Hindawi Publishing Corporation
Advances in Civil Engineering
Volume 2014, Article ID 483729, 14 pages
http://dx.doi.org/10.1155/2014/483729

*Research Article*

# Mobile Imaging and Computing for Intelligent Structural Damage Inspection

**ZhiQiang Chen[1] and Jianfei Chen[2]**

[1] *Department of Civil and Mechanical Engineering, University of Missouri-Kansas City, 5100 Rockhill Street, Kansas City, MO 64110, USA*
[2] *Department of Computer Science Electrical Engineering, University of Missouri-Kansas City, 5100 Rockhill Street, Kansas City, MO 64110, USA*

Correspondence should be addressed to ZhiQiang Chen; chenzhiq@umkc.edu

Received 7 February 2014; Revised 10 July 2014; Accepted 27 August 2014; Published 1 October 2014

Academic Editor: Ren-Jye Dzeng



Optical imaging is a commonly used technique in civil engineering for aiding the archival of damage scenes and more recently for image analysis-based damage quantification. However, the limitations are evident when applying optical imaging in the field. The most significant one is the lacking of computing and processing capability in the real time. The advancement of mobile imaging and computing technologies provides a promising opportunity to change this norm. This paper first provides a timely introduction of the state-of-the-art mobile imaging and computing technologies for the purpose of engineering application development. Further we propose a mobile imaging and computing (MIC) framework for conducting intelligent condition assessment for constructed objects, which features in situ imaging and real-time damage analysis. This framework synthesizes advanced mobile technologies with three innovative features: (i) context-enabled image collection, (ii) interactive image preprocessing, and (iii) real-time image analysis and analytics. Through performance evaluation and field experiments, this paper demonstrates the feasibility and efficiency of the proposed framework.

## 1. Introduction

*1.1. Background and Rationale.* Sensing-based inspection technologies are recognized as critical components for solutions to quantitative and risk-informed condition assessment for buildings, bridges, and other civil infrastructure systems [1, 2]. Among numerous sensing technologies, visual inspection is commonly used in engineering practice for the purpose of archiving damage scenes and patterns. For example, visual inspection is considered the predominant approach to condition assessment for the majority of bridge inventories in the United States [3]. In these practices, visual inspection is often companied by the use of digital camera for the purpose of digital archival through photographing (viz., optical imaging). The fundamental basis of optical imaging is that through measuring photonic energy emanating from distant objects that are degraded or damaged, disturbed spatial or spectral patterns on the surface of the objects are recorded in a digital format (i.e., two-dimensional images) [4]. In practice, a large percentage of damage patterns, which are visible on the surfaces of structural or geotechnical members (e.g., cracking, spalling, deformation, or collapse-induced debris), can be captured using a commercial digital camera.

In recent years, many research endeavors attempt to empower visual inspection and optical imaging by quantitative image analysis; hence, automatic or semiautomatic damage detection may be achieved [5–9]. Among these efforts, digital images are captured using digital cameras in the field first; then image processing or pattern recognition methods are employed to detect or classify visible damage, respectively. Research findings in this direction have ultimately driven the development of vehicle-mounted imaging and real-time processing for highway or road surface damage inspection [10, 11].

For conducting image-based condition assessment within a *complex* or/and *hard-access* built environment, however, limited technology solutions exist. Figure 1 shows two typical

(a) (b)

Figure 1: Bridge inspection in practice (photos taken in 2007 courtesy of Virginia Department of Transportation, VDOT; creative commons rights are reserved to VDOT): (a) steel bridge joint inspection with complex geometric configurations and (b) hard-access inspection of the side of the concrete bridge deck.

field operation scenes of bridge inspection, wherein the region of inspection interest is considerably hard to access and subject to potential operation risk. Robotic imaging and inspection have been attempted (e.g., climbing robots) [12–14]; particularly, the use of unmanned aerial vehicles (UAVs) for civil infrastructure inspection is on the horizon [15–17]. However, all of them are experimental tools and are heavily customized to a certain built environment. In the current practice, human inspection still dominates the practice for general inspection in a complex or/and hard-access built environment.

In light of the dominance of human inspection, one major limitation to date is the lack of real-time imaging, computing, and analytics capability that can be interactively controlled and utilized by the inspection engineers in the field. The state-of-the-practice visual inspection activities for those complex or hard-access areas usually involve a team of engineers who visually inspect structures and use their discretion to capture images in the field. Towards quantitative assessment after the fieldwork, it is expected that these field images can be digitally processed and damage can be extracted using a suite of imaging processing packages. However, one should realize that first most civil engineers are not equipped with sufficient image processing and analysis knowledge and they mostly expect a rapid analysis procedure. Second, once the images are taken away from the field situation, much of the contextual knowledge about the images, such as the damage type of the object and the complexity of the damage scene, is prone to be lost subject to the engineer's memory. On the other hand, such contextual information, if utilized, could greatly alleviate the difficulty in processing of structural damage images that usually come with various complex scenes.

The aforementioned limitation can be readily resolved by employing the modern mobile technologies. Mobile technologies are closely associated with the advancement of mobile devices, which can be traced down to the movement of small-form-factor computers in the late last 90s (e.g., various brands of Palms or pocket PCs) [18]. Today's mobile devices tightly integrate high-resolution cameras, multicore computing units, flexible communication and data networks, interactive human-device interfaces, geographical position service, and other auxiliary sensors. Therefore, these devices are commonly regarded as "smart" and are represented by smartphones and smartphone-alike tablet devices (the latter usually lack cellular network support). These smart mobile devices have become truly ubiquitous. There are over one billion mobile devices around the world at the end of 2012 [19], and many of them have embedded cameras. Compared to traditional cameras, the advantage of mobile devices may include easy to carry, easy to share, and easy to modify due to the small form-factor hardware design, the communication capability, and the computing capability of mobile devices. As such, today's mobile devices as everyday imaging utility tend to be much more frequently used than traditional cameras.

We therefore envisage that with the ubiquitous mobile devices (e.g., smartphones) and their mobile imaging and computing (MIC) capability, a new norm for structural damage inspection and condition assessment featuring real-time image collection, computing, and damage analytics is on the horizon. Such a framework, if implemented, may significantly alleviate the cumbersome labor involved in practical visual-based civil infrastructure inspection activities for these complex or hard-access structures (as shown in Figure 1).

*1.2. Contribution.* The key platform considered in this paper is smart devices, including smartphones and mobile tablets, which have appeared as a commonplace device and venue for running smart applications used by the general public. Although a few industrial applications have appeared using smartphones for imaging of structural damage, a comprehensive investigation is not found in the literature. The major contribution of this paper is development of a general mobile imaging and computing framework for civil infrastructure damage inspection that tightly integrated the prominent features of smart devices: in situ imaging, computing, and interactive human-device interfaces for both data logging and visual analytics.

*1.3. Organization.* In the following, first, the related mobile imaging and computing technologies are reviewed. Then

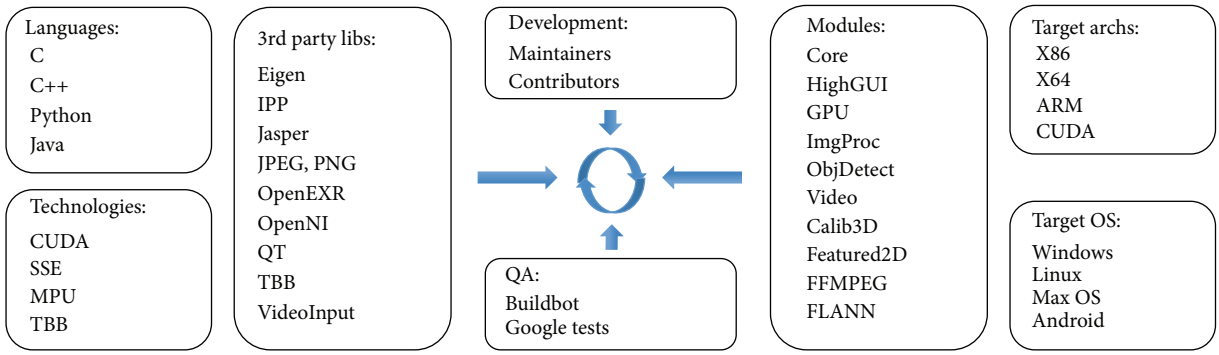



Figure 2: OpenCV architecture and library components.

software environments and mobile imaging and computing libraries that are crucial for engineering development are introduced. These two sections aim to provide a timely and critical review of the state-of-the-art mobile imaging and computing technologies to readers and developers in the civil engineering communities. Subsequently, the framework for intelligent mobile structural inspection is proposed and described. Then this paper focuses on performance evaluation of the proposed MIC framework using a developed prototype application. A summary of conclusions of this paper is presented in the end.

## 2. MIC: Development Environment and Engineering Applications

*2.1. Overview of Mobile Development.* The advances of modern mobile imaging and computing have been tightly relied on two core technological advances that occurred during the last 20 years: mobile imaging sensors and mobile computing units. Enabled by the high-resolution imaging and high-performance computing power that are built into smart mobile devices, it is possible to use these mobile devices as alternative to professional equipment like camera and workstation. Prosperous research endeavors have also been witnessed in an emerging field, namely, mobile visual computing [20], which is closely related to the novel endeavor in this paper.

To develop mobile visual computing applications, the first primary task is to select mobile operating systems (OS), which provide the vital connections between imaging, computing, user interface, and integration of auxiliary sensor data. Two of the mainstream mobile operating systems are Android and iOS [21]. For most smart apps to date, therefore, usually at least two versions are released—one for Android platforms and the other for iOS platforms. In this paper, the Android system is selected as the primary development mobile OS.

Android is open source and Google releases the code under the Apache License. It includes a set of C/C++ libraries used by various components of the Android system. It also includes a set of core libraries that provides most of the functionality using the Java programming language. Every Android application runs in its own process, with its own instance [22]. For application developers, Google Inc. has provided Android software development kit (SDK). The SDK includes a comprehensive set of development tools, including a debugger, software libraries, a handset emulator, documentation, sample code, and tutorials.

*2.2. OpenCV and Mobile Visual Computing.* Most applications on Android platform are written in Java programming language, which can satisfy most mobile and interactive applications. For serious visual computing tasks, however, efficient image and vision libraries are entailed. Modern imaging and vision libraries have been developed for solving various visual computing problems. These libraries include VXL, VLFeat, RAVL, OpenSURF, CImg, ImageJ, and BoofCV. These libraries mainly aims at solving specific problems like feature extraction, classification, or simple image I/O operation. MATLAB-based computer vision toolbox is another tool that can be used for imaging computation, visualization, and programming.

Among these vision libraries, OpenCV is a popular one [23, 24], which provides both computer vision and machine learning programming functions mainly aimed at real-time image processing and understanding. Compared to other libraries mentioned earlier, OpenCV provides the most comprehensive set of optimized algorithms. It has C++, C, Python, and Java interfaces and supports multiple platforms like Windows, Linux, Mac OS, and now Android systems. Figure 2 shows the OpenCV architecture design. Compared to other vision libraries, OpenCV is optimized for mobile platform in real-time processing.

Among the components in the architecture, HighGUI allows users to interact with the operating system, the file system, and the imaging hardware such as cameras and touch screens. It allows a simple way to query a camera and retrieve the latest image from the camera. The file system focuses on loading and saving images. HighGUI uses an abstract method to load and save images and videos without concerning the particular device. Hence, reading and saving images will only be one line of code in an Android-enabled device. Besides HighGUI, OpenCV library provides over 2500 optimized algorithms that operate on images to accomplish specific tasks, which include basic methods for image enhancement,

modification, and transformation and advanced methods for image matching, segmentation, motion tracking, and camera calibration.

*2.3. Engineering Applications.* A plethora of mobile imaging and computing applications has been developed in many areas, such as robotics, automation, medical informatics, and engineering [14, 25–33]. One interesting application is recognition of 2-dimensional quick response (QR) codes, a ubiquitous coding and marking utility to date. In these efforts, mobile devices capture the images of QR patterns and identify in real time [32, 33]. Another noteworthy work by Hull et al. argues that mobile imaging and computing are the most desirable architecture for a series of mobile image recognition problems compared to a mobile-server computing architecture. They further reported a mobile imaging and computing experiment for automatic recognition of articles, photos, and advertisements in newspapers using a mid-range smartphone, in which the average run time for an image is 1.2 secs and the recognition error rate is less than 1% [34]. This paper indicates with concrete evidence for the great potential of mobile imaging and computing when today's smart devices are used.

In civil engineering, mobile computing has been recognized as an important utility for realizing efficient information sharing, design, and construction collaboration. Nagayama et al. built a video-based application with simultaneous acceleration recordings for road condition assessment [35]. By using accelerometer, GPS, and magnetometer sensors on smartphones, a nonintrusive method was introduced by Bhoraskar et al. to detect city traffic and road condition [36]. Combining accelerometer, gyroscope, magnetometer, GPS, and camera on mobile devices, researchers also developed a system to classify and recognize driving styles [37]. Behzadan et al. exploited the location service of mobile devices for realizing context-aware delivery in construction sites [38]. A wearable system for onsite building information modeling and sharing was recently developed through integrating mobile phones and displaying devices for visual analytics [39].

Among these applications, the notation of MIC applications based on the state-of-the-art mobile technologies has four core features: (i) portable generation of digitalized images via mobile imaging sensors, (ii) contextual integration with human-device interface (e.g., touch screens) and other auxiliary sensor data, (iii) real-time or near-real-time computing and analysis using mobile imaging and contextual data, and (iv) deliverable products through mobile communication networks. These features, either all or part of them, are manifested in the previously reviewed mobile applications. It is noted that, however, MIC applications in civil engineering reviewed before are in general lacked behind in terms of applying the state-of-the-art technologies. Among these applications, image-centric in situ computing in the mobile ends (e.g., image analysis-based recognition or information extraction) has not been emphasized due to the limited capability of earlier-generation mobile devices or their specific purposes of applications.

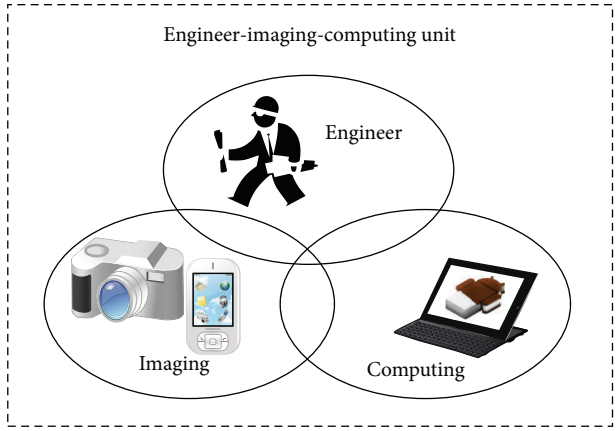



Figure 3: The structure of an engineer, imaging, and computing unit in the field.

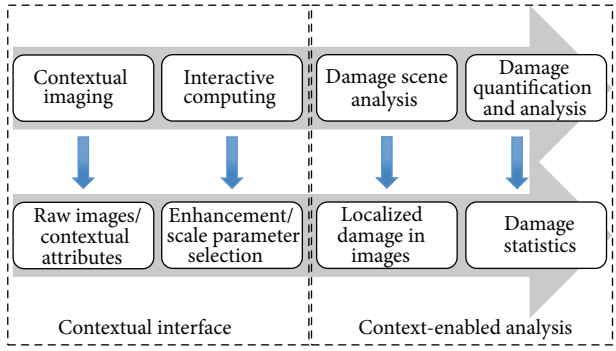



Figure 4: MIC framework for structural inspection: the operation flow (above) and the correspondingly generated dataflow (bottom).

## 3. Intelligent MIC Methodology for Structural Inspection

The research problem in this paper is to conduct intelligent structural damage inspection in a complex or hard-access built environment through exploiting the power of mobile imaging and computing technologies. In the following, an *intelligent* mobile imaging and computing framework for structural inspection is proposed.

A novel concept is proposed to build this framework, termed *engineer-imaging-computing (EIC) unit*, which consists of an engineer with his or her equipment in the work zone. As illustrated in Figure 3, each EIC unit has two basic elements, which are the engineer and mobile devices. It assumes that an EIC unit is equipped with at least one Android-based mobile device. Besides using mobile devices as the basic imaging equipment, digital cameras can be used. A basic requirement for regular digital cameras is that they can be wirelessly tethered to the mobile computing device.

Build upon an EIC unit, Figure 4 illustrates the intelligent mobile imaging and computing framework, which emphasizes its operation and data flow. The term of "intelligent" has two meanings relevant to the framework: (1) the potential human-centered interaction between the human and

Table 1: Contextual information examples: engineering damage related empirical and embedded sensor-generated information.

| Category | Contextual attributes | Values (examples) |
|---|---|---|
| Engineering damage related information | Structure Type | Building, bridges, and pavement |
| | Structural element | Column, pier, beam/girder, wall, foundation, |
| | Structural material | Concrete, wood, steel, masonry, asphalt, soil, and sand |
| | Damage pattern | Line type: crack, void, and delamination<br>Texture type: pothole, spalling, corrosion, and debris |
| | Scene complexity | Humidity, vegetation, oil-spill, and dirt-contaminated |
| Embedded sensor-generated information | Geolocation | Latitudes and longitudes |
| | Ambient light | Ambient light value |
| | Acceleration | Motion and gravity acceleration |

the devices that is based on the human's engineering knowledge and (2) the interactive and context-enabled imaging and computing within an EIC unit.

Correspondingly, the operation and dataflow are divided into two components and four functions: (1) contextual interface including the functions of contextual imaging and interactive computing and (2) context-enabled analysis, including the functions of damage scene analysis, quantification, and analytics. Details in fulfilling these two primary function components are presented as follows.

*3.1. Contextual Interface.* The imaging activity in the field includes image and video capturing. This is considered the basic sensing activity in the proposed framework. Contextual interface implies that besides the basic imaging function, an information augmentation process by recording or computing key contextual information is associated with the imaging activity. The contextual information herein has three categories: (1) engineering knowledge that is related to damage scenes, (2) embedded sensor-generated data accompanying the imaging activity, and (3) interactively obtained image processing parameters key to subsequent advanced analysis. This contextual information along with raw or processed imagery data forms complex imagery metadata.

In Table 1, engineering knowledge related and embedded sensor-generated contextual attributes and their values are listed. In the current design, this contextual information in terms of attributes values is implemented and integrated with the imaging interface. For embedded sensor-generated information in Table 1, no user input is needed.

For engineering knowledge-related information, it is obvious that user input is a straightforward approach to obtaining their values. It is noted that artificial intelligence-enabled automatic recognition is a possible solution that may automatically recognize, for example, the material or damage pattern types given the imagery information. However, we recognize that a fully automatic solution is a difficult task if it is charged to function at any imaging condition for any scene of complexity, considering the objective of recognizing structural element, material types, and damage types at the same time. For simple concrete surface and crack damage recognition, this may be feasible. One may refer to the first author's earlier work, where a feature extraction and neural network-based approach was successfully implemented [40]. On the other hand, if this accurate contextual engineering information is provided for complex damage scenes, it will dramatically improve the robustness and efficiency of the subsequent damage analysis and quantification function.

Another important category of contextual information is image analysis parameters. The interactive computing function implies that human-based gesture and input (e.g., typing or voice-based input) can be used to provide key contextual parameters for advanced image analysis parameters. In the proposed framework, these interactive parameters include important values that are important to the preprocessing steps of the following automatic image-based damage analysis: (1) bounding box parameters for initial damage boundaries and (2) image processing parameters for line- or region-damage type detection. As will be implemented in the next section, this combined approach, integrating engineering knowledge-based context logging and interactive bounding box/processing parameter selection, solves the generally hard visual understanding problem (e.g., recognition and localization in natural images [41]).

*3.2. Context-Enable Damage Analysis.* Given the captured imagery data and the associated contextual values, context-enabled damage analysis is implemented. First, several segmentation and edge or contour detection methods are implemented in the framework for damage scene analysis, which usually results in location of certain damage (e.g., crack or spalling). Again as indicated earlier, no damage recognition is needed herein since it is provided by the contextual input at the imaging interface (Table 1). Therefore, based on this damage type information, either a regional damage detection (e.g., for spalling, pothole, etc.) method or linear damage types (e.g., for cracks) will be initiated automatically to perform the damage localization task.

When location of damaged patterns is determined, which may be in terms of lines or contours in the image domain,

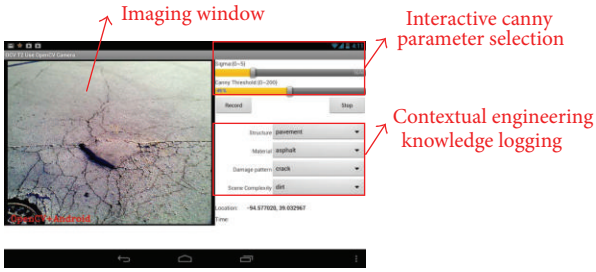



Figure 5: Mobile interface for contextual imaging of structural damage.

geometric parameters summarizing the damage can be easily calculated. However, one should be aware that the imaging spatial domain is only a 2D linear projection of the actual 3D physical object subject to nonlinear imaging distortion. For lens-induced distortion, a camera calibration procedure should be applied before initial imaging [42]. However, we note a special situation that if a planar damage pattern (which may lie in a geometrically complex 3-dimensional object) is parallel to the imaging plane of lens, then the extracted damage patterns in images are approximately only subject to a constant geometric scaling (after distortion correction).

## 4. Prototype Implementation and Demonstration for Crack Analysis

*4.1. Prototype Implementation.* Based on the proposed MIC methodology, a prototype application for intelligent structural inspection is developed. Android-based smart devices and systems are chosen for the development environment. The interface of the front-end mobile application is implemented using Android software development kit (SDK) (version 4.1), and the underlying imaging and computing functions are realized through the OpenCV library (version 2.4). Figure 5 shows the human-device interface. One can see that the interface is designed to realize three basic functionalities for realizing: (1) real-time imaging, (2) image analysis parameter selection, and (3) engineering knowledge-based attributes logging. Besides, geospatial location service is used to save the latitude and longitude coordinates when an image is captured, and acceleration values and ambient lighting values are recorded in the background. This interface design collectively realizes the proposed *contextual interface* illustrated in the framework (Figure 5).

*4.2. Implementation for Mobile Imaging and Computing for Crack Analysis.* Crack detection is one of the most common tasks in structural damage inspection activities. The tradition usually follows a routine, in which, first, images are captured using digital cameras, and then digital processing by means of a robust edge detector is performed after the field inspection. Utilizing the proposed intelligent MIC framework and the developed interface (Figure 5), a full implementation of the MIC framework for structural crack analysis is realized in this paper. The working protocol using the developed prototype application is shown in Figure 6.

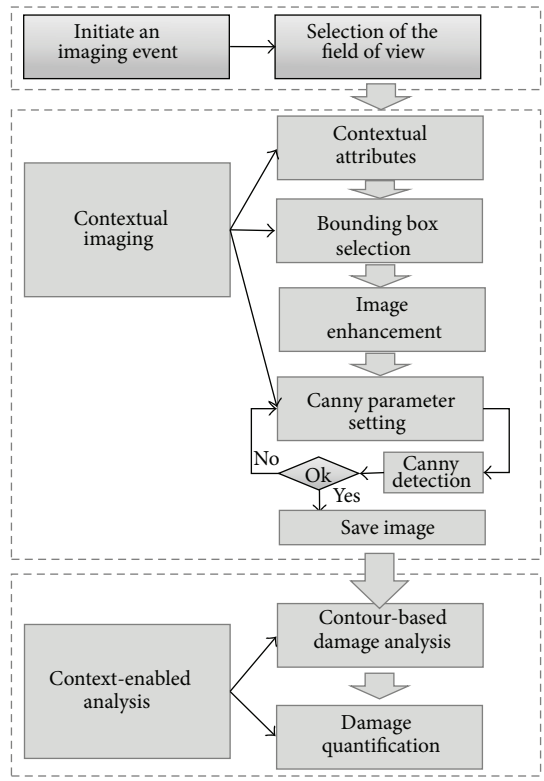



Figure 6: Mobile imaging and computing protocol for a typical crack scene analysis.

In this working protocol, an engineer starts an imaging event by empirically selecting a proper field of view. In the following, technical details regarding how to implement these steps are given below.

*(1) Contextual Engineering Attributes Logging.* Currently this is realized through touchscreen gesture-based dropdown menu, which is implemented using Android' SDK. Nonetheless, we envision that future development can integrate speech-based logging based on natural language processing libraries (e.g., [43]).

*(2) Bounding Box Selection for Damage Localization.* Damage patterns in captured images are usually located in one portion of the images. The bounding box envelopes the damage scene providing an approximate manual segmentation of the damage pattern of interest. Therefore, quality damage detection implies that localization of the overall damage pattern is an important initial step. Such localization usually calls for computationally demanding segmentation algorithms. Advanced contour algorithms can be found in the literature (e.g., deterministic level-set or statistical texture-modeling methods) [44, 45]; however, these methods usually entail considerable parameter tuning or training data preparation and high computational demands. Within the imaging window, this is interactively realized first using Android's SDK to obtain the coordinates of the touch gesture within the real-time imaging window; then the OpenCV functions "*boundingRect*" and "*drawContour*" are used to create a bounding box, respectively.

*(3) Automatic Image Enhancement*. In our experiment, this image enhancement step is an important preliminary processing that improves the latter edge detection and contour analysis. The image enhancement is executed automatically for each input true-color image. The process has three steps. First, the image is converted into a gray-level intensity image then a distortion correction is applied using the existing camera correction matrix if it is available. Second, histogram equalization is used to enhance image contrast, which is a widely used low-level technique for image enhancement. Histogram equalization is a nonparametric nonlinear filtering, which stretches out the pixel intensity range of an image by mapping the original histogram distribution to a more uniform distribution of intensity values [46]. The OpenCV function "*equalizeHist*" is used.

A standard mean-shift filter is applied to further smooth out the image noises and in the meantime to preserve the discontinuity (e.g., edges, lines, and other linear transitions in the image domain). Mean-shift filtering is a widely used common technique for edge-preserving smoothing and image segmentation. A standard mean-shift filter needs one spatial and one range scale parameters, denoted by $\sigma_s$ and $\sigma_r$, respectively. In this paper, constants are used to set the two scale parameters, $(\sigma_s, \sigma_r) = (3, 6)$, which are proved to be a proper setting for preserving edge/line features while moderately smoothing out noise texture for our image database [47]. Correspondingly, the OpenCV function "*meanShiftFiltering*" is used.

With the above setting, the combined image enhancement is implemented without user intervention, which runs in the background automatically for each image prior to advanced image analysis.

*(4)* Interactive Canny Edge Parameter Selection. Common surface damage patterns (e.g., line-like cracks or regional damage with a clear boundary) are generally easy to detect using edge detectors. However, edge detection usually needs an appropriate scale factor for defining the detector's filtering kernel size ($\sigma$). For a particular detector, for example, the Canny detector, a hysteresis threshold ($\alpha$) is further needed [48] (note that the Canny function in OpenCV uses two hysteresis threshold parameters, the upper and the lower one; herein, $\alpha$ denotes that the upper threshold is used, and the lower threshold is obtained as $\alpha/2$). Automatic selection of these parameters in general is a challenging problem. Many advanced or automatic scale parameter selection procedures exist [49, 50]. However, these methods only yield statistically "objective" edge detection results, which are still subjected to visual approval and inferior to expert-eye visual result. In light of these, interactive and manual parameter selection for obtaining the smoothing scale parameter ($\sigma$) and the hysteresis thresholding parameter ($\alpha$) for the Canny detection is implemented in the prototype using the Android SDK functions and the selected parameters are fed to the OpenCV function "*Canny*" [48].

In the implementation, default values for both $\sigma$ and $\alpha$ are used, which are 2.0 and 40%, respectively, as shown in Figure 5. Our field-based experience is that in most cases, the two values suffice the need; therefore, the users may not need to change them for clearly defined cracks in the images. In case of necessity, the user can first choose to modify the smoothing factor ($\sigma$) to a visually suitable detection result. In most cases, the default hysteresis threshold value ($\alpha$) does not need to modify significantly. This setting largely reduces the interactive time cost.

*(5) Contour Analysis and Damage Quantification*. After saving the enhanced image and key contextual information from the previous steps, the subsequent image scene analysis and damage quantification are automatic. Contour analysis can be realized using advanced geometric modeling (e.g., [51, 52]). In the current implementation, a simple contour analysis algorithm is adopted, which computes contours based on any binary imagery [53]. Hence, the OpenCV function "*findContours*" in the OpenCV library is used. In our implementation, the binary output from a Canny detector is used to generate binary inputs. Therefore, the same scale and threshold parameters interactively selected previously are used in the contour analysis. It is noted that when the Canny detection returns 1-pixel wide line artifacts, the contour analysis outputs lines as well. We particularly indicate that in reality if the crack width is small, user can take photos at a closer distance to avoid this situation.

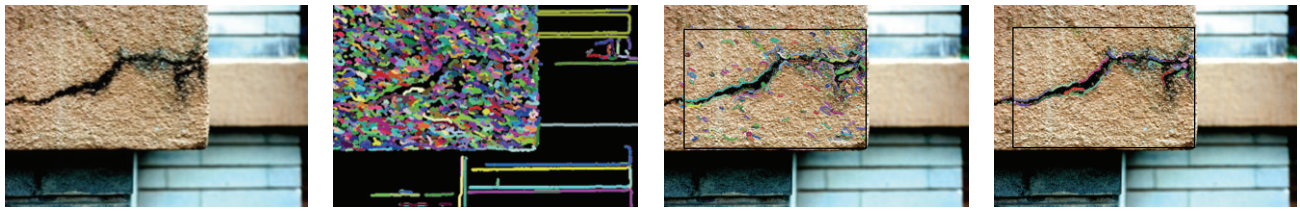
One merit of using this simple contour analysis is that many structural contour-based calculation functions are abundant, which are ready to output important geometric parameters for detected contours, such as area and perimeter. These geometric parameters provide important statistics for summarizing the damage quantitatively.

It is noted that if the camera is positioned parallel to the surface, there will be only a scale factor that accounts for the geometric correction between the image domain and the actual damage scene. Considering other complex situations, we will suggest to use a check board (which should be used for image calibration first) in the imaging field. This check board will be used to extract the projective matrix for the imagery domain, which leads to correct transformation of the images and, hence, accurate damage extraction (such as crack width, [6]). This camera calibration and image correction study is beyond the scope of this paper.

*4.3. Concrete Crack Imagery Database.* To evaluate the performance of this MIC methodology and the application prototype, a large database of concrete crack images is established. The images are gathered from two ways. First, we searched the Internet with certain key words and filters and obtained 140 concrete surface and crack images (which are free to be used by Google copyright terms [54]). Second, the developed prototype application was in two Android-based smart devices to capture road surface crack images, which resulted in an imagery database with 110 images. The two resulting imagery databases are also found in our research website as an open-source project for community-based development [55]. Figures 7(a) and 7(b) show part of the image samples, one in term of an image gallery and the other in terms of searchable images within a Google map,


(a) (b)

Figure 7: Concrete crack imagery database: (a) a database based Internet searching and (b) a database from field-based imaging using the developed application.



(a) (b) (c) (d)

Figure 8: Demonstration of a mobile imaging and computing-based crack damage analysis: (a) original true-color image, (b) contour analysis without a bounding box or optimal Canny parameters ($\sigma = 1.0$; $\alpha = 10$), (c) analysis with a bounding box without optimal Canny parameters, and (d) analysis with a bounding box and optimally selected parameters ($\sigma = 2.0$; $\alpha = 40$).

respectively. It is noted that both databases possess images with complex scenes.

### 4.4. Experimental Results

*4.4.1. Effectiveness of Context-Enabled Visual Analysis.* To demonstrate the effectiveness of context-enabled visual analysis due to the contextual logging and interactive selection of key analysis parameters, Figure 8 illustrates an example wherein an image with a complex scene is selected. In Figure 8(a), one can see that the damage of interest is a crack in the foreground that is discolored and contaminated by humid. Without selecting a bounding box and visually selected Canny parameters, the contour analysis will produce a result as shown in Figure 8(b). When a bounding box is preselected, contour analysis will be focused on the region of interest wherein an elongated crack is found; however, many noisy contours are found in the result (Figure 8(c)). In Figure 8(c) , a relatively optimal parameter setting resulting from the interactive parameter selection using a touched-based smartphone is applied. One can see that visually satisfactory crack detection result is obtained within the bounding box. As such, simple postprocessing by summarizing the geometric shapes of contours will produce much more accurate crack statistics (Figure 8(d)).

*4.4.2. Context-Enabled Visual Analysis and Damage Quantification.* Table 2 reports the results of 10 cases of context-enabled visual analysis and damage quantification results (the first five images are from the Internet search-based database and the latter five are based on the field experiment, and all images are resampled to have a width of 256 with a variable height). Besides the ultimate contour analysis output, we report one intermediate result—Canny edge detection result with optimal interactively selected parameters and a selected preliminary bounding box. Based on the subsequent contour analysis, simple damage quantification is conducted. It is noted that without considering a scale/dimension conversion, the reported crack lengths and areas (Len, Area) in the able are in terms of pixels and squared pixels, respectively.

The images shown in Table 2 manifest scene complexity of different levels. For some images, for example, Images numer 2 or number 4, bounding box may be not necessary due to the less noisy image background and the clearly developed crack feature, or because the captured damage fills the image. For many other images, for example, Images number 6, 8, and 10, a bounding box is necessary to define an initial boundary of the damage. Contour analysis results provide damage mapping in terms of either closed contours or single edges. As mentioned earlier, the contour analysis

Table 2: Context-enabled image analysis and quantification for concrete crack damage.

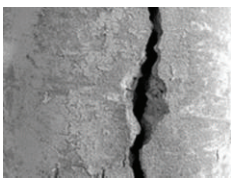
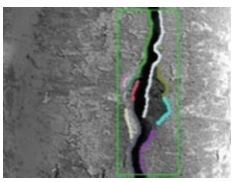
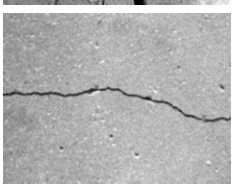
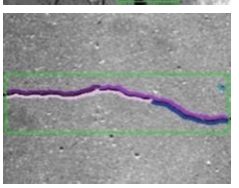
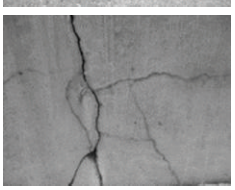
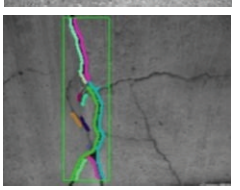
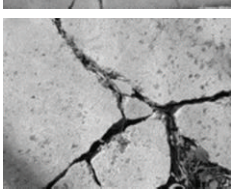
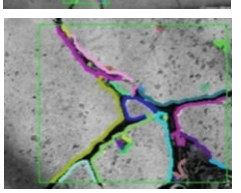
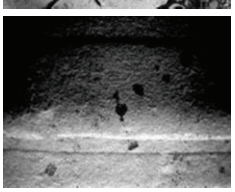
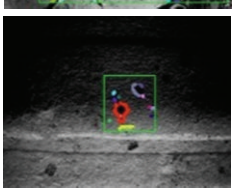
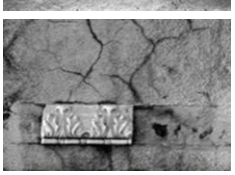
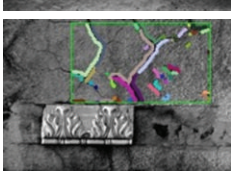
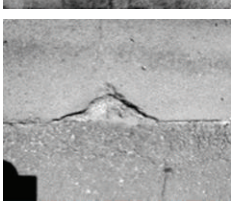
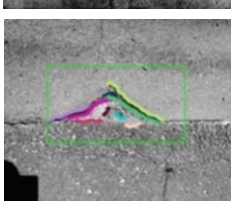
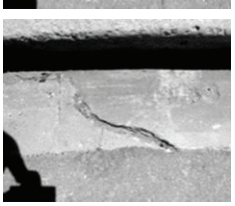
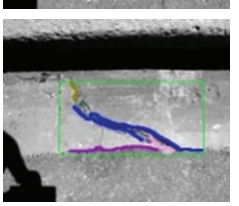
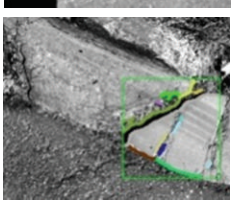
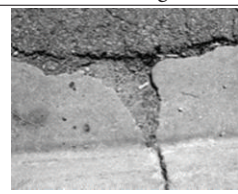
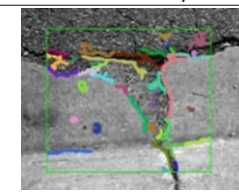
| Number | Raw image | Edge detection | Contour analysis | (Len, area) |
|---|---|---|---|---|
| 1 |  |  |  | (564, 84) |
| 2 |  |  |  | (559, 51) |
| 3 |  |  |  | (538, 105) |
| 4 |  |  |  | (1330, 379) |
| 5 |  |  |  | (129, 30) |
| 6 |  |  |  | (724, 211) |
| 7 |  |  |  | (369, 95) |
| 8 |  |  |  | (435, 89) |
| 9 |  |  |  | (483, 132) |

Table 2: Continued.

| Number | Raw image | Edge detection | Contour analysis | (Len, area) |
|---|---|---|---|---|
| 10 |  |  |  | (1342, 328) |

results actually return coordinates of detected contours, which enable fast computation of geometric parameters of the damage. The last column of Table 2 reports the summed lengths and areas of detected damage for all the images. These statistics, when translated into real units (through a simple geometric calibration process), provide key decision-making aids during field damage assessment.

*4.5. Computational Performance Evaluation.* One factor that is critical to success of mobile imaging and computing for improved and intelligent structural damage inspection is its computational performance. To evaluate the performance of mobile imaging and computing, we differentiate two major operations in the proposed MIC framework as well as in the developed prototype. One is interactive parameter selection or engineering knowledge logging. The other is automatic image analysis running in the background of a smart device. For the former, there is no need for evaluating its computational efficiency, since it is part of user-engaged manual operation amid field-based decision-making. Therefore, the performance evaluation is focused on the image analysis components. Therefore, the most obvious and relevant task is that whether mobile computing can encourage or discourage real-time decision-making (such as onsite condition assessment) in field inspection activities. The major parameter is time consumption for a particular computing task. Then the primary criterion is to justify that if the time cost is less than 1.0 sec—if any analysis is greater than 1.0 sec then users may feel delay of operation which discourages real-time decision-making.

In the following, we conduct a comparative study by comparing the computational performance of the image analysis methods when implemented using a smart device and a workstation. In this study, two representative computing platforms are used. The smart mobile device is the Google's Nexus 7 tablet, which features an Nvidia 1.6 GHz quad-core CPU, 1 G RAM, and an Android 4.3 operating system. The workstation is a Linux computer featuring Intel Core 2 Duo CPU E8400 with 3 GHz frequency, 4 G RAM, and Ubuntu Linux 12. The two visual analysis methods are implemented in the prototype as presented earlier.

(1) Image enhancement includes the conjunct use of histogram equalization and mean-shift filtering. However, we note that the method of histogram equalization (the function "*equalizeHist*" in OpenCV) is a fairly low-cost algorithm, which processes images regardless of regular image sizes with time negligible to users (for images with sizes up to $576 \times 720$, the time consumption is about $0.3 \times 10^{-3}$ to $0.9 \times 10^{-3}$ sec, or 1 to 9 millisec using the workstation, and about 1.5 to 5.0 millisec using the smart tablet). On the other hand, the time-cost and efficiency of the mean-shift method highly depend on image sizes. Considering the previous range of image sizes, the time cost of mean-shift filtering may be up to the order of one second. Therefore, in this paper, only the mean-shift filtering (using "*meanShiftFiltering*" in OpenCV with $\sigma_s = 3$ and $\sigma_r = 6$) is evaluated for both computational platforms.

(2) Contour analysis uses an embedded Canny detector for binary edge generation. Due to the large number of images, optimal Canny parameters are not sought from the gesture-based interactive selection using the interface shown in Figure 5. Instead a standard Canny parameter set $(\sigma, \alpha)$ is used for the contour analysis in both platforms ($\sigma = 2$, $\alpha = 40\%$). In addition, bounding box is not used; rather, the contour analysis is conducted over the total image domain. For contour analysis, two performance measures are used: one is the time cost and the other is the number of contours. The rationale for outputting the number of contours is that it is a better indicator of the underlying scene complexity in the image.

The performance evaluation is tested on the imagery database based on the Internet search. The rationale is that, first, image size is obviously the primary factor that affects the computational time; second, images taken in the field vary in sizes when different imaging devices are used or user may choose different resizing ratios to decrease the image sizes. The Internet-searched images with different contents vary in sizes from $200 \times 300$ (about 0.05 Megapixel) to $1024 \times 1024$ (or 1 Megapixel). Therefore, this imagery database serves as an appropriate testing database. Seventy images are chosen and tested on the two platforms for both the image analysis procedures above. In the meantime, since a mean-shift filter uses an iterative procedure to locate local maxima of density functions, it is possible that when image sizes are fixed, image contents may affect the computational time. Therefore, the need of performing a variability evaluation of time cost using the same sized images with different scene complexities is warranted.

Figure 9 presents the scatter plots of the running time from the image enhancement procedure (i.e., using the mean-shift filtering method) against the image size of input images,

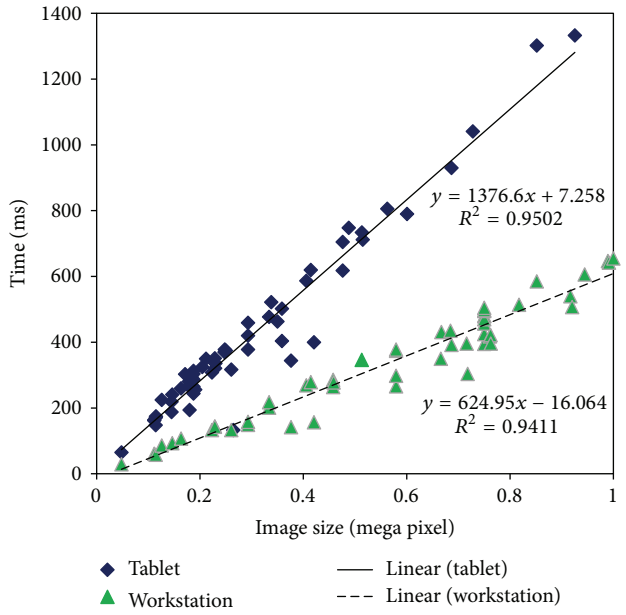



Figure 9: Computational time for image enhancement with different image size.

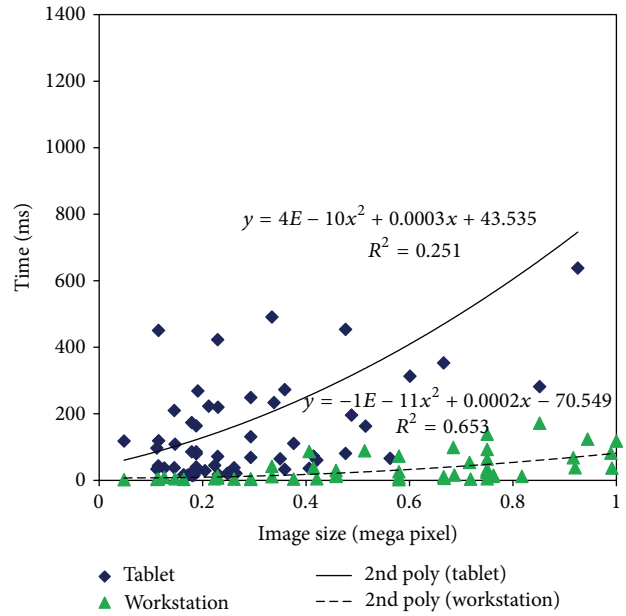



Figure 11: Computational time for contour analysis with different image size.

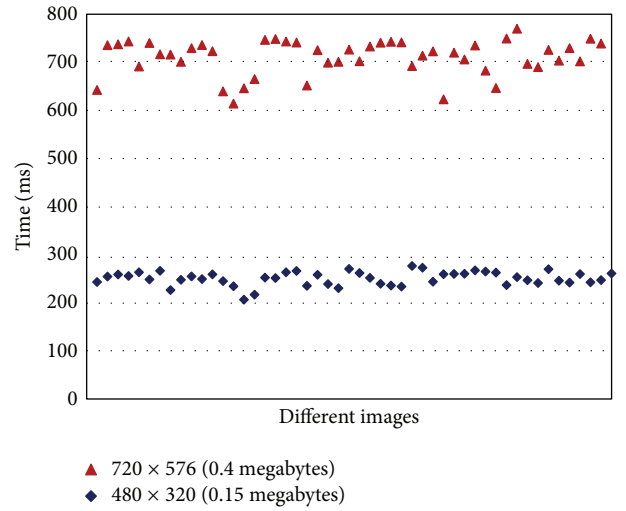



Figure 10: Computational time variability for fixed-size images for image enhancement in the tested smart tablet.

resulting from the smart tablet and the workstation, respectively. First of all, one can see that the image sizes in terms of mega pixels statistically predict well the time consumption. Comparing between the tablet and the workstation, the regression analysis returns that the efficiency of using the smart tablet is about 1376 millisec/Megapixel, whereas it is 625 millisec/Megapixel for the workstation. The tablet is about 45% of the efficiency of the workstation, when the mean-shift filtering function is used. This is not a surprising result considering the lower-end computing hardware configuration of the mobile device. In the meantime, when small to moderate-sized images are used, for example, 480 × 320 (or 0.15 mega pixels), the time consumption from Figure 9 is about 200 millisec if the smart device is used.

To further evaluate the potential variability of time consumption on image contents, Figure 10 illustrates the time consumption results of using two different sizes of images representing small to moderate sizes. When images with a size of 480 × 320 are used, the image enhancement time ranges from 200 to 290 milliseconds, with an average value of 250 milliseconds and a standard deviation of 14.4 milliseconds. For images of 720 × 576 (0.4 mega pixels), the running time is 600 to 780 milliseconds with an average value of 710 milliseconds and a standard deviation of 36.7 milliseconds. This experiment shows the overall consistency of predicting time consumption using a fixed image size. Therefore, it is ready to reason that if an image has a small to moderate size, the time cost is statistically less than 800 millisec (or 0.8 sec) using the smart tablet. This statistical study indicates that performing the proposed low-level image enhancement does not prevent the overall interactive imaging and computing as well as real-time decision-making.

For the damage quantification analysis using the contour detection method, Figure 11 illustrates the running time against the image size. However, in this case, the image size is not a good predicator of the potential running time reflected by the considerable scatterings in the plot. Nonetheless, it seems that when the image size is less than 0.4 Megapixels, the contour analysis costs less than 300 millisec when the tablet is used. This is a promising result in light of realizing real-time imaging and computing. In addition, one can observe with a weak statistical significance that the workstation is about 10 times faster than the tablet (based on the first-order polynomial coefficients of the 2nd order regression results).

It is found that the number of contours in the images, which is a direct measure of the scene complexity in the

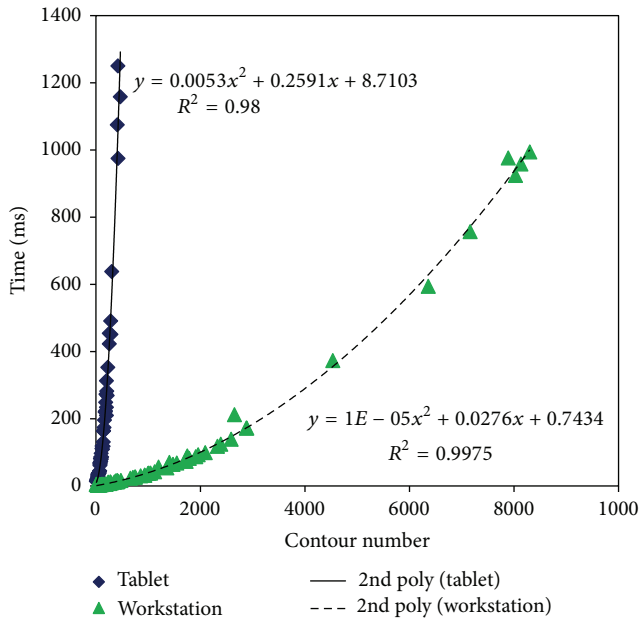



Figure 12: Computational time for contour analysis with different scene complexity.

image domain, is a much better predictor of the time consumption. In Figure 12, the scatter plots of time consumption and number of contours are plotted. Using second-order polynomial fitting, the underlying plots yield much improved statistical significance ($R^2 > 0.98$ in both cases). The plots indicate that it is the number of contours not the image size that can accurately predict the time consumption. First, the difference between the smart tablet and the workstation is obvious. For the workstation, the time cost relatively slowly grows against the number of contour (or equivalently the scene complexity), whereas in the case of the tablet, the time cost increases rapidly as the number of the contours increases. For example, regardless of the image size, if the number of contours reduces from 300 to 100, the time cost changes from 600 millisec to about 60 millisec using the tablet. This difference is due to the underlying different hardware and software configurations. When the number of contours is less than 100, one can reason that the proposed contour analysis will not impose significant computational hurdle from realizing real-time imaging, computing, and decision computing. This observation implies that edge/line-preserving smoothing should be seriously considered to remove noisy contours prior to any advanced image analysis for damage quantification when a smart mobile device is used.

## 5. Conclusions

The main purpose of this paper is to explore the potential of modern mobile imaging and computing for its use in structural damage inspection in the routine practice of civil infrastructure management. In this paper, we address the critical need that arises from performing intelligent and efficient inspection within a complex or hard-access build environment, wherein visual inspection dominates the practice.

Through developing a mobile imaging and computing prototype, the proposed mobile damage inspection is verified and evaluated in terms of both practical feasibility and computational performance for enabling real-time field-based decision-making. Particularly, the following technical conclusions are obtained using the state-of-the-art smart mobile device (e.g., the commercial Google Nexus 7 tablet).

(1) Interactively logged and selected contextual parameters, including types of damage, preliminary image analysis parameters, and bounding boxes, are essential towards yielding quality damage detection and quantification. If these parameters are otherwise not available, significant analysis challenges will be encountered for processing structural damage patterns as complex natural scenes of the built environment.

(2) In most applications, it is suggested that image sizes should be at a moderate level, for example, 576 × 720 (or 0.4 Megabytes). Such level of image sizes is considered appropriate for processing common damage patterns (e.g., cracks) using smart devices.

(3) Typical image enhancement should be conducted and can be implemented in real-time in mobile devices while accompanying the imaging and computing activity during field inspection.

(4) Complexity of image contents can be reduced through two operations: (i) through the selection of bounding boxes and (ii) through image smoothing such as the mean-shift filtering in this paper. The two operations can significantly reduce the number of contours thus enabling real-time damage analysis.

With these technical conclusions, we finally conclude that mobile imaging and computing via today's smart devices possess tremendous potential for realizing intelligent and efficient structural damage inspection, and such smart applications are specially justified when the inspection areas are complex or hard to access.

Last, it should be pointed that for quantitative damage extraction using mobile devices is subject to more image-based development. In practice, the common concrete damage statistics include length and area, which are readily to be extracted using the methods in this paper. For crack, width is an important index as well. However, for crack width, it is a complex image analysis problem involving extraction of the central line of a crack. One approach has been proposed in the first author's previous work [6], which could be implemented for mobile smart devices. However, these potential quantitative extraction and methods are subject to field validation, which is not in the scope of this paper.

## Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

## References


[1] F. N. Catbas and A. E. Aktan, "Condition and damage assessment: issues and some promising indices," *Journal of Structural Engineering*, vol. 128, no. 8, pp. 1026–1036, 2002.

[2] B. R. Ellingwood, "Risk-informed condition assessment of civil infrastructure: state of practice and research issues," *Structure and Infrastructure Engineering*, vol. 1, no. 1, pp. 7–18, 2005.

[3] M. Moore, B. Phares, B. Graybeal, D. Rolander, and G. Washer, *Reliability of Visual Inspection for Highway Bridges, Volume I*, Federal Highway Administration, 2001.

[4] M. J. Olsen, Z. Chen, T. Hutchinson, and F. Kuester, "Optical techniques for multiscale damage assessment," *Geomatics, Natural Hazards and Risk*, vol. 4, no. 1, pp. 49–70, 2013.

[5] C. Koch and I. Brilakis, "Pothole detection in asphalt pavement images," *Advanced Engineering Informatics*, vol. 25, no. 3, pp. 507–515, 2011.

[6] Z. Chen and T. C. Hutchinson, "Image-based framework for concrete surface crack monitoring and quantification," *Advances in Civil Engineering*, vol. 2010, Article ID 215295, 18 pages, 2010.

[7] S. R. Cruz-Ramírez, Y. Mae, T. Arai, T. Takubo, and K. Ohara, "Vision-based hierarchical recognition for dismantling robot applied to interior renewal of buildings," *Computer-Aided Civil and Infrastructure Engineering*, vol. 26, no. 5, pp. 336–355, 2011.

[8] T. Nishikawa, J. Yoshida, T. Sugiyama, and Y. Fujino, "Concrete crack detection by multiple sequential image filtering," *Computer-Aided Civil and Infrastructure Engineering*, vol. 27, no. 1, pp. 29–47, 2012.

[9] M. R. Jahanshahi and S. F. Masri, "Adaptive vision-based crack detection using 3D scene reconstruction for condition assessment of structures," *Automation in Construction*, vol. 22, pp. 567–576, 2012.

[10] H. Cheng and C. Glazier, *Automated Real-Time Pavement Crack Deflection/Classification System*, IDEA Program, Transportation Research Board, 2002.

[11] S. Ghanta, R. Birken, and J. Dy, "Automatic road surface defect detection from grayscale images," in *Proceedings of the SPIE 8347, Nondestructive Characterization for Composite Materials, Aerospace Engineering, Civil Infrastructure, and Homeland Security*, vol. 83471E, 2012.

[12] C. Balaguer, A. Giménez, J. M. Pastor, V. M. Padrón, and M. Abderrahim, "Climbing autonomous robot for inspection applications in 3D complex environments," *Robotica*, vol. 18, no. 3, pp. 287–297, 2000.

[13] J.-K. Oh, G. Jang, S. Oh et al., "Bridge inspection robot system with machine vision," *Automation in Construction*, vol. 18, no. 7, pp. 929–941, 2009.

[14] S.-N. Yu, J.-H. Jang, and C.-S. Han, "Auto inspection system using a mobile robot for detecting concrete cracks in a tunnel," *Automation in Construction*, vol. 16, no. 3, pp. 255–261, 2007.

[15] C. Eschmann, C.-M. Kuo, C.-H. Kuo, and C. Boller, "Unmanned aircraft systems for remote building inspection and monitoring," in *Proceedings of the 6th European Workshop on Structural Health Monitoring (EWSHM '12)*, pp. 1179–1186, Dresden, Germany, July 2012.

[16] S. Rathinam, Z. W. Kim, and R. Sengupta, "Vision-based monitoring of locally linear structures using an unmanned aerial vehicle," *Journal of Infrastructure Systems*, vol. 14, no. 1, pp. 52–63, 2008.

[17] N. Metni and T. Hamel, "A UAV for bridge inspection: visual servoing control law with orientation limits," *Automation in Construction*, vol. 17, no. 1, pp. 3–10, 2007.

[18] G. H. Forman and J. Zahorjan, "The challenges of mobile computing," *Computer*, vol. 27, no. 4, pp. 38–47, 1994.

[19] Mobithinking, "Global mobile statistics 2013 Part A: mobile subscribers; handset market share; mobile operators," 2013, http://mobithinking.com/mobile-marketing-tools/latest-mobil-stats/a.

[20] K. Pulli, W.-C. Chen, N. Gelfand et al., "Mobile visual computing," in *Proceedings of the International Symposium on Ubiquitous Virtual Reality (ISUVR '09)*, pp. 3–6, July 2009.

[21] I. Gartner, "Gartner says worldwide sales of mobile phones," 2013, http://www.gartner.com/newsroom/id/2017015.

[22] D. Ehringer, "The dalvik virtual machine architecture," Tech. Rep., 2010.

[23] G. Bradski and A. Kaehler, *Learning OpenCV: Computer Vision with the OpenCV Library*, O'Reilly Media, 2008.

[24] OpenCV, "*Open Source Computer Vision*," 2013, http://opencv.org/.

[25] W.-H. Cho and T.-C. Kim, "Image enhancement technique using color and edge features for mobile systems," in *Digital Photography VII*, Proceedings of SPIE, San Francisco, Calif, USA, January 2011.

[26] T. Yeh, K. Grauman, K. Tollmar, and T. Darrell, "A picture is worth a thousand keywords: image-based object search on a mobile platform," in *Proceedings of the Extended Abstracts on Human Factors in Computing Systems (CHI '05)*, pp. 2025–2028.

[27] R. Andrade, A. V. Wangenheim, and M. K. Bortoluzzi, "Wireless and PDA: a novel strategy to access DICOM-compliant medical data on mobile devices," *International Journal of Medical Informatics*, vol. 71, no. 2-3, pp. 157–163, 2003.

[28] Y. Kondo, "Medical image transfer for emergency care utilizing internet and mobile phone," *Nippon Hoshasen Gijutsu Gakkai Zasshi*, vol. 58, no. 10, pp. 1393–1401, 2002.

[29] Z. Tu and R. Li, "Automatic recognition of civil infrastructure objects in mobile mapping imagery using a markov random field model," in *Proceedings of the 19th Congress of ISPRS*, pp. 16–23, 2000.

[30] C. Tao, R. Li, and M. A. Chapman, "Automatic reconstruction of road centerlines from mobile mapping image sequences," *Photogrammetric Engineering and Remote Sensing*, vol. 64, no. 7, pp. 709–716, 1998.

[31] D. Wagner, G. Reitmayr, A. Mulloni, T. Drummond, and D. Schmalstieg, "Real-time detection and tracking for augmented reality on mobile phones," *IEEE Transactions on Visualization and Computer Graphics*, vol. 16, no. 3, pp. 355–368, 2010.

[32] Y. Liu, J. Yang, and M. Liu, "Recognition of QR Code with mobile phones," in *Chinese Control and Decision Conference, 2008 (CCDC '08)*, pp. 203–206, Yantai, China, July 2008.

[33] A. Sun, Y. Sun, and C. Liu, "The QR-code reorganization in illegible snapshots taken by mobile phones," in *Proceedings of the International Conference on Computational Science and Its Applications (ICCSA '07)*, pp. 532–538, Kuala Lampur, Malaysia, August 2007.

[34] J. J. Hull, X. Liu, B. Erol, J. Graham, and J. Moraleda, "Mobile image recognition: architectures and tradeoffs," in *Proceedings of the 11th Workshop on Mobile Computing Systems Applications*, pp. 84–88, 2010.

[35] T. Nagayama, A. Miyajima, S. Kimura, Y. Shimada, and Y. Fujino, "Road condition evaluation using the vibration response

of ordinary vehicles and synchronously recorded movies," in *Proceedings of the SPIE Smart Structures and Materials+ Nondestructive Evaluation and Health Monitoring*, p. 86923A, March 2013.

[36] R. Bhoraskar, N. Vankadhara, B. Raman, and P. Kulkarni, "Wolverine: traffic and road condition estimation using smartphone sensors," in *Proceedings of the 4th International Conference on Communication Systems and Networks (COMSNETS '12)*, pp. 1–6, Bangalore, India, January 2012.

[37] D. A. Johnson and M. M. Trivedi, "Driving style recognition using a smartphone as a sensor platform," in *Proceedings of the 14th IEEE International Intelligent Transportation Systems Conference (ITSC '11)*, pp. 1609–1615, Washington, DC, USA, October 2011.

[38] A. H. Behzadan, Z. Aziz, C. J. Anumba, and V. R. Kamat, "Ubiquitous location tracking for context-specific information delivery on construction sites," *Automation in Construction*, vol. 17, no. 6, pp. 737–748, 2008.

[39] K. C. Yeh, M. H. Tsai, and S. C. Kang, "The iHelmet: an AR-nhanced wearable display for BIM information," in *Mobile and Pervasive Computing in Construction*, pp. 149–168, Wiley-Blackwell, 2012.

[40] Z. Chen, R. R. Derakhshani, C. Halmen, and J. T. Kevern, "A texture-based method for classifying cracked concrete surfaces from digital images using neural networks," in *Proceedings of the International Joint Conference on Neural Network (IJCNN '11)*, pp. 2632–2637, San Jose, Calif, USA, August 2011.

[41] D. A. Forsyth and J. Ponce, *Computer Vision: A Modern Approach*, Prentice Hall, New York, NY, USA, 2002.

[42] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000.

[43] C. Sphinx, "Open source toolkit for speech recognition," 2011, http://cmusphinx.sourceforge.net/.

[44] A. K. Jain and F. Farrokhnia, "Unsupervised texture segmentation using Gabor filters," *Pattern Recognition*, vol. 24, no. 12, pp. 1167–1186, 1991.

[45] L. A. Vese and T. F. Chan, "A multiphase level set framework for image segmentation using the Mumford and Shah model," *International Journal of Computer Vision*, vol. 50, no. 3, pp. 271–293, 2002.

[46] H. D. Cheng and X. J. Shi, "A simple and effective histogram equalization approach to image enhancement," *Digital Signal Processing: A Review Journal*, vol. 14, no. 2, pp. 158–170, 2004.

[47] D. Comaniciu and P. Meet, "Mean shift analysis and applications," in *Proceedings of the 7th IEEE International Conference on Computer Vision (ICCV '99)*, pp. 1197–1203, September 1999.

[48] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, no. 6, pp. 679–698, 1986.

[49] R. Koren and Y. Yitzhaky, "Automatic selection of edge detector parameters based on spatial and statistical measures," *Computer Vision and Image Understanding*, vol. 102, no. 2, pp. 204–213, 2006.

[50] Y. Yitzhaky and E. Peli, "A method for objective edge detection evaluation and detector parameter selection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 8, pp. 1027–1033, 2003.

[51] S. Kichenassamy, A. Kumar, P. Olver, A. Tannenbaum, and A. Yezzi, "Gradient flows and geometric active contour models," in *Proceedings of the 5th International Conference on Computer Vision*, pp. 810–815, 1995.

[52] V. Caselles, R. Kimmel, and G. Sapiro, "Geodesic active contours," *International Journal of Computer Vision*, vol. 22, no. 1, pp. 61–79, 1997.

[53] S. Suzuki and K. be, "Topological structural analysis of digitized binary images by border following," *Computer Vision, Graphics and Image Processing*, vol. 30, no. 1, pp. 32–46, 1985.

[54] Google, *Find Content to Reuse*, Google, 2013, https://support.google.com/websearch/answer/29508.

[55] Z. Chen, "*UMKC Concrete Damage Imagery Database*," 2013, http://lasir.umkc.edu/cdid/.