the overall performance of the APDR system trained with grayscale PGM samples. For such tests, we created a new set of samples composed of 1000 images with a 70% of samples representing the three pavement distresses of interest. The results, presented in Table 5, underline the ability of our automatic recognition system to correctly identify the three pavement distress categories, providing good results in term of Precision, Recall, Accuracy and F-Measure. However, it is important to notice that the overall performance of the APDR system applying the three cascade classifiers is worse than the one shown in Table 2, and this difference is highlighted by the Precision and Accuracy metrics. Focusing our attention on the Accuracy metric, after the analysis of the results database, it is clear that the system is able to correctly detect nearly the whole set of pavement potholes and longitudinal-transversal cracks used in the testing set, but it mainly fails in the detection of the fatigue cracks. In addition, during the testing of the whole solution, the system detects as fatigue cracks the negative samples that represent a grid pavement. This is due to the performance of the Fatigue

Table 2
Performance of the three developed classifiers in terms of Precision, Recall and F-Measure.

	Fatigue cracks classifier		Potholes classifier		Transversal & longitudinal cracks classifier	
File type	.jpg	.pgm	.jpg	.pgm	.jpg	.pgm
Precision	0.7181	0.72251	0.74729	0.8181	0.7508	0.7673
Accuracy	0.696	0.702	0.711	0.74	0.702	0.736
Recall	0.7364	0.74864	0.73535	0.7674	0.7696	0.7791
F-Measure	0.7271	0.73535	0.74127	0.7919	0.7601	0.7692

Table 5
Overall performance of the APDR system.

	Proposed APDR system
Precision	0.7535
Recall	0.7695
Accuracy	0.728
F-Measure	0.7614

Cracks classifier, which affects the performance of the whole APDR system, generating an accuracy error of 0.272.

In Fig. 8, we provide an example of false detection for each pavement distress, highlighting the drawback of the current approach.

In particular, in Fig. 8.a the algorithm detects a typical spot dedicated to the planting of trees in Rome as a pothole. In Fig. 8.b, the proposed approach identifies a thin line created by the difference in color between wet and dry asphalt as a longitudinal crack. Finally, Fig. 8.c shows the false detection of a fatigue crack when analyzing a framed pavement composed of irregular porphyry stones.

The common factor among these false detections is the inability of the proposed approach to properly analyze and recognize the jagged outlines typical of pavement distresses. To overcome such issues and improve the system's performance, the training dataset of each classifier has to be improved by adding a specific set of negative samples for each pavement distress and by considering some pre-processing techniques, such as the Canny edge detection, to create three enhanced LBP cascade classifiers. Finally, the LIBSVM library will be investigated to verify if it is possible to improve the current ADPR system by replacing the proposed LBP classifiers with a SVM classifier created through LIBSVM.

Finally, the last battery of test is the analysis of the ADPR system behavior through the exploitation of the mobile devices in Table 1. The tests were performed on the same testing dataset used to evaluate the Precision, Recall, Accuracy, and F-Measure reported in Table 5. The obtained results confirm the performance depicted in Table 5, underlining that the hardware and software characteristics do not affect the recognition process in terms of Precision, Recall and F-Measure. However, the RAM and the CPU of the devices affect the detection time required for the recognition of a distress in a frame. In particular, the device 1 (i.e. the oldest one) is able to provide a correct detection in 1 s, while devices 2 and 3, which can exploit a powerful CPU and more RAM than device 1, provide a reliable result in 0.48 s and 0.327 s, respectively.

6. Conclusion and future works

The analysis of civil infrastructures allows specialized teams of structural engineers and inspectors to provide detailed reports on the physical and functional condition of civil infrastructures (i.e. roads, bridges, etc.) giving an indication of their serviceability. The most common technique to perform this task is the manual visual inspection, a time-consuming and subjective technique influenced by the experience of the team. To aid inspectors in their analysis, we have proposed a new automatic pavement distress recognition system able to classify potholes, fatigue cracks and longitudinal and transversal cracks. The APDR system is based on the common CV techniques provided by OpenCV and it exploits three LBP cascade classifiers, one for each pavement distress, trained with sets of sample composed of *grayscale pgm* images. The system is then embedded in a mobile application for Android-based devices and managed through the Android NDK, which allows us to improve the memory performance of the application through the low level programming approach in C++. Finally, the application allows inspectors to perform a real-time classification of distresses relying solely on the computation resources of the mobile device. The classification results (i.e. the name of the distresses, their numbers, locations, images and the acquisition of the dis-

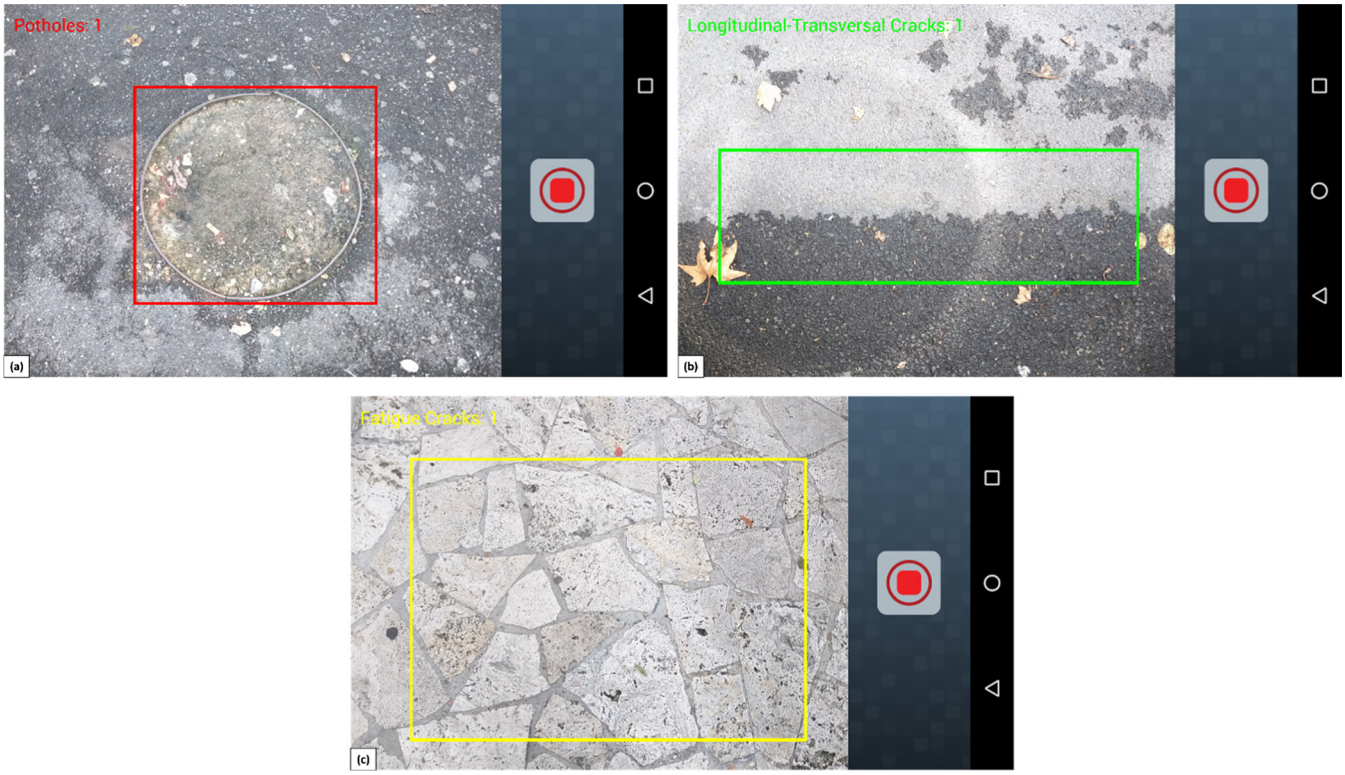

Fig. 8. Examples of false detection of (a) Potholes; (b) longitudinal-transversal crack; (c) fatigue cracks.

tress) are then saved in the local memory of the device, allowing inspectors to create automatic reports of their inspections.

The batteries of tests run to assess the performances of each classifier and of the whole system underline the ability of our application to recognize the pavement distresses of interest reaching more than 0.7 of Precision, Recall and F-Measure. In addition, the tests performed on three different mobile devices show the portability of our application on both older versions of Android and hardware components, therefore providing a responsive application for the newest devices. This application promises to improve the on-site work of inspectors by decreasing the time required to perform inspections while ensuring, at the same time, a higher level of accuracy. However, the proposed system has some limitations: it is not able to classify all the existing type of pavement distresses, such as patches, and to extract the crack curves or pothole shapes.

Futures works will focus on the improvement of the existing classifiers by applying more sophisticated pre-processing steps (e.g. edge detection). Such technique would allow to highlight the shapes and curves of pothole and crack damages, respectively, to extract them, once the ADPR system has identified where the distress is located in the analyzed frame, and to save them as picture in the local memory of the device with the report file. In addition, we will add a new LBP classifier specialized in the detection of patches by defining a new training set composed of positive samples related to this type of distress. Finally, we will focus on the addition of new features, such as the ability to estimate the size of a detected distress and to assess its severity level by comparing the computed size with the thresholds defined in the manuals for visual inspections.

References

[1] U.S. Department of Transportation, Traffic Safety Facts – Crash Stats, June 2015. <http://www-nrd.nhtsa.dot.gov/Pubs/812160.pdf>. Last access 05/11/2016.

[2] European Commission, Road Safety in the European Union, March 2015. <http://ec.europa.eu/transport/road_safety/pdf/vademecum_2015.pdf>. Last access 11/05/2016.

[3] C. Koch, K. Georgieva, V. Kasireddy, B. Akinci, P. Fieguth, A review on computer vision based defect detection and condition assessment of concrete and asphalt civil infrastructure, Adv. Eng. Inform. 29 (2) (2015) 196–210.

[4] Y.O. Ouma, J. Opudo, S. Nyambenya, Comparison of fuzzy AHP and fuzzy TOPSIS for road pavement maintenance prioritization: methodological exposition and case study, Adv. Civil Eng. (2015).

[5] M. Frangopol, K.-Y. Lin, C. Estes, Life-cycle cost design of deteriorating structures, J. Struct. Eng. 123 (10) (1997) 1390–1401.

[6] C.P. Papageorgiou, M. Oren, T. Poggio, A general framework for object detection, in: Sixth International Conference on Computer Vision, Bombay, 1998, pp. 555–562.

[7] Android. <http://developer.android.com/sdk/index.html>. Last access 11/05/2016.

[8] OpenCV. <http://opencv.org/>. Last access 11/05/2016.

[9] S. Liao, X. Zhu, Z. Lei, L. Zhang, S.Z. Li, Learning multi-scale block local binary patterns for face recognition, International Conference on Biometrics (ICB) (2007) 828–837.

[10] National Institute for Phys Planning & Constr Res, Catalog of Road Defects (CORD). <https://trid.trb.org/view.aspx?id=279060>. Last access: 11/05/2016.

[11] J.S. Miller, W.Y. Bellinger. Distress identification manual for the long-term pavement performance program, FHWA-HRT-13-092, 2014.

[12] G. Xu, J. Ma, F. Liu, X. Niu, Automatic recognition of pavement surface crack based on BP neural network, in: International Conference on Computer and Electrical Engineering, ICCEE 2008. Phuket, 2008, pp. 19–22.

[13] J. Bray, B. Verma, Xue Li, W. He, A neural network based technique for automatic classification of road cracks, in: The 2006 IEEE International Joint Conference on Neural Network Proceedings, Vancouver, BC, 2006, pp. 907–912.

[14] N. Li, X. Hou, X. Yang, Y. Dong, Automation recognition of pavement surface distress based on support vector machine, in: Second International Conference on Intelligent Networks and Intelligent Systems, ICINIS '09. Tianjin, 2009, pp. 346–349.

[15] H. Oliveira, J.J. Caeiro, P.L. Correia, Improved road crack detection based on one-class Parzen density estimation and entropy reduction, in: IEEE International Conference on Image Processing, Hong Kong, 2010, pp. 2201–2204.

[16] S. Awate, R. Whitaker, Unsupervised, information-theoretic, adaptative image filtering for image restoration, IEEE Trans. Pattern Anal. Mach. Intell. 28 (2006) 364–376, Mar..

[17] H. Oliveira, J. Caeiro, P.L. Correia, Accelerated unsupervised filtering for the smoothing of road pavement surface imagery, in: 22nd European Signal Processing Conference (EUSIPCO), Lisbon, 2014, pp. 2465–2469.

[18] R. Medina, J. Llamas, E. Zalama, J.G.G. Bermejo, Enhanced automatic detection of road surface cracks by combining 2D/3D image processing techniques, in: IEEE International Conference on Image Processing (ICIP), Paris, 2014, pp. 778–782.

[19] Y. Freund, R.E. Schapire, A decision-theoretic generalization of on-line learning and an application to boosting, J. Comput. Syst. Sci. 55 (1) (1997) 119–139, August.

[20] L. Wu, S. Mokhtari, A. Nazef, B. Nam, H. Yun, Improvement of crack-detection accuracy using a novel crack defragmentation technique in image-based road assessment, J. Comput. Civil Eng. 30 (1) (2014).

[21] M. Avila, S. Begot, F. Duculty, T.S. Nguyen, 2D image based road pavement crack detection by calculating minimal paths and dynamic programming, in: IEEE International Conference on Image Processing (ICIP), Paris, 2014, pp. 783–787.

[22] H. Oliveira, P.L. Correia, CrackIT – an image processing toolbox for crack detection and characterization, in: IEEE International Conference on Image Processing (ICIP), Paris, 2014, pp. 798–802.

[23] G. Wu, X. Sun, L. Zhou, H. Zhang, J. Pu, Research on crack detection algorithm of asphalt pavement, in: IEEE International Conference on Information and Automation, Lijiang, 2015, pp. 647–652.

[24] R. Kapela, A. Turkot, A. Rybarczyk, A. Pożarycki, P. Rydzewski, M. Wyczałek, A. Błoch, Asphalt surfaced pavement cracks detection based on histograms of oriented gradients, in: 22nd International Conference on Mixed Design of Integrated Circuits & Systems (MIXDES), Torun, 2015, pp. 579–584.

[25] V. Baltazart, J.-M. Moliard, R. Amhaz, L.-M. Cottineau, A. Wright, D. Wright, M. Jethwa, Automatic Crack Detection on Pavement Images for Monitoring Road Surface Conditions—Some Results From the Collaborative fp7 Trimm Project, Springer Netherlands, Dordrecht, 2016, pp. 719–724.

[26] S.-K. Ryu, T. Kim, Y-R Kim, Image-based pothole detection system for ITS service and road management system, Math. Problems Eng. (2015).

[27] C. Koch, I. Brilakis, Pothole detection in asphalt pavement images, Adv. Eng. Inform. 25 (3) (2011) 507–515.

[28] C. Koch, G.M. Jog, I. Brilakis, Automated pothole distress assessment using asphalt pavement video data, J. Comput. Civil Eng. 27 (4) (2014) 370–378.

[29] G.M. Jog, C. Koch, M. Golparvar-Fard, I. Brilakis, Pothole properties measurement through visual 2D recognition and 3D reconstruction, in: Proceedings of the ASCE International Conference on Computing in Civil Engineering, 2012.

[30] Z. Zhang, X. Ai, C.K. Chan, N. Dahnoun, An efficient algorithm for pothole detection using stereo vision, in: IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2014, pp. 564–568, May.

[31] S. Li, C. Yuan, D. Liu, H. Cai, Integrated processing of image and gpr data for automated pothole detection, J. Comput. Civil Eng. (2016).

[32] Y. Tai, C. Chan, J.Y. Hsu, Automatic road anomaly detection using smart mobile device, in: Conference on Technologies and Applications of Artificial Intelligence, Hsinchu, Taiwan, 2010.

[33] A. Mednis, G. Strazdins, R. Zviedris, G. Kanonirs, L. Selavo, Real time pothole detection using Android smartphones with accelerometers, in: International Conference on Distributed Computing in Sensor Systems and Workshops (DCOSS), Barcelona, 2011, pp. 1–6.

[34] R. Madli, S. Hebbar, P. Pattar, V. Golla, Automatic detection and notification of potholes and humps on roads to aid drivers, IEEE Sens. J. 15 (8) (Aug. 2015) 4313–4318.

[35] S. Cafiso, A. Di Graziano, S. Battiato, Evaluation of pavement surface distress using digital image collection and analysis, in: Seventh International Congress on Advance in Civil Engineering, Istanbul, Turkey, 2006.

[36] X. Yao, M. Yao, B. Xu, Automated detection and identification of area-based distress in concrete pavements, in: Seventh Int. Conf. Manag. Pavement Assets, 2008.

[37] S.C. Radopoulou, I. Brilakis. Improving patch distress detection using vision tracking on video data, in: 21st International Workshop on Intelligent Computing in Engineering, 2014.

[38] S.C. Radopoulou, I. Brilakis, Patch detection for pavement assessment, Autom. Construct. 53 (2015) 95–104, May.

[39] G.M. Hadjidemetriou, S.E. Christodoulou, P.A. Vela, Automated detection of pavement patches utilizing support vector machine classification, in: 18th Mediterranean Electrotechnical Conference (MELECON), Lemesos, Cyprus, 2016, pp. 1–5.

[40] A. Mednis, G. Strazdins, R. Zviedris, G. Kanonirs, L. Selavo, Real time pothole detection using Android smartphones with accelerometers, in: 2011 International Conference on Distributed Computing in Sensor Systems and Workshops (DCOSS), Barcelona, 2011, pp. 1–6.

[41] V. Douangphachanh, H. Oneyama, Using smartphones to estimate road pavement condition, in: International Symposium for Next Generation Infrastructure. Wollongong, Australia, October 1-4, 2013.

[42] Y. Hu, C.-X. Zhao, H.-N. Wang, Automatic pavement crack detection using texture and shape descriptors, IETE Tech. Rev. 27 (2010) 398–405.

[43] A. Banharnsakun, Hybrid abc-ann for pavement surface distress detection and classification, Int. J. Machine Learning Cybernet. (2015) 1–12.

[44] E. Salari, X. Yu, Pavement distress detection and classification using a Genetic Algorithm, in: 2011 IEEE Applied Imagery Pattern Recognition Workshop (AIPR), Washington, DC, 2011, pp. 1–5.
[45] F. Seraj, B.J. van der Zwaag, A. Dilo, T. Luarasi, P. Havinga, Roads: A Road Pavement Monitoring System for Anomaly Detection Using Smart Phones, Springer International Publishing, 2016, pp. 128–146.
[46] C.W. Hsu, C.C. Chang, C.J. Lin, A Practical Guide to Support Vector Classication, Department of Computer Science, Taipei 106, Taiwan, 2003.
[47] K. Pulli, W-C. Chen, N. Gelfand, R. Grzeszczuk, M. Tico, R. Vedantham, X. Wang, Y. Xiong, Mobile visual computing, in: International Symposium on Ubiquitous Virtual Reality, Gwangjux, 2009, pp. 3–6.
[48] E. Gamma, R. Helm, R. Johnson, J. Vlissides, Design Patterns: Elements of Reusable Object-Oriented Software, ISBN-10:0201633612, 1994.
[49] C. Larman, Applying UML and Patterns – An Introduction to Object-Oriented Analysis and Design and Iterative Development, Prentice Hall, New Jersey, ISBN: 0-13-148906-2.
[50] P. Rogers, R. Brien, Kantar Worldpanel ComTech Smartphone OS Barometer 03-12-14, Jan. 2014, <http://uk.kantar.com/>.
[51] A. Tedeschi, A. Liguori, F. Benedetto, Information security and threats in mobile appliances, Recent Patents Comput. Sci. 7 (1) (2014) 3–11, http://dx.doi.org/10.2174/2213275907666140610200010.
[52] Android NDK. <http://developer.android.com/tools/sdk/ndk/index.html>. Last access 11/05/2016.
[53] R. Lienhart, J. Maydt. An extended set of Haar-like features for rapid object detection, in: Proceedings, 2002 International Conference on Image Processing, vol. 1, 2002, pp. I-900–I-903.
[54] M. Gavilán, D. Balcones, M.A. Sotelo, D.F. Llorca, O. Marcos, C. Fernández, I. García, R. Quintero, Surface classification for road distress detection system enhancement, in: Computer Aided Systems Theory – EUROCAST 2011: 13th International Conference, Springer, Berlin, Heidelberg, 2012.
[55] F. Benedetto, A. Tedeschi, Big data sentiment analysis for brand monitoring in social media streams by cloud computing, in: Witold Pedrycz, Shyi-Ming Chen (Eds.), Sentiment Analysis and Ontology Engineering: An Environment of Computational Intelligence, Springer-Verlag, 2016.
[56] OpenCV, Cascade Classifier Training. <http://docs.opencv.org/2.4/doc/user_guide/ug_traincascade.html>. Last access 11/05/2016.
[57] Android, Getting Started with the NDK. <https://developer.android.com/ndk/guides/index.html>. Last access 21/07/201.
[58] Google, Google APIs for Android. <https://developers.google.com/android/reference/com/google/android/gms/location/package-summary>. Last access 11/05/2016.
[59] Android, Making Your App Location-Aware. <https://developer.android.com/training/location/index.html>. Last access 21/07/2016.
[60] Android, Saving Files. <https://developer.android.com/training/basics/data-storage/files.html>. Last access 21/07/2016.
[61] Android. AsyncTask. <https://developer.android.com/reference/android/os/AsyncTask.html>. Last access 21/07/2016.