COMPUTER-AIDED CIVIL AND INFRASTRUCTURE ENGINEERING

Computer-Aided Civil and Infrastructure Engineering **00** (2018) 1–18

Autonomous UAVs for Structural Health Monitoring Using Deep Learning and an Ultrasonic Beacon System with Geo-Tagging

Dongho Kang & Young-Jin Cha*

Department of Civil Engineering, University of Manitoba, Winnipeg, MB, Canada

Abstract: *Visual inspection has traditionally been used for structural health monitoring. However, assessments conducted by trained inspectors or using contact sensors on structures for monitoring are costly and inefficient because of the number of inspectors and sensors required. To date, data acquisition using unmanned aerial vehicles (UAVs) equipped with cameras has become popular, but UAVs require skilled pilots or a global positioning system (GPS) for autonomous flight. Unfortunately, GPS cannot be used by a UAV for autonomous flight near some parts of certain structures (e.g., beneath a bridge), but these are the critical locations that should be inspected to monitor and maintain structural health. To address this difficulty, this article proposes an autonomous UAV method using ultrasonic beacons to replace the role of GPS, a deep convolutional neural network (CNN) for damage detection, and a geo-tagging method for the localization of damage. Concrete cracks, as an example of structural damage, were successfully detected with 97.7% specificity and 91.9% sensitivity, by processing video data collected from an autonomous UAV.*

1 INTRODUCTION

Traditional structural health monitoring (SHM) approaches usually require a dense array of contact sensors to measure vibrations of the structures or human inspector assessments. This makes it expensive to install and maintain a monitoring system. To overcome these issues, many computer vision-based noncontact sensing techniques have been developed (Abdel-Qader et al., 2003; Lee and Shinozuka, 2006; Chen et al., 2015; Cha et al., 2016).

Recently, Cha et al. (2017a) proposed a deep learning-based crack damage detection method with automatic feature extraction and the ability to learn damage-sensitive features with robustness to various types of environmental noise. The convolutional neural network (CNN)-based method effectively extracts and learns damage-sensitive features from input image data. Unlike a standard artificial neural network, it does not require definition of specific damage-sensitive features. Interest in the SHM discipline has been increasing in the application of the powerful CNN approach (Soukup and Huber-Mörk, 2014; Lin et al., 2017; Rafiei et al., 2017). And other recent engineering applications of deep learning have been researched for SHM (Koziarski and Cyganek, 2017; Ortega-Zamorano et al., 2017; Rafiei and Adeli, 2017). Moreover, the faster region-based CNN (Faster R-CNN) method (Ren et al., 2015) has been applied to the detection and localization of multiple damage types for a steel girder bridge (Cha et al., 2017b).

To maximize the use of computer vision sensors, unmanned aerial vehicles (UAVs) have been applied to SHM problems (Metni and Hamel, 2007; Chen et al., 2011; Eschmann et al., 2012; Zhang and Elaksher, 2012; Hallermann and Morgenthal, 2013; Sankarasrinivasan et al., 2015; Gillins et al., 2016). UAVs offer cost-efficient risk reduction even if the location monitored is isolated or hazardous (Metni and Hamel, 2007). They also provide a time-saving solution (Gillins et al., 2016) for data acquisition (Eschmann et al., 2012; Hallermann and Morgenthal, 2013; Sankarasrinivasan et al., 2015). However, skilled pilots are typically required to run UAVs on-site, even though some techniques have been developed for remote control (Eschmann et al., 2012; Gillins et al., 2016). Autonomous navigation methods of UAVs have been studied to address this drawback.

*To whom correspondence should be addressed. E-mail: *young.cha@umanitoba.ca*.

DOI: 10.1111/mice.12375

To the best of our knowledge, there is no clear definition or established concept and theory for the levels of autonomous UAV navigation in the robotics discipline. However, six levels of autonomous vehicle navigation have been defined by the National Highway Traffic Safety Administration (NHTSA and SAE International, 2014), which is a part of the U.S. Department of Transportation. The six levels of autonomous vehicle navigation are also applicable to autonomous UAV navigation. Level 0 is completely manual control of navigation by pilots. Level 1 is UAV navigation performed by pilots but with some automation applied to specific flight modes, such as holding altitude and hovering. In Level 2 automation, users can define multiple flight modes for automation, and the UAV then navigates based on the scheduled flight modes if there is no unexpected change in the flying environment. In Level 3, a UAV understands changing flying environments and controls flight modes itself to navigate the new environments. In Level 4 navigation, a UAV can adaptively react when there is any system anomaly or a sudden accident, such as a collision with other objects. In Level 5, a UAV can autonomously navigate in all environments and situations. In this article, we focus on realizing Level 2 automation of UAV navigation for SHM in GPS-denied areas or complex geometric navigation environments.

Mapping and localization are critical to realization of Level 2 autonomous navigation. Planning and control of UAV navigation can be accomplished using an existing commercial mission planner and flight controller (i.e., Pixhawk). Multiple types of sensors are available for a UAV to determine its position. These sensors provide vehicle position data for the UAV to conduct its scheduled mission. For example, GPS is the most popular option for position sensors, as GPS sensors are cheaper and easier to use than other types of position sensors. A simple autonomous outdoor navigation by waypoints has been demonstrated using the GPS for localization of the UAV (Carvalho et al., 2017). However, there are several reasons why other localization sensor systems are required. First, the usage of a UAV is often limited to outdoor environments because a GPS signal is not reliable in certain locations like beneath bridge decks. Second, dynamic and complex topographic environments of the navigation space for SHM require higher accuracy in UAV localization than commercial GPS can provide. For example, dynamic water levels under bridge systems and complex indoor geometries of civil infrastructures require localization accuracy exceeding that of GPS systems.

To overcome these problems, a distance sensor and optical flow (Honegger et al., 2013) can be used with the simultaneous localization and mapping (SLAM) technique. However, the performance of SLAM and optical flow are dependent on the environment. For example, if the vision sensors cannot obtain features adequate to identify a UAV's location, they incur a high computational cost and accumulate localization errors (Hess et al., 2016). The real-time kinematic global positioning system (RTK GPS) was developed to address this issue (Stempfhuber and Buchholz, 2011). RTK GPS is highly accurate, but it is still not available in GPS-denied environments. As another approach, motion capture-based localization provides inspiration to extensive research in aerial robotics, allowing for complex and high-precision navigation that does not require any satellite signal (Orsag et al., 2013). However, its implementation requires a complex and expensive motion capture system (Deutscher et al., 2000).

An ultra-wideband beacon system was developed to provide high precision positioning to enable a new range of applications in GPS-denied environments (Vossiek et al., 2003; Zwirello et al., 2012; Sung et al., 2016). However, some experiments have shown millimeter-level accuracy of ultra-wideband beacon positioning systems, but the direct application is not practical in UAV systems due to high cost and a lack of integration (Zhang et al., 2006). However, an ultrasonic beacon system (UBS) can be an alternative for a practical mapping and localization system using low-cost hardware. The UBS offers a similar concept and mechanism to the ultra-wideband beacon, but it is cheaper and easier to integrate into UAVs than the ultra-wideband beacon. The UBS provides centimeter-level accuracy with proper parameter tuning (Díaz et al., 2017). For this reason, we chose UBS as the local mapping and localization sensor for Level 2 autonomous navigation of a UAV equipped with a camera for structural damage detection.

In the present study, we propose an autonomous UAV-based SHM method using UBS. We used CNN with a sliding window technique (Cha et al., 2017a) as an example damage detection method. The detected damage is localized by geo-tagging method. Section 2 describes the UBS-based autonomous UAV system and CNN layers that we used for concrete crack detection. In Section 3, experimental tools, including UAV fabrication and UBS are described and test scenarios and the results of UAV autonomous navigation experiments are discussed. Section 4 provides conclusions, discusses study limitations, and suggests future improvements.

2 METHODOLOGY

To develop autonomous navigation for a UAV, UBS was used for local mapping and localization positioning

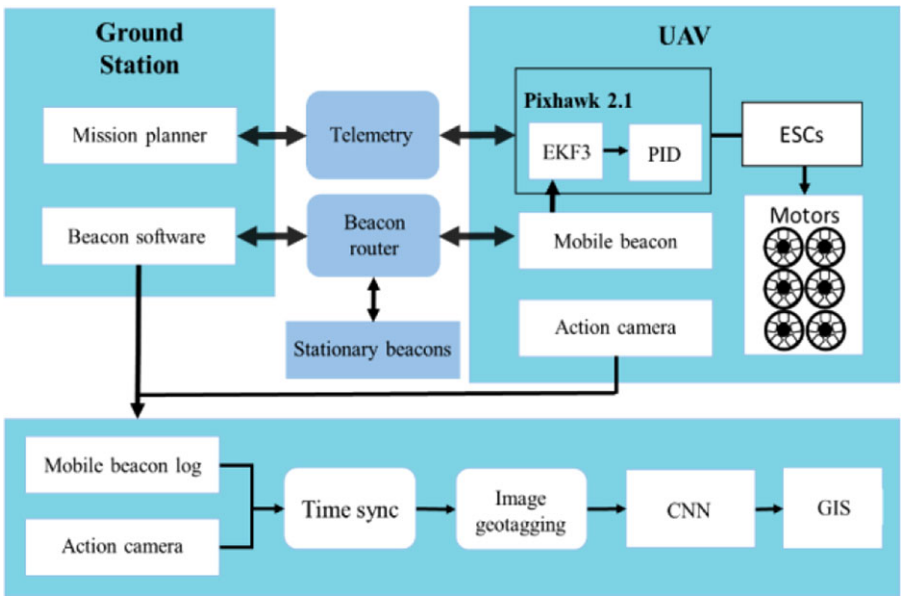


Fig. 1. Overall architecture of an autonomous UAV using a Pixhawk 2.1 flight controller.

sensors, a ground station including a mission planner was used to assign a navigation plan, and two UAVs were fabricated and equipped with a flight controller and an action camera. Figure 1 depicts the architecture of the autonomous UAV using a Pixhawk 2.1 flight controller. A commodity computer can serve as a ground station, where the role of mission planner is to assign a navigation plan and monitor the UAV. The first system used Pixhawk 2.1, which is commercial flight controller hardware (Proficnc, 2017). The Pixhawk 2.1 has Ardupilot 3.5 (Ardupilot, 2017), which is open source, installed as firmware. Ardupilot 3.5 has a feedback proportional–integral–derivative (PID) controller (Minorsky and Nicolas, 1922; Lim at al., 2012) that was used for this study as a default controller to control the speed of motors through the electronic speed controllers (ESCs). The position data from a mobile beacon was estimated using an extended Kalman filter 3 (EKF3) algorithm as input to the PID control system. The UBS has multiple mobile and stationary beacons. The EKF3 is an updated version of the original EKF (Smith et al., 1962) for flight control. Image geo-tagging was conducted based on the runtime history of the video footage collected from the action camera and the UAV running time. A CNN with a sliding window technique (Cha et al., 2017a) was used for crack detection in concrete as an example of a type of structural damage. A geographic information system (GIS) was also used to collect all video data of damage detected in the structure, along with location information. The details of autonomous flight, using UBS and CNN, are explained in the following sections.

Figure 2 depicts the architecture of an autonomous UAV using the commercially available drone, Parrot Bebop2 Power (Bebop2). The only difference between this drone and the first UAV is that the commercial Bebop2 UAV has an imbedded controller. It was adopted because it is capable of more complex autonomous navigation than the previous UAV, and the experimental space at the University of Manitoba was limited. The previously described Pixhawk UAV is larger than the Bebop2; therefore, only limited missions were possible with this vehicle.

2.1 Programmable UAV fabrication and mapping system

As the first step for Level 2 autonomous navigation of Pixhawk UAV, a programmable UAV was fabricated to enable modification of the flight controller source code. The fabricated programmable Pixhawk UAV shown in Figure 1 includes various hardware components: legs and motors, telemetry, ESCs, a flight controller, a mobile beacon, an action camera with vibration damper, and batteries. A detailed view of the components is presented in Figure 3.

We modified the DJI F550 frame of a multirotor Erle copter by installing six propeller motors as the UAV frame. It can carry a 2 kg payload for 15 minutes of flight time. The six motors are brushless motors that require 30A ESCs. For telemetry, a 3DR 915 MHz radio is used to communicate between the ground station and UAV. The Pixhawk 2.1 was selected as a flight controller due to its improved inertial measurement unit

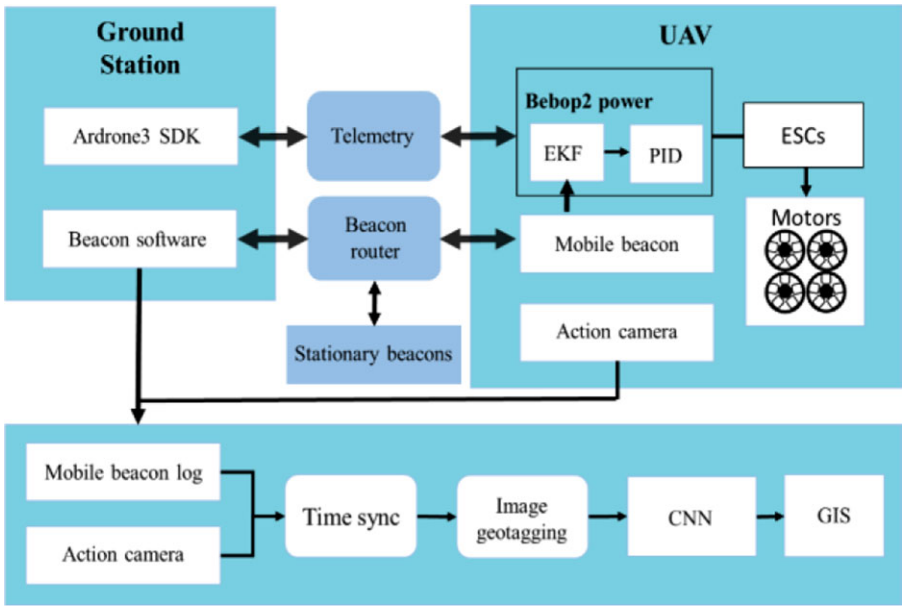


Fig. 2. Overall architecture of autonomous commercial Bebop2 UAV.

(IMU) (Meier et al., 2011). The Ardupilot 3.5 open-source code was installed in the Pixhawk. Cube design was applied to the Pixhawk 2.1 to reduce vibration (Proficnc, 2017).

To monitor a region of interest (ROI), a Sony FDR-X3000 action camera was installed in the UAV as shown in Figure 3c. Its 60 frames per second (FPS) recording capability ensures stable video even if the camera is vibrated by the UAV motors. The camera supports 4K image resolution, but the 1,080 pixel resolution was selected due to shutter speed. Sony's Playmemories application allows the user to remotely check and change the camera setting. The lightweight camera (114 g with a battery) is ideal for a UAV application. Antivibration foam that serves as a gimbal was installed to reduce the jello-effect.

As the second UAV used in this study, the Bebop2 has its own camera, but the camera angle was modified to a right angle to the ground to collect clear images of the concrete surface and detect cracks, as shown in Figure 4. It uses a removable battery (3,350 mAh), which allows the UAV to fly up to 30 minutes. A mobile beacon was installed on top of the Bebop 2 UAV, as depicted in Figure 4. The Bebop2 is programmable, and the manufacturer also offers a software development kit.

A mobile beacon was installed in the UAV frame to provide 3-dimensional (3-D) position data (x, y, z) of the two UAVs. The Marvelmind Robotics UBS is composed of a mobile beacon, multiple stationary beacons, a router, and the Dashboard beacon software, as shown in Figure 5. The UBS generates a local pseudo-3-D map, which is depicted by the blue dotted line in Figure 5. The ROI should be located within this pseudo-3-D map. The stationary beacons define the border lines of the map and can be installed on the wall or with a tripod. Each beacon has five transceivers.

The stationary beacons are similar to a GPS, sending ultrasonic signals and calculating distances to the mobile beacon installed in the UAV through a router. The position of the mobile beacon can be calculated using Equation (1) below:

$$p(t) = \sqrt{(x_s - x_m)^2 + (y_s - y_m)^2 + (z_s - z_m)^2} \quad (1)$$

where (x_s, y_s, z_s) represents the stationary beacon, and (x_m, y_m, z_m) represents the mobile beacon's coordinates. The Marvelmind UBS supports the GPS format of the National Marine Electronics Association (NMEA). The mobile beacon should always be within the pseudo-3-D beacon map for navigation to work properly. In this study, the mobile beacon was installed on top of the UAV to avoid blocking the beacon signal. We used the programmable Ardupilot 3.5 open source code in the Pixhawk 2.1 flight controller to program the autonomous navigation of the first Pixhawk UAV based on the pseudo-3-D beacon map. The modified source code was developed in C++. The open source code was injected through the mission planner in the ground station (Carvalho et al., 2017). For the Bebop2 UAV, the firmware provided by the manufacturer was used.

(a) Fabricated UAV system

(b) Pixhawk 2.1

(c) Sony action camera

(d) Mobile beacon

(e) Telemetry

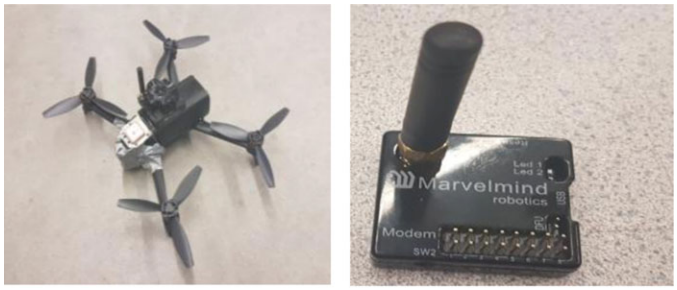
Fig. 3. Components of the fabricated Pixhawk UAV.

2.2 Ground station

A Samsung nt500r5h laptop, a commodity commercial computer, was used as the main ground station computer. It has a 2.2 GHz computer processing unit (CPU) and 8 GB memory. For the Pixhawk UAV, Mission Planner and beacon software were installed at the ground station. The Bebop2 did not require a Mission Planner software but used the same ground station computer. The Mission Planner is open source software that provides a graphical user interface (GUI) to manage and monitor the navigation of the UAV. The Mission Planner has many roles. It displays the status of a UAV through the MAVLink protocol (Meier et al., 2011). The Mission Planner can record and replay the log data of a UAV flight. Error messages associated with the status and navigation plan, including environmental noise such as magnetic interference, can be reviewed by a user.

2.3 Flight controller

The role of the flight controller is critical in an autonomous UAV. In the present study, for Pixhawk UAV, Pixhawk 2.1 with an embedded Ardupilot 3.5 source code was used for control of the autonomous UAV. The Ardupilot 3.5 uses a PID controller with an EKF3 algorithm to remove noise and enhance the estimation of UAV position measurement data from UBS, as shown in Figure 6a. A similar controller to that of the Pixhawk UAV was used for the Bebop2, but it had a vision positioning system (VPS) in the control system, as shown in Figure 6b.



(a) Bebop2 Power with a mobile beacon

(b) Beacon router

Fig. 4. Components of the Bebop2 Power.

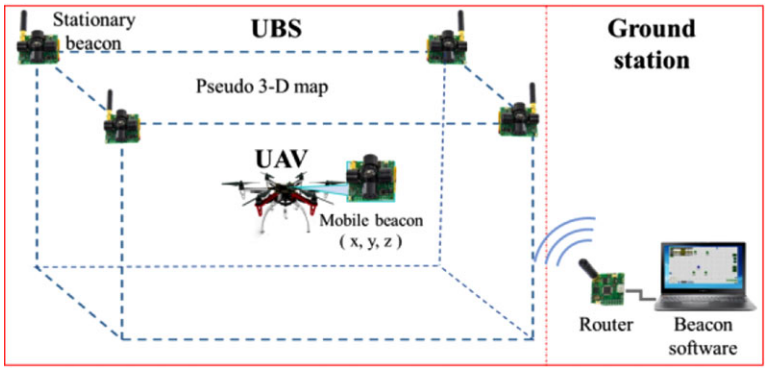


Fig. 5. Relationship between stationary beacons and mobile beacon.

The Pixhawk 2.1 has an embedded IMU composed of three accelerometers, three compasses based on gyroscopes and magnetometers, and two barometers. For autonomous flight of the UAV in GPS-denied locations, information from the three accelerometers and the compass are used as IMU data. The barometer information was not used for indoor flight because the measured data is not accurate due to the nature of indoor operations. The PID controller normally uses GPS input for outdoor flight when GPS is reliable. However, the mobile beacon data replaced the GPS input data in this study to support operations in indoors or GPS-denied areas. The EKF3 algorithm predicts the current location of the UAV based on the mobile beacon position data and IMU data. These parameters can be defined by the Mission Planner using MAVLink.

For the Bebop2 UAV, the same beacon system was used but without a mission planner. To develop autonomous navigation for the Bebop2 UAV, VPS, altimeter, ultrasound, and beacon data were integrated into the Bebop2's existing controller using PID and EKF, as shown in Figure 6b. The inertial navigation system of the Bebop2 consisted of a three-axis gyroscope, an accelerometer, and a magnetometer.

2.4 Beacon-based geo-tagging

To track the location of the UAVs and localize the damage detected during the deep CNN process, a geo-tagging method is used in this study. From these autonomous UAV systems, video data and GPS coordinates collected from UBS are sent to the base station. Time synchronization is an important aspect of geo-tagging. The time steps of the video and beacon systems are synchronized based on the servo information of the UAVs, which provides the start and finish times of the UAV navigations. For convenience, we have extracted image data by seconds. Because UBS is not an actual GPS system, we need to give GPS coordinates as a starting point for UBS settings. The UBS system uses degrees based on latitude (north) and longitude (west). Therefore, the beacon coordinates x and y can be converted following Equation (2) (ArduPilot, 2017).

$$\begin{aligned} V_t &= V_0 - x_t \cdot \cos(rad(\varepsilon)) \cdot \kappa + y_t \cdot \sin(rad(\varepsilon)) \cdot \kappa \\ H_t &= H_0 - x_t \cdot \sin(rad(\varepsilon)) \cdot \lambda + y_t \cdot \cos(rad(\varepsilon)) \cdot \lambda \end{aligned} \quad (2)$$

where V_0 and H_0 are the latitude and longitude at time zero, respectively. t is a time step. ε is the angle difference between the north of the real map and the north of the virtual map. κ and λ are hyperparameters (9.010063270126722e-06, 1.130896616413607e-05).

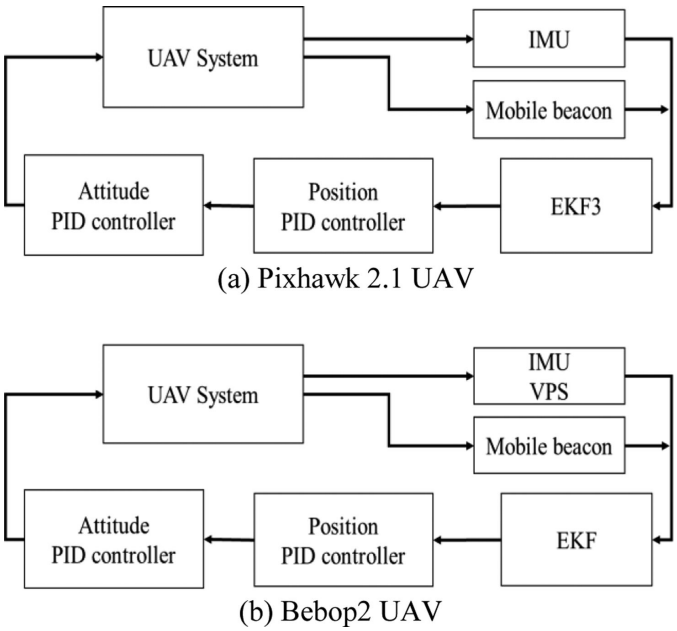


Fig. 6. Flight control systems of the two UAVs.

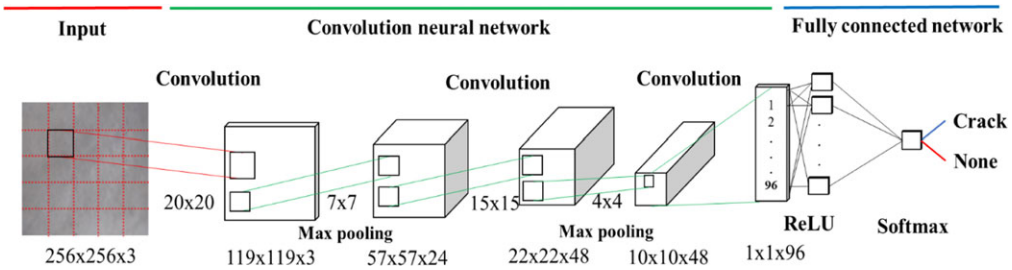


Fig. 7. CNN architecture.

2.5 Crack detection using deep convolutional neural network

To detect structural damage, a deep CNN with the sliding window technique (Cha et al., 2017a) was used to analyze video data collected from the UAV. The sliding window technique uses a predefined size of window to localize the detected damage (Cha et al., 2017a). In the present study, an existing CNN architecture was used. To train the CNN classifier, we used a training data set of raw images of concrete surfaces with a broad range of image variations, including spot lighting and shadows. A Sony FDR-X3000 camera, shown in Figure 3c, was used for the test data set because the payload of UAV is small and cannot carry a digital single lens reflex camera. The prepared training image set fed into a CNN to form a CNN classifier to classify intact and cracked concrete areas. The CNN used in this study, shown in Figure 7, was composed of input, convolution, pooling, activation, and output layers. Auxiliary layers, such as dropout and batch normalization layers, were also used. The details of the CNN are presented by Cha et al. (2017a).

2.5.1 Convolution layer. A convolution layer performs a dot product between a subarray of an input array and a filter. The initial and bias weight values of the filter are randomly generated. Both values are tuned by training that uses a stochastic gradient descent algorithm. The multiplied values are summed, and bias is added to these values. An additional hyperparameter of the layer is the pixel stride, which indicates how many columns and rows (pixels) slide at a time across the input array's width and height.

2.5.2 Pooling layer. The CNN's pooling layer, which reduces the spatial size of an input array to reduce the computation costs, is also important. Max pooling takes the maximum value from

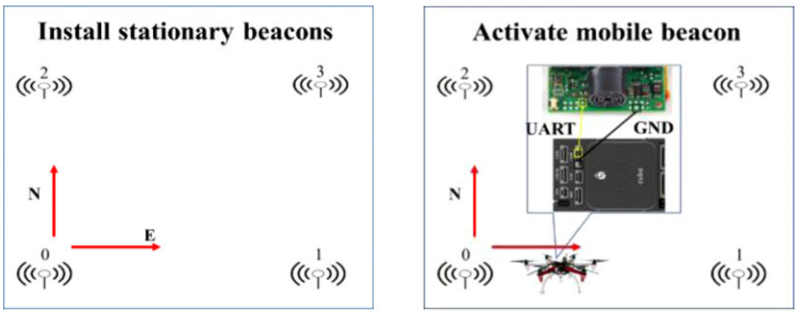


Fig. 8. Set North in the map.

Table 1
Experiment 1 hovering test results

	Latitude (y)	*Longitude (x)*	*Altitude (z mm)*	*Latitude difference*	*Longitude difference*
Test 1	1.37E-05	1.7E-05	1,000	0.0000002 (2.0 cm)	0.000001 (10.0 cm)
Error	1.39E-05	1.8E-05	900		
Test 2	1.4E-05	1.7E-05	1,000	0.0000005 (5.0 cm)	0.0000015 (15.0 cm)
Error	1.4E-05	1.9E-05	863		

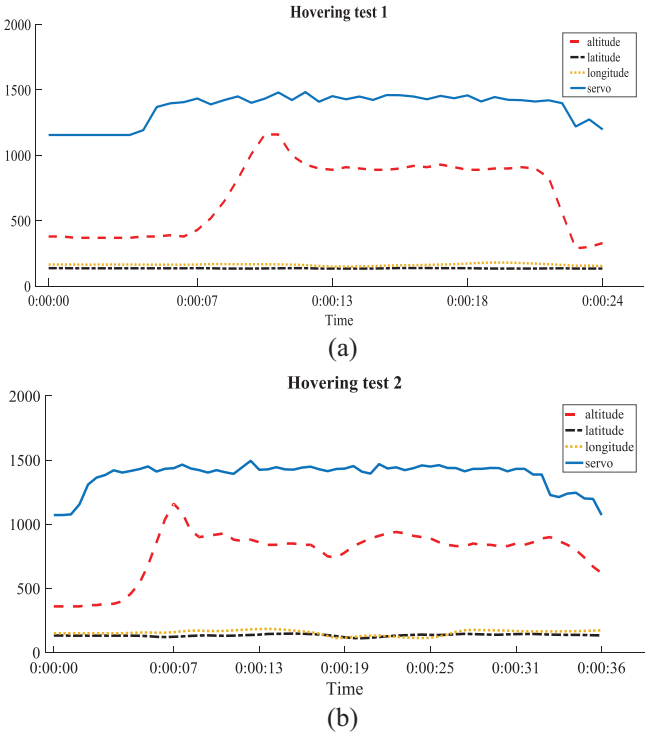


Fig. 9. Hovering tests.

the subarray of the input array, whereas mean pooling takes the mean value. In this article, all pooling layers are used as max pooling because the performance of the max pooling operation is better than that of the other operation (Scherer et al., 2010).

2.5.3 Auxiliary layers. Dropout layers were used to solve the overfitting problem (Srivastava et al., 2014). Training a network with large numbers of neurons often results in overfitting due to the complex connections. The main idea of the dropout technique is to randomly disconnect the connections between neurons of connected layers following the dropout rate. Batch normalization is also used after the first, third, and fifth layers. The batch normalization algorithm is a technique to improve the performance and stability of neural networks (Ioffe and Szegedy, 2015). It normalizes the layer inputs; as a result, this technique facilitates a high learning rate and leads to much faster network convergence.

2.5.4 Activation function. The most typical way to provide nonlinearity in a standard artificial neural network is to use sigmoidal functions. In this study, rectified linear units (ReLU) were chosen because they achieve better accuracy in the CNN (Nair and Hinton, 2010). Compared to other nonlinear functions, the ReLU has no bounded outputs except for its negative input values. Equation (3) represents the ReLU.

$$f = \begin{cases} (x < 0)\, f(x) = 0 \\ (x > 0)\, f(x) = x \end{cases} \tag{3}$$

2.5.5 Softmax layer. To classify final output data, the softmax layer, represented in Equation (4), is the last layer of the CNN architecture. It classifies input data as either intact or cracked. For the input data from the last

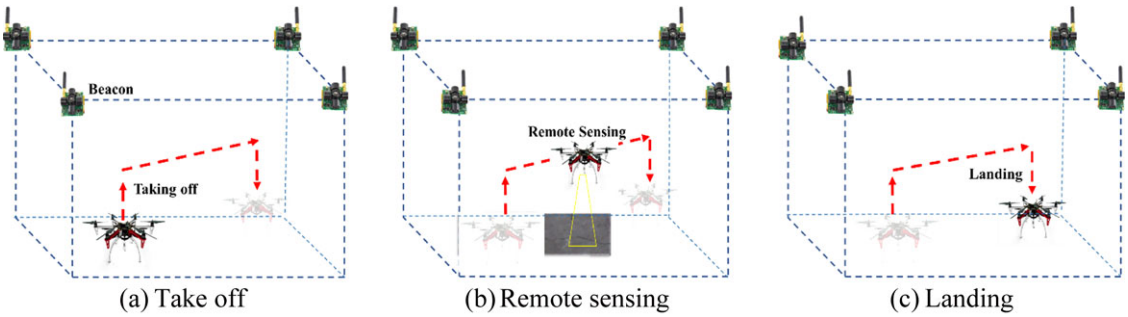


Fig. 10. Experiment scenario.

Table 2
Experiment 2 test results

	Latitude	*Longitude*	*Altitude*
Starting point	50.000003	97.0000207	0
Waypoint	50.0000173	97.0000238	1
Arrived point	50.0000173	97.0000241	0.98
Error	0	0.0000003 (2.72 cm)	0.02 (2 cm)

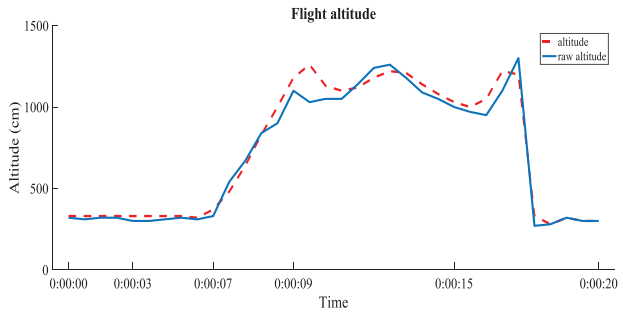


Fig. 11. Altitude time history.

layer through the softmax layer, the range of values is 0 to 1.

$$\sigma(z)j = \frac{e^{z_j}}{\sum_{k=1}^{k} e^{z_k}} \text{for } j = 1, \ldots, K \qquad (4)$$

where K is the number of categories.

2.5.6 Training process. As the initial values of the filter weights in each layer are randomly assigned, the predicted and actual classes do not usually coincide. To calculate the level of deviation between predicted and actual classes, a softmax loss function was used. Logarithmically decreasing learning rates were used to update the gradient descent. As mentioned earlier, we used the pretrained network developed by Cha et al. (2017a) to obtain the advantage of its high damage detection accuracy (98%). We used commercial Matlab software to implement our CNN architecture. However, other frameworks such as Tensorflow and Caffe can be implemented to reproduce this work. For training and validation, the pretrained network used 40,000 images of cracked and intact concrete with 256 × 256 pixel resolution, taken at the Engineering and Information Technology Complex at the University of Manitoba. However, these images were not taken in the same classrooms (E2-229, E2-399) where the autonomous UAV tests were conducted for this article. Further details of the CNN are available in Cha et al. (2017a).

3 CASE STUDIES

To validate the performance of the proposed method, the structural damage detection method using autonomous UAVs with UBS and deep learning was applied to a concrete crack detection problem in the classrooms at the University of Manitoba. The method explained in Section 2 can be extended to use beneath bridge systems or other indoor environments using the experimental procedures described in this section

3.1 Experiment setup

Due to the nature of a complex, autonomous UAV systems, a significant amount of hardware and software had to be designed, configured, and/or integrated. The fundamental procedures for experimental settings were:

Step 1: Preparation of ground station, including installing Mission Planner and beacon software

Step 2: Installation of flight controller firmware in the UAV

Step 3: UAV sensor calibrations

Step 4: Modification of Ardupilot 3.5 reinstallation in the UAV

Step 5: Installation of beacon software and physical installation of mobile beacon in the UAV and stationary beacons on-site, including beacon router to generate a pseudo-3-D beacon map through ground station

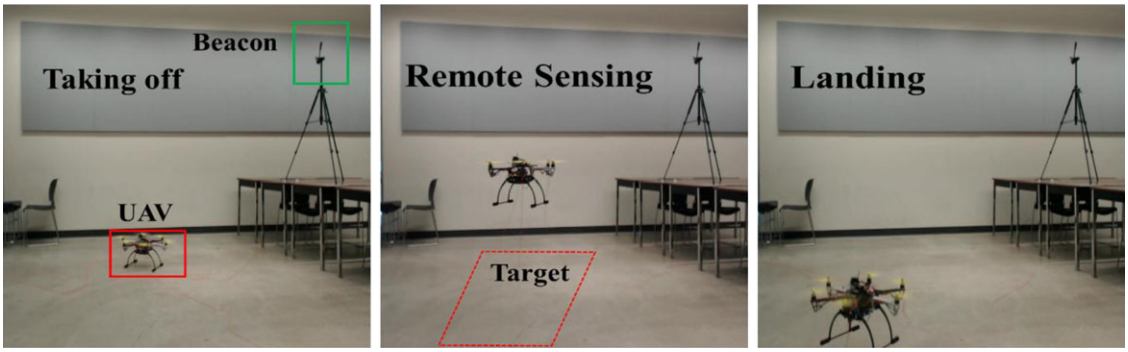


Fig. 12. Test in classroom (E2-399).

Fig. 13. Large conference room (E2-229) for Bebop2 UAV tests.

Table 3
Three experimental test results

Unit (cm)	*Starting point*	*Waypoint1*	*Waypoint2*	*Waypoint3*	*End point*
Mission point1	(170, 675)	(900, 675)	(900, 525)	(170, 525)	(170, 675)
Arrived point1	(170, 675)	(900, 679)	(900, 535)	(169.4, 515)	(171.4, 665.1)
Error	(0, 0)	(0, 4)	(0, 10)	(0.6, 10)	(1.4, 11.9)
Mission point2	(220, 675)	(1,200, 675)	(1,200, 400)	(200, 400)	(200, 675)
Arrived point2	(220, 675)	(1,200, 685)	(1,217, 400)	(190, 375)	(183, 670)
Error	(0, 0)	(0, 10)	(17, 0)	(10, 15)	(7, 5)

Step 6-1: Activation of the Pixhwak UAV, flight path planning using the mission planner (for the Pixhawk UAV), and entry of the mission into the flight controller through MAVLink

Step 6-2: Activation of the Bebop2 UAV, flight path planning, and entry of the mission into the flight controller through MAVLink

Step 7: Rebooting the UAV, including camera

Step 8: Commanding of the start of autonomous flight through the Mission Planner at the ground station

Because the first three steps and Steps 7 and 8 in the above procedures are straightforward and general steps for any UAV flight using the ground station, this section focuses on explaining Steps 4–6 as the key steps for UBS-based autonomous flight for this study. In Step 4, hyperparameters in the flight controllers were redefined from the default setting through the Mission Planner. For example, to activate the EKF3 function, the parameter "Ahrs_EKF_TYPE" was defined as "3" instead of "2," which represents the EKF2 function. EKF3 has many hyperparameters, as shown in Table A1. These parameters should be determined via trial and error based on limitations of the flight controller (Meng et al., 2010). The Step 4 is unnecessary for Bebop2.

In Step 5, UBS firmware was installed using the Dashboard beacon software through the ground station to calibrate the UBS. The calibrated UBS can generate a pseudo-3-D beacon map based on stationary beacon locations. For this study, four stationary beacons (0-3) were used, as shown in Figure 8a. To generate the pseudo-map, the North and East directions must be defined. Default compass 1 in the Ardupilot 3.5 was nullified by deactivating "Tbn_zero_Yaw" from Euler function in the EKF3. The line between beacons 0 and 1 defines east; based on this east direction, the line between beacons 0 and 2 is automatically determined

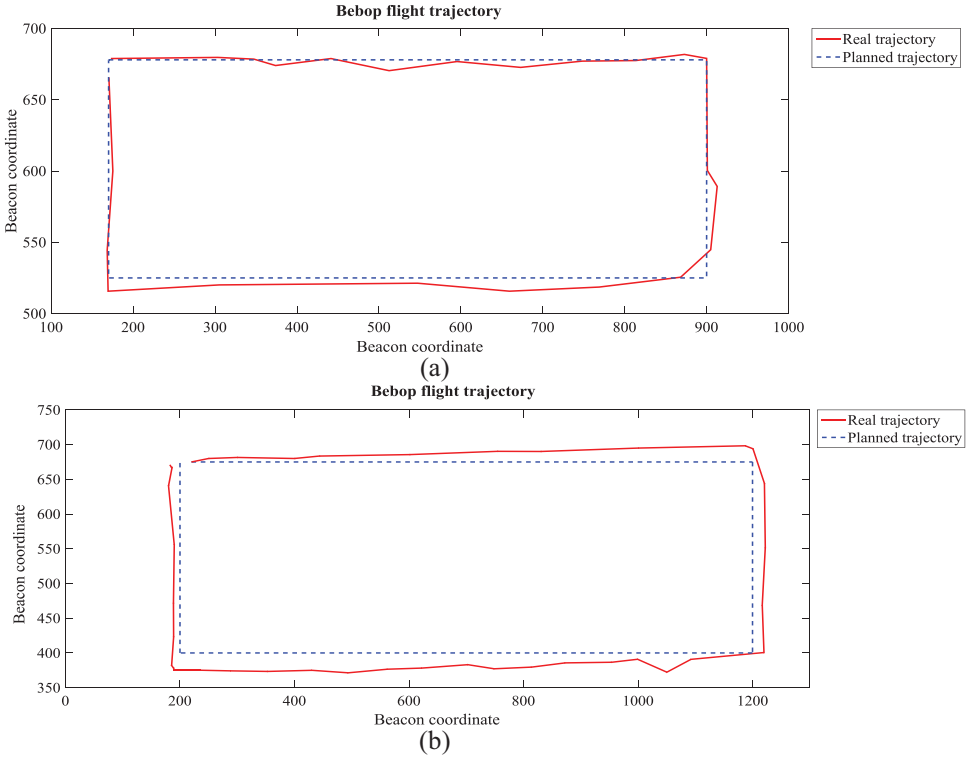


Fig. 14. Trajectories of Bebop2 UAV in E2-229.

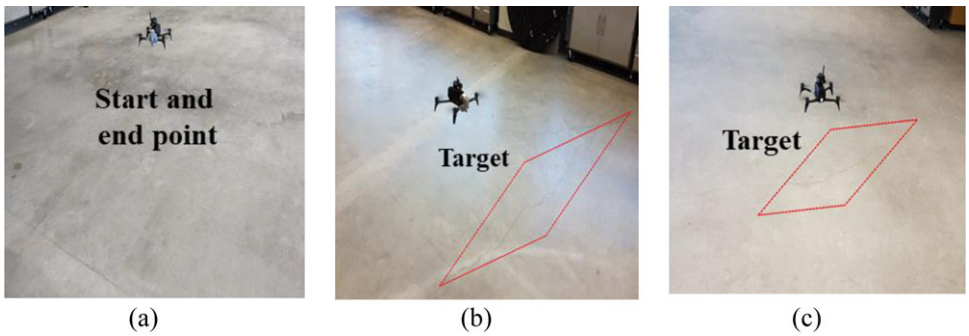


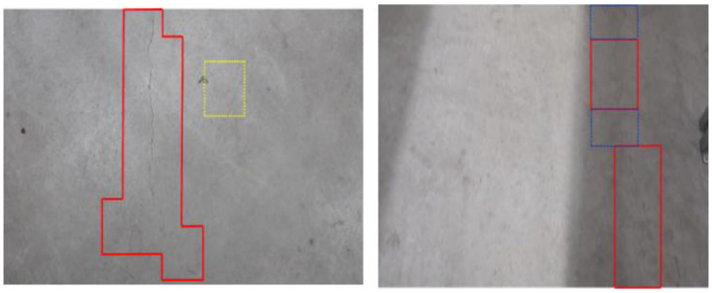
Fig. 15. Test in conference room (E2-229).

to be north. The east direction should be set first for the Marvelmind beacon system.

A data rate of 500 kbps was used in the radio profile of all the beacons for improved position data transmission. The mobile beacon was physically installed in the UAV, and the universal asynchronous receiver/transmitter (UART) and ground (GND) ports of the mobile beacon were connected to the GPS port of the Pixhawk 2.1, as shown in Figure 8b. Next, mobile beacon parameters (i.e., $GPRMC, $GPGGA and $GPVTG) were activated through the Dashboard. Lastly, the UBS router was connected to the ground station.

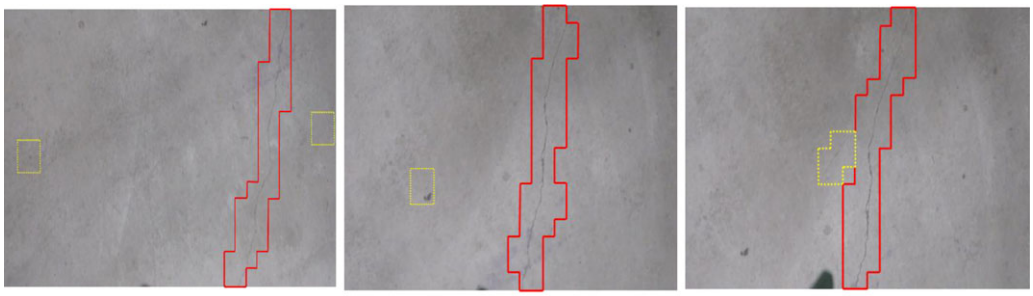
3.2 Experimental tests using Pixhawk UAV

Due to flight restrictions by Transport Canada, field testing of the proposed approach was not possible. As an alternative, two different experiments are conducted in Room E2-399 of the engineering building at the University of Manitoba. There is a concrete crack on the floor visible to the human eye, which was good for validation of the proposed method. The first experiment consisted of a hovering test to validate autonomous navigation of the UAV. In a hovering test, the UAV stays in a virtual circle and attempts to stabilize its position in the area of the circle. The Mission Planner in the ground



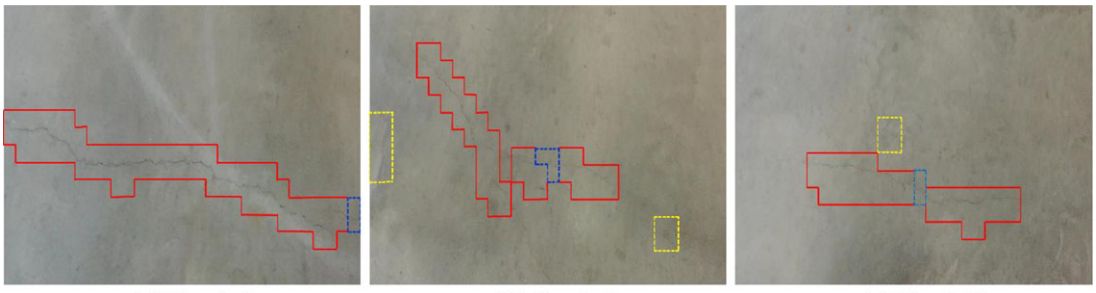
(a) Image from autonomous Pixhawk UAV (b) Image from manual

Fig. 16. Concrete crack detection results from E2-399.



(a) (b) (c)

Fig. 17. Concrete crack detection results from E2-229.



(a) Crack A (b) Crack B (c) Crack C

Fig. 18. Concrete crack detection result.

station was used to command a hovering mode for the UAV. In this test, a 50 cm diameter circle was set as the hovering area based on practices in the UAV discipline (Teuliere et al., 2015; Carvalho et al., 2017).

The flight parameters and results of two hovering tests are presented in Table 1 and Figure 9. The UAV flight was stable, with small errors in latitude, longitude, and altitude. The durations of Tests 1 and 2 were 20 and 35 seconds, respectively. The latitude difference did not exceed 10 cm. The maximum longitude error was 15 cm. These errors are quite acceptable in the UAV discipline, passing our 50 cm tolerance (Teuliere et al., 2015; Carvalho et al., 2017). In Figure 9, the three parameters represent altitude, latitude, and longitude, respectively, measured in mm. Servo, shown on the right-hand side, is a dimensionless variable for pulse width modulation (PWM) control. An ESC determines the proper power to give the motor based on servo which is related to PWM. The overall start and end points of the flight can be monitored based on this line. Based on these hovering tests, the overall setting and performance of the UAV are validated for the practical and complex mission of structural damage detection.

For structural damage detection using this autonomous UAV system, a mission-based flight was assigned through the mission planner in the ground station. This mission required the UAV to take off within 1 m, fly straight North for 1.9 m, and land with video

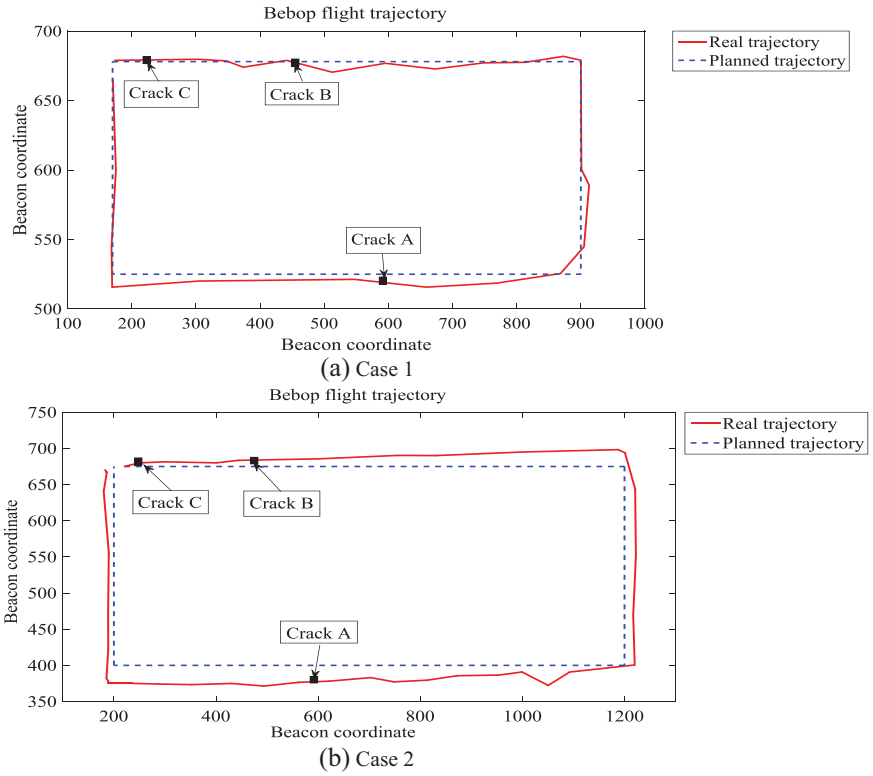


Fig. 19. Geo-tagging results of the Bebop2 autonomous UAV.

data collected with the action camera installed in the bottom of the UAV, as shown in Figure 10 and Table 2. The GPS coordinates can be converted from 1 to 111 km. Based on this scale, the 3×10^7 creates a difference in length of approximately 3 cm. The results in Table 2 indicate that the overall mission was conducted very accurately.

The UAV parameters are plotted in Figure 11 to validate the overall autonomous flight mission of Test 2. Figure 11 shows the time history of UAV altitude. The raw altitude data measured from UBS is plotted as a solid blue line, and predicted altitude data calculated from EKF3 is plotted as a dotted red line. The latitude and longitude time histories are provided in the appendix. Figure A1a shows the raw latitude time history as a red line and predicted time history from EKF3 as a blue line. As described above, the unit of latitude and longitude is degrees (°). Figure A1b shows the time histories of longitude data. The raw and predicted time histories for latitude and longitude data matched well. Using the root mean square error, the first experiment resulted in differences of 7 cm for latitude and 3.1 cm for longitude.

Figure 12 shows the actual flight trajectory of the UAV. The distance from home to the waypoint was 1.9 m, based on the latitudes and longitudes in Table 2. There was approximately a 3 cm longitude error in the waypoint (i.e., 97.0000238-97.0000241). This result is quite accurate compared to previous research using an ultra-wide band beacon system (Zwirello et al., 2012; Tiemann et al., 2015).

3.3 Experimental tests using the Bebop2 UAV

To conduct complex missions over a large area, the Pixhawk UAV is not appropriate due to the physical size of UAV and safety issues. Therefore, the Bebop2 UAV, which is a small UAV suitable for a classroom, can be used for more complex missions. A large conference room (E2-229) with a wide variety of cracks in the floor was selected, as shown in Figure 13. The virtual map size is approximately 10 m × 17 m. The scope of the beacon system can easily be extended by installing additional stationary beacons. In each corner of the virtual map, stationary beacons were installed on the ceiling. The idea was to make a rectangular trajectory in the center of the virtual map to detect cracks in the concrete floor. The trajectory of the Bebop2 UAV is presented in Figure 14. Two different tests were conducted. The actual flight trajectory is presented in Table 3 and plotted as solid red lines in Figure 14. Based on the

planned trajectory, the error is within 20 cm at each waypoint.

The targets (concrete cracks) and positions of the Bebop2 UAV are depicted in Figure 15. Figure 15a shows the start and end waypoints. Figures 15b and c show the targets of the UAV for SHM as an example study.

3.4 Computer vision evaluation

The sets of video data collected by the action camera and Bebop2 UAV camera during the autonomous flight of the missions described in the previous section are presented. Using the collected video data, the CNN-based concrete crack detection method (Cha et al., 2017a) was applied to detect concrete cracks. The original video data were 2,304 × 1,296 pixels but was resized to 2,304 × 1,280 pixels for input to the CNN-based detection method. RGB video data was used instead of grayscale images for the test in this article because our eventual objective is the detection of various types of damage, such as steel cracks, corrosion, and the delamination of steel and concrete members. The results of the CNN-based method using video data collected from the autonomous UAVs are presented in Figure 16a. These results from the autonomous UAV navigation are quite similar to results using the video from the manual action camera, shown in Figure 16b.

The dotted yellow box in Figure 16 indicates a false positive image. The dashed blue box on the image indicates a false negative. To compare the accuracy (α) of the results, we determined the sum of the true positives (Tp) and true negatives (Tn) and divided it by the total number of sliding windows ($Tnsw$) tested, as expressed in Equation (5). To compare different types of accuracy measurements, specificity (β) and sensitivity (γ) were used, as shown in Equations (6) and (7). Fn is the number of false negatives. Fp is the false positives. Here, positive means that the surface is damaged (i.e., a concrete crack), and negative means that the surface is intact.

$$\frac{Tp + Tn}{Tnsw} = \alpha \tag{5}$$

$$\frac{Tp}{Tp + Fn} = \beta \tag{6}$$

$$\frac{Tn}{Tn + Fp} = \gamma \tag{7}$$

Both images exhibited the same accuracy, 97.6%. The accuracy determined by this study agrees well with the previous results of Cha et al. (2017a). For more extensive testing, we used three additional image frames from the video data collected from the autonomous Pixhawk UAV based on UBS. As shown in Figure 17, the concrete cracks were detected well. These results show that autonomous UAV navigation based on UBS is quite promising for SHM. The results also show that autonomous UAV-based monitoring has significant potential for future infrastructure health monitoring.

Based on the above achievements, we carried out more complex missions with longer navigation distances in the conference room (E2-229), as shown in Figure 14. The detected concrete cracks are presented in Figure 18. Based on the results obtained from the autonomous flight of the Bebop2 UAV, the accuracy was 96.6%, the sensitivity was 91.9%, and the specificity was 97.9%. All of these detected cracks were localized by the geo-tagging method described in Section 2.4. The localized damage information was plotted in Figure 19 for the two autonomous navigation cases.

There are possible limitations in this work. The flight controller (ArduPilot 3.5) works by trial and error to tune its parameters. We also used a cheap frame and batter, which can reduce the flight time of the UAV. The manufacturer suggests that the Pixhawk UAV can fly for 15 minutes, but it does not guarantee the actual flight time. In most cases of normal use, only 70% of this projected flight time is achieved. The potential payload is 2 kg, but the actual payload is around 80% of this Figure. Even though video data can be transferred in real time from a UAV to a base station, we did not pursue real-time processing for the CNN analysis because the CNN requires approximately 6 seconds to analyze one frame of image with the current pixel resolution. However, deep learning is an emerging area of research, and we expect that, in the near future, real-time processing will become feasible. The first limitation of UBS is that it requires an installation before data acquisition can be performed. However, modern SHM still needs sensors to be installed for data acquisition under the bridge. The second limitation of UBS is that the coverage area of a set beacon system is 30 m × 30 m, meaning that the beacon system is particularly well suited for short-distance applications (Perez-Grau et al., 2017). It is possible to increase this coverage by installing additional stationary beacons. The third limitation of UBS is that ultrasonic signals cannot penetrate walls or other obstacles, but this is similar to any other GPS, beacon, or Vicon system. However, a virtual map can be easily expanded by installing more stationary beacons.

4 CONCLUSION

This article proposed an autonomous UAV-based damage detection method using an UBS for indoor

environments and areas in which GPS is denied or unreliable. Based on our extensive literature review, there are no published papers that propose autonomous navigation methods in GPS-denied infrastructure areas for structural health monitoring. The main contributions of this article are as follows: (1) it is the first application of ultrasonic beacon for UAV navigation for a GPS-denied environment, such as indoors and beneath a bridge (which is a critical area that should be monitored) or indoors for SHM, (2) we examined the possibility of using the video data collected from the UAV for deep learning-based automatic damage detection (Video data collected from the UAV has vibration issues, including the jello effect. Until now, little research has been done to detect structural damage using video data collected from a UAV with deep learning. Our previous deep learning-based damage detection (Cha et al., 2017a) employed only hand-held camera data that had no vibration issues.), (3) the detected damage was localized using the geo-tagging method, and (4) all these advanced technologies were integrated to realize autonomous UAV-based damage detection for a GPS-denied environment.

To realize this proposed method, we conducted the following: (1) fabricated a UAV using various parts that are commercially available, such as a frame, a flight controller, a telemeter, and an action camera, instead of using a premanufactured UAV, for which it is not generally possible to modify the source code for autonomous navigation; (2) modified the source code of the flight controller firmware; (3) integrated an UBS with the autonomous flight controller; (4) replaced the GPS coordinates with a UBS signal in the image metadata for geotagging; and (5) adopted a small, commercially available Bebop2 UAV for a more complex mission with longer distance navigation, integrating the UBS and modifying the source codes to control its flight.

To demonstrate the proposed approach, three different indoor tests were conducted: (1) a hovering test to validate autonomous mission flight, (2) a waypoint-based mission test with a specific waypoint, and (3) a complex mission with long-distance navigation using the commercial Bebop2 UAV with geo-tagging for damage localization. The autonomous flight of the UAV was successfully validated based on flight log data from these tests. The overall flight was quite accurate, but there were some fluctuations in altitude.

As the final objective of this autonomous UAV, a concrete crack was detected with high accuracy (96.6%), sensitivity (91.9%), and specificity (97.9%) using video data collected from the autonomous UAV. This CNN performance was well matched to the author's previous results using deep CNN-based damage detection (Cha et al., 2017a). The results of the image collected from the UAV were also compared to the results of manual image collection to validate the potential of the autonomous UAV-based health monitoring of infrastructure. The results from both images agreed well with high accuracy. As a future study, an additional obstacle avoidance sensor will be installed in our UAV to avoid the obstacle, and a more complex autonomous mission will be conducted for the SHM application. Real-world application will be carried out in the future to examine environmental effects, such as temperature and wind.

REFERENCES

Abdel-Qader, I., Abudayyeh, O. & Kelly, M. E. (2003), Analysis of edge-detection techniques for crack identification in bridges, *Journal of Computing in Civil Engineering*, **17**(4), 255–63.

Ardupilot (2017), Available at: https://github.com/ArduPilot/ardupilot.git, accessed March 2017.

Carvalho, J. P., Jucá, M. A., Menezes, A., Olivi, L. R., Marcato, A. L. M. & dos Santos, A. B. (2017), Autonomous UAV outdoor flight controlled by an embedded system using Odroid and ROS, in *CONTROLO 2016*, Springer, Cham, 423–37.

Cha, Y. J., Choi, W. & Büyüköztürk, O. (2017a), Deep learning-based crack damage detection using convolutional neural networks, *Computer-Aided Civil and Infrastructure Engineering*, **32**(5), 361–78.

Cha, Y. J., Choi, W., Suh, G., Mahmoudkhani, S. & Büyüköztürk, O. (2017b), Autonomous structural visual inspection using region-based deep learning for detecting multiple damage types, *Computer Aided Civil and Infrastructure Engineering*, https://doi.org/10.1111/mice.12334.

Cha, Y. J., You, K. & Choi, W. (2016), Vision-based detection of loosened bolts using the Hough transform and support vector machines, *Automation in Construction*, **71**, 181–88.

Chen, S.-E., Rice, C., Boyle, C. & Hauser, E. (2011), Small-format aerial photography for highway-bridge monitoring, *Journal of Performance of Constructed Facilities*, **25**(2), 105–12.

Chen, J. G., Wadhwa, N., Cha, Y. J., Durand, F., Freeman, W. T. & Buyukozturk, O. (2015), Modal identification of simple structures with high-speed video using motion magnification, *Journal of Sound and Vibration*, **345**, 58–71.

Deutscher, J., Blake, A. & Reid, I. (2000), Articulated body motion capture by annealed particle filtering, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, **2**, 126–33.

Díaz, E., Pérez, M. C., Gualda, D., Villadangos, J. M., Ureña, J. & García, J. J. (2017), Ultrasonic indoor positioning for smart environments: a mobile application, in *Experiment@International Conference*, Faro, Portugal, IEEE, 280–85.

Eschmann, C., Kuo, C. M., Kuo, C. H. & Boller, C. (2012), Unmanned aircraft systems for remote building inspection and monitoring, in *Proceedings of the 6th European Workshop on Structural Health Monitoring*, Dresden, Germany, 1179–86.

Gillins, M. N., Gillins, D. T. & Parrish, C. (2016), Cost-effective bridge safety inspections using unmanned aircraft

systems (UAS), in *Proceedings of the Geotechnical and Structural Engineering Congress*, 1931–40.
Hallermann, N. & Morgenthal, G. (2013), Unmanned aerial vehicles (UAV) for the assessment of existing structures, *International Association for Bridge and Structural Engineering*, IABSE Symposium Report, **101**(14), 1–8.
Hess, W., Kohler, D., Rapp, H. & Andor, D. (2016), Real-time loop closure in 2D LIDAR SLAM, in *Proceedings of the 2016 IEEE International Conference on Robotics and Automation (ICRA)*, 1271–78.
Honegger, D., Meier, L., Tanskanen, P. & Pollefeys, M. (2013), An open source and open hardware embedded metric optical flow CMOS camera for indoor and outdoor applications, in *Proceedings of the 2013 IEEE International Conference on Robotics and Automation (ICRA)*, 1736–41.
Ioffe, S. & Szegedy, C. (2015), Batch normalization: accelerating deep network training by reducing internal covariate shift, in *Proceedings of the International Conference on Machine Learning*, 448–56.
Koziarski, M. & Cyganek, B. (2017), Image recognition with deep neural networks in presence of noise—dealing with and taking advantage of distortions, *Integrated Computer-Aided Engineering*, **24**(4), 337–50.
Lee, J. J. & Shinozuka, M. (2006), A vision-based system for remote sensing of bridge displacement, *NDT & E International*, **39**(5), 425–31.
Lim, H., Park, J., Lee, D. & Kim, H. J. (2012), Build your own quadrotor: open-source projects on unmanned aerial vehicles, *IEEE Robotics & Automation Magazine*, **19**(3), 33–45.
Lin, Y. Z., Nie, Z. H. & Ma, H. W. (2017), Structural damage detection with automatic feature-extraction through deep learning, *Computer-Aided Civil and Infrastructure Engineering*, **32**(12), 1025–46.
Marvelmind Robotics (2017), Available at: https://marvelmind.com, accessed March 2017.
Meier, L., Tanskanen, P., Fraundorfer, F. & Pollefeys, M. (2011), Pixhawk: a system for autonomous flight using onboard computer vision, in *Proceedings of the 2011 IEEE International Conference on Robotics and Automation (ICRA)*, 2992–97.
Meng, L., Li, L. & Veres, S. M. (2010), Aerodynamic parameter estimation of an unmanned aerial vehicle based on extended Kalman filter and its higher order approach, in *Proceedings of the 2010 2nd International Conference on Advanced Computer Control (ICACC)*, **5**, 526–31.
Metni, N. & Hamel, T. (2007), A UAV for bridge inspection: visual servoing control law with orientation limits, *Automation in Construction*, **17**(1), 3–10.
Minorsky, N. (1922), Directional stability of automatically steered bodies, *Naval Engineers Journal*, **32**(2), 280–309.
Nair, V. & Hinton, G. E. (2010), Rectified linear units improve restricted Boltzmann machines, in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 807–14.
NHTSA & SAE International (2014), Available at: https://www.sae.org/misc/pdfs/automated_driving.pdf, accessed March 2017.
Orsag, M., Korpela, C. & Oh, P. (2013), Modeling and control of MM-UAV: mobile manipulating unmanned aerial vehicle, *Journal of Intelligent & Robotic Systems*, **69**, 1–14.
Ortega-Zamorano, F., Jerez, J. M., Gómez, I. & Franco, L. (2017), Layer multiplexing FPGA implementation for deep back-propagation learning, *Integrated Computer-Aided Engineering*, **24**(2), 171–85.
Perez-Grau, F. J., Caballero, F., Merino, L. & Viguria, A. (2017), Multi-modal mapping and localization of unmanned aerial robots based on ultra-wideband and RGB-D sensing, in *International Conference on Intelligent Robots and Systems (IROS 2017)*, IEEE/RSJ, Vancouver, Canada, September 2017, 3495–502.
Proficnc (2017), Pixhawk v2 feature overview. Available at: www.proficnc.com, accessed March 2017.
Rafiei, M. H. & Adeli, H. (2017), A novel machine learning based algorithm to detect damage in highrise building structures, *The Structural Design of Tall and Special Buildings*, **26**, 18, https://doi.org/10.1002/tal.1400.
Rafiei, M. H., Khushefati, W. H., Demirboga, R. & Adeli, H. (2017), Supervised deep restricted Boltzmann machine for estimation of concrete compressive strength, *ACI Materials Journal*, **114**(2), 237244.
Ren, S., He, K., Girshick, R. & Sun, J. (2015), Faster R-CNN: towards real-time object detection with region proposal networks, in *Proceedings of the Advances in Neural Information Processing Systems*, 91–99.
Sankarasrinivasan, S., Balasubramanian, E., Karthik, K., Chandrasekar, U. & Gupta, R. (2015), Health monitoring of civil structures with integrated UAV and image processing system, *Procedia Computer Science*, **54**, 508–15.
Scherer, D., Müller, A. & Behnke, S. (2010), Evaluation of pooling operations in convolutional architectures for object recognition, in *International Conference on Artificial Neural Networks*, Springer, Berlin, Heidelberg, September 2010, 92–101.
Smith, G. L., Schmidt, S. F. & McGee, L. A. (1962), *Application of Statistical Filter Theory to the Optimal Estimation of Position and Velocity On-Board a Circumlunar Vehicle*, Technical Report, TR R-135, National Aeronautics and Space Administration.
Soukup, D. & Huber-Mörk, R. (2014), Convolutional neural networks for steel surface defect detection from photometric stereo images, in *Proceedings of the International Symposium on Visual Computing*, Springer International Publishing, 668–77.
Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. (2014), Dropout: a simple way to prevent neural networks from overfitting, *Journal of Machine Learning Research*, **15**(1), 1929–58.
Stempfhuber, W. & Buchholz, M. (2011), A precise, low-cost RTK GNSS system for UAV applications, in *Proceedings of the Conference on Unmanned Aerial Vehicle in Geomatics*, Zürich, 289–93.
Sung, Y., Kwak, J., Jeong, Y. S. & Park, J. H. (2016), Beacon distance measurement method in indoor ubiquitous computing environment, in *Proceedings of the Advances in Parallel and Distributed Computing and Ubiquitous Services*, Springer, Singapore, 125–30.
Teuliere, C., Marchand, E. & Eck, L. (2015), 3-D model-based tracking for UAV indoor localization, *IEEE Transactions on Cybernetics*, **45**(5), 869–79.
Tiemann, J., Schweikowski, F. & Wietfeld, C. (2015), Design of an UWB indoor-positioning system for UAV navigation in GNSS-denied environments, in *Proceedings of the 2015 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, 1–7.
Vossiek, M., Wiebking, L., Gulden, P., Weighardt, J. & Hoffmann, C. (2003), Wireless local positioning-concepts, solutions, applications, in *Proceedings of the Radio and Wireless Conference*, 219–24.

Zhang, C. & Elaksher, A. (2012), An unmanned aerial vehicle-based imaging system for 3D measurement of unpaved road surface distresses, *Computer-Aided Civil and Infrastructure Engineering*, **27**(2), 118–29.

Zhang, C., Kuhn, M., Merkl, B., Mahfouz, M. & Fathy, A. E. (2006), Development of an UWB indoor 3D positioning radar with millimeter accuracy, in *Microwave Symposium Digest*, IEEE MTT-S International, 106–09.

Zwirello, L., Schipper, T., Harter, M. & Zwick, T. (2012), UWB localization system for indoor applications: concept, realization and analysis, *Journal of Electrical and Computer Engineering*, **4**, 1–11.

APPENDIX

Table A1
Hyperparameters of the flight controller (Pixhawk 2.1)

	Parameter	*Value*		*Parameter*	*Value*
1	Ahrs_EKF_TYPE	3	26	EK3_IMU_MASK	3
2	GPS_TYPE	5	27	EK3_LOG_MASK	1
3	EK3_ENABLE	1	28	EK3_MAG_CAL	2
4	EK3_GPS_TYPE	1	29	EK3_MAG_I_GATE	300
5	EK3_ABIAS_P_NSE	0.003	30	EK3_MAG_M_NSE	0.05
6	EK3_ACC_BIAS_LIM	1	31	EK3_MAG_MASK	1
7	EK3_ACC_P_NSE	0.35	32	EK3_MAGB_P_NSE	0.0001
8	EK3_ALT_M_NSE	2	33	EK3_MAGE_P_NSE	0.001
9	EK3_ALT_SOURCE	0	34	EK3_MAX_FLOW	2.5
10	EK3_BCN_DELAY	50	35	EK3_NOAID_M_NSE	50
11	EK3_BCN_I_GTE	500	36	EK3_POS_I_GATE	500
12	EK3_BCN_M_NSE	1	37	EK3_POSNE_M_NSE	0.1
13	EK3_CHECK_SCALE	50	38	EK3_RNG_I_GATE	500
14	EK3_EAS_I_GATE	400	39	EK3_RNG_M_NSE	10
15	EK3_EAS_M_NSE	1.4	40	EK3_RNG_USE_HGT	−1
16	EK3_ENABLE	1	41	EK3_RNG_USE_SPD	2
17	EK3_FLOW_DELAY	10	42	EK3_TAU_OUTPUT	25
18	EK3_FLOW_I_GATE	300	43	EK3_VEL_I_GATE	500
19	EK3_FLOW_M_NSE	0.25	44	EK3_VELD_M_NSE	0.5
20	EK3_GBIAS_P_NSE	0.001	45	EK3_VELNE_M_NSE	3
21	EK3_GLITCH_RAD	25	46	EK3_WIND_P_NSE	0.1
22	EK3_GPS_CHECK	31	47	EK3_WIND_PSCALE	0.5
23	EK3_GYRO_P_NSE	0.05	48	EK3_YAW_I_GATE	500
24	EK3_HGT_DELAY	0	49	EK3_YAW_M_NSE	0.1
25	EK3_HGT_I_GATE	500			

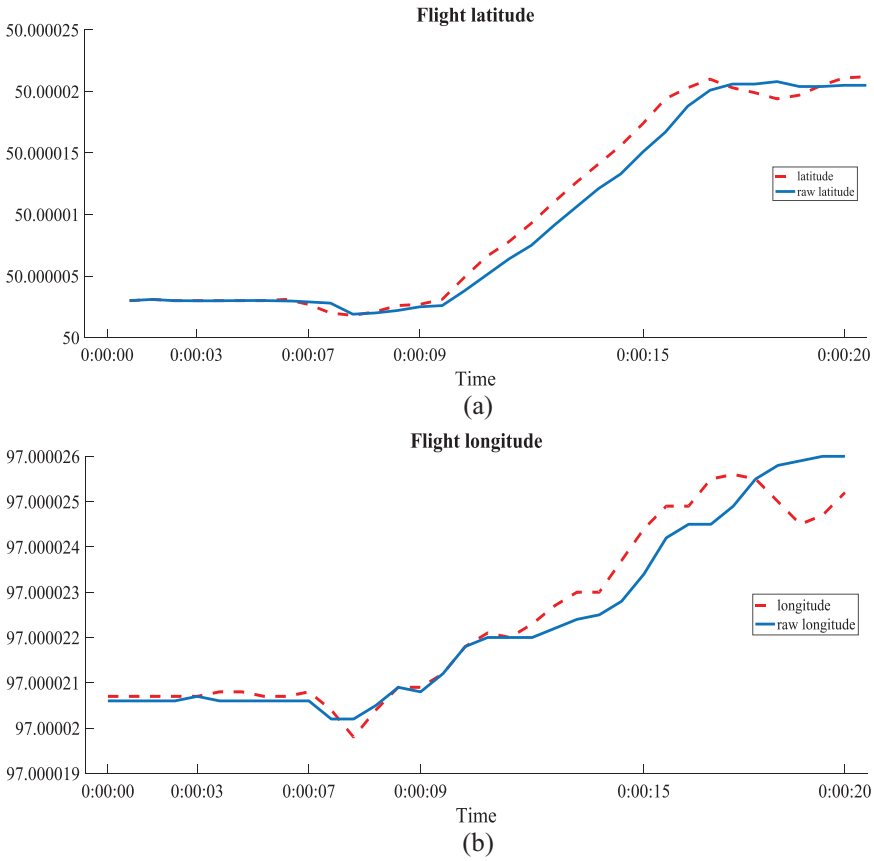


Fig. A1. Beacon latitude and longitude.