AUTOMATED IMAGE LOCALIZATION AND DAMAGE LEVEL EVALUATION FOR RAPID POST-EVENT BUILDING ASSESSMENT

by

Xiaoyu Liu

A Dissertation

Submitted to the Faculty of Purdue University

In Partial Fulfillment of the Requirements for the degree of

Doctor of Philosophy

School of Mechanical Engineering

West Lafayette, Indiana

December 2022

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF COMMITTEE APPROVAL

Dr. Shirley J. Dyke, Chair
School of Mechanical Engineering

Dr. David Cappelleri
School of Mechanical Engineering

Dr. Juan P. Wachs
School of Industrial Engineering

Dr. Song Zhang
School of Mechanical Engineering

Approved by:
Dr. Nicole Key

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to my advisor Prof. Shirley J. Dyke. I would like to thank her for giving the guidance not only about knowledge, research also about how to be a better person. I would like to thank her for giving me the opportunity to "have come a long way" to become who I am right now. I would like to thank her for supporting me, on and off research, without her, I would not have what I have right now.

I also would like to thank my committee members Prof. David Cappelleri, Prof. Juan P. Wachs and Prof. Song Zhang. I would like to thank the chances that I am given to learn from them. I would like to thank the insights that they give to my research, and their valuable time and effort to review my dissertation.

I also would like to thank Prof. Ilias Bilionis and Prof. Julio Alfonso Ramirez for their valuable guidance on my research. I would also like to thank Prof. Thomas J. Hacker and Prof. Bedrich Benes for the knowledge I learnt from them through research collaboration. I also would like to thank Prof. Chuck Krousgrill for tutoring me how to be a better teaching assistant.

I also would like to thank all the team members I have met in IISL, Purdue and Rethi, Purdue. I would like to thank every alumnus in the computer vision group, including Dr. Chul Min Yeum, Dr. Jongseong Choi, Dr. Ali Lenjani and the current team members Xin Zhang, Lissette Iturburu, Jean Kwannandar and Benjamin Eric Wogen. I would like to thank the CDCM group from Rethi, including Dr. Amir Behjat, Roman Ibrahimov, and all the other members. I would like to thank the project members in CDSE and CSSI, Zhiwei Chu and other members.

I also would like to thank my family for their faith and support in me for all the years.

At last, I would like to acknowledge the financial support from multiple sources, including National Science Foundation under Grant No. NSF 1608762 and Grant No. NSF-OAC-1835473, Joint Transportation Research Program administered by the Indiana Department of Transportation and Purdue University (projectSPR-4222), and NASA under grant or cooperative agreement award number 80NSSC19K1076.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Image data remains an important tool for post-event building assessment and documentation. After each natural hazard event, significant efforts are made by teams of engineers to visit the affected regions and collect useful image data. In general, a global positioning system (GPS) can provide useful spatial information for localizing image data. However, it is challenging to collect such information when images are captured in places where GPS signals are weak or interrupted, such as the indoor spaces of buildings. An inability to document the images' locations would hinder the analysis, organization, and documentation of these images as they lack sufficient spatial context. This problem becomes more urgent to solve for the inspection mission covering a large area, like a community. To address this issue, the objective of this research is to generate a tool to automatically process the image data collected during such a mission and provide the location of each image. Towards this goal, the following tasks are performed. First, I develop a methodology to localize images and link them to locations on a structural drawing (Task 1). Second, this methodology is extended to be able to process data collected from a large scale area, and perform indoor localization for images collected on each of the indoor floors of each individual building (Task 2). Third, I develop an automated technique to render the damage condition decision of buildings by fusing the image data collected within (Task 3). The methods developed through each task have been evaluated with data collected from real world buildings. This research may also lead to automated assessment of buildings over a large scale area.

1. INTRODUCTION

Engineers often learn from observing the consequences of natural disasters on our physical infrastructure by studying the real world. A large amount of data is collected after each hazard event. Among the various types of data being collected, image data offer the most direct and useful way to record the impact of these events on our physical infrastructure. By utilizing inexpensive cameras or cell phones, engineers and researchers can rapidly capture and document damage or failures in a building such as spalling, shear cracks, deformation, etc. Although these images are clearly useful to the researchers that collected the specific data, labeling and organizing that data to make them accessible to other researchers is quite time-consuming. Thus, a large fraction of the data often go unused. The rapid organization, analysis, and publication of these data is a valuable activity for the hazards community.

In the US, the National Science Foundation supports several research facilities to collect and store reconnaissance data. The Natural Hazards Engineering Research Infrastructure (NHERI) is a shared-use facility developed to support natural hazards engineering research. Two components of NHERI, the Rapid Response Research (RAPID) Facility and DesignSafe-CI, serve in this capacity. RAPID supports field data collection, and DesignSafe-CI is a data repository for storing, publishing and sharing. Around the world other organizations maintain data repositories with similar goals. These include the Earthquake Engineering Research Institute, DataCenterHub, Pacific Earthquake Engineering Research Center, and QuakeCore [1-4]. However, these platforms do not offer functionalities to support researchers in sorting, classifying, organizing, and analyzing these data. Other platforms have been developed, e.g., Automated Reconnaissance Image Organizer (ARIO), which are designed to provide automated image classification and report generation services [5].

Despite the investments made in acquiring and storing these data, the current suite of data repositories may not be adequate for conducting in-depth research with images. Inherently, image data lack spatial context. When 3D scenes are captured with 2D images, the relative locations between the scenes on the images are needed to understand spatial relationships from 2D images [5-7]. To capture visual details, field engineers may take photos close to the object being documented within the scenes-of-interest. However, with such images, one cannot easily infer any of the relevant spatial contexts to make use of the information extracted. For example, assume that

an engineer takes pictures of a damaged column of the building from a close distance. The images may not contain contextual information associated with, for instance, the column location on the floor, its relative size, or the relative conditions of other nearby building components. To obtain such information the engineer will need to sift through the set of previous or next pictures collected near this region. However, this is challenging, especially when one considers that a building is likely to have multiple components that are built with a common style or appearance [4]. In other words, many columns or walls in a given building do look similar to each other. In this circumstance, using images without contextual information will inevitably lead to untrustworthy results in practice, and will certainly result in needless consumption of time and manual labor involved in inferring spatial information. GPS metadata on images can provide approximate geospatial information, but only for those data captured outside the building or in open spaces without interference. Additionally, 3D sensors like Light Detection and Ranging (LiDAR) can be used for reconstructing the scenes in 3D for image localization, but they are expensive, and at this time require considerable extra time and effort to use [8,9].

Once the image is collected and the location is known, lessons about individual buildings can be learned. Lessons and new lines of inquiry are normally identified by observing and classifying the damage in buildings. The most common and critical damage to get is the classification of the overall condition of the building in terms of reinforced concrete (RC) components (structural members) and masonry (M) components (non-structural members). To automatically extract the relevant information from visual data saves valuable human time to browse through the data and interpret them. This option is made possible by the recent success of understanding the visual contents of single images using convolutional neural networks. However, knowledge learned from the visual data are highly constrained to single images. To make an overall evaluation decision about a building of its structural and non-structural members mostly depends on images that contain an overview of the entire building. Images that provide the detailed conditions of the building can be captured but can have little influence in such decision-making processes, unless experts spend large amounts of time to study the images collected and come to some conclusion. Inevitably, this stalls the recovery of a community from critical events.

1.1 Overview of the Tool Developed in This Research

To address the above issues, the objective of this research is to develop a systematic tool that supports automated visual data localization without adding extra burden to the engineering work, and generate damage level evaluation of buildings based on a typical set of reconnaissance images collected from a single building in the field. The research focuses on providing location information in a typical post-event environment. The goal is to provide an approach that requires little need for special requirements or equipment, and no significant amount of time to set up before collecting the data. Engineers would only need to use a camera as a data collection tool, without engaging other additional sensors. Then, based on the location information, damage level evaluations are made for regions and overall condition of buildings, and possibly communities. The research is carried out through the following tasks:

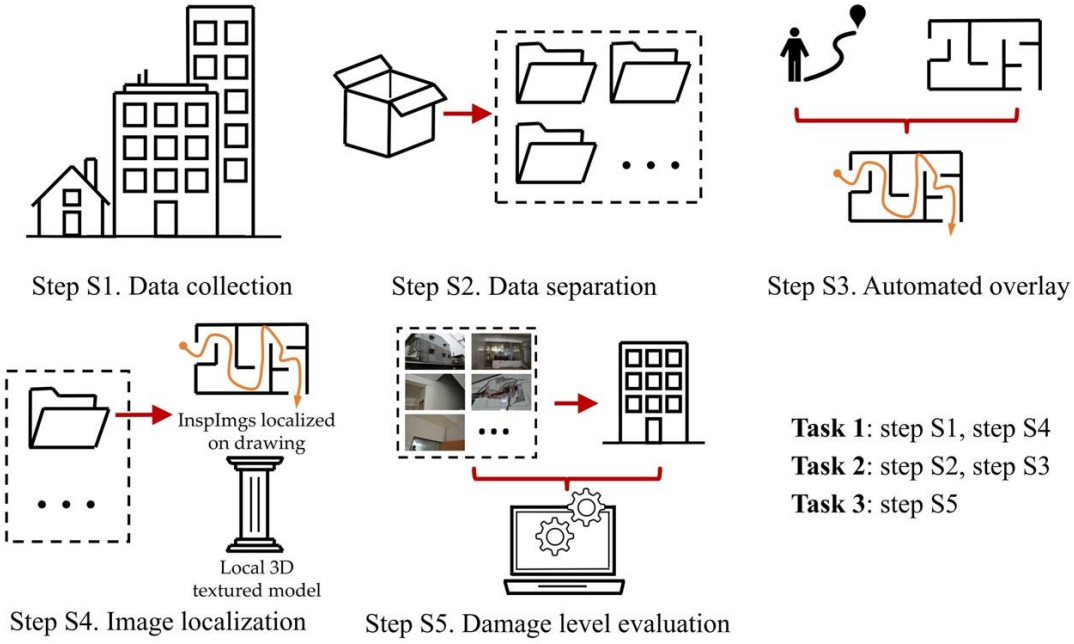


Figure 1.1. Overview of the tool.

Step S1 is data collection. This step is to develop guidance for engineers to collect data for this tool. The input data include three types of data: 1) video footage (hereafter, PathVideo) to record the scenes right in front of the data collector as they walk through the building; 2) visual data, or inspection images (hereafter, InspImgs) that are collected to document the consequences of natural hazards on the buildings; and, 3) digital images of the structural drawings (hereafter,

SDI) of buildings visited during the mission. PathVideo can readily be gathered along the path taken through a building using a motion camera. Meanwhile, a parallel set of InspImgs that are collected for building assessment is linked to the PathVideo using time information. By developing the following steps of this tool, the efforts required in the data collection step is minimized to the least amount. Engineers only need to wear a motion camera with a supporting gimbal on their body. Besides this, engineers just collect InspImgs with another DSLR camera at their will. This step is completed in Task 1. For convenience, the abbreviations used herein are defined in Table 1.1.

Table 1.1. Abbreviation table

Abbreviation	Definition
InspImgs	inspection images
Path	indoor 3D path that data collector takes
PathVideo	video footage recorded with motion camera
PathImgs	frames of PathVideo
Pcl	point cloud model
PathPcl	Path and Pcl
SDI	digital image of structural drawing

Step S2 is data separation. The data collected in Step S1 is automatically separated, so they can be processed and visualized in the following steps. First, the frames of PathVideo (PathImgs) are classified and labeled as an indoor or outdoor image by put through a convolutional neural network classifier. After clustering the probability of each image, the whole collection of data is separated to indoor groups and outdoor groups. Indoor groups representing images in one building are further separated based on the height information from the indoor 3D path (hereafter, Path), which is reconstructed with a visual odometry technique. Now PathVideo is separated to individual floors. InspImgs are related to them by comparing timestamp. Digital images of structural drawing (hereafter, SDI) are simply separated by their file names. Then, PathImgs, InspImgs and SDI are put into the corresponding folder of individual floors of different buildings. This step is performed in Task 2.

Step S3 is automated overlay. When processing the separated reconnaissance data in each folder, Path needs to be overlaid onto the structural drawing. I form an optimization problem and design a highly reliable and fast search method to automatically perform this process. A cost function is defined using information of black pixels and white pixels in SDI, standing for visible

components and open space in the building. Then a search method is developed based on particle swarm optimization algorithm. The method is combined with several tactics, as multiple time initialization, coarse-to-fine search using image pyramids. This task is completed in Task 2.

Step S4 is image localization. The aim of this step is to visualize the location of InspImgs. After successfully overlay of Path onto the structural drawing, PathImgs can be located. InspImgs are then linked to PathImgs or PathVideo using time stamp matching in Step S2, data separation. Thus, InspImgs are also located on the structural drawing along Path. Additionally, this tool is also able to provide local 3D textured model for a nearby environment of each InspImg. This approach will provide clear visual context and spatial information for InspImgs. This step is meant to help us understand and study inspection data. This task is completed in Task 1.

Step S5 is damage condition evaluation of buildings. This step is to jointly analyze InspImgs collected inside a building (outcome of Step S1-S4) and generate the overall damage condition of the building. CNN based image classifiers are designed to extract initial damage information from each image collected in this building. The damage information is fused engaging a naïve Bayesian fusion method. Following a series of automated process, including classifying images, results filtering, probability sampling, probability fusion, the evaluation for a building is given by its health condition in reinforcement concrete and masonry. This step is carried out in Task 3.

1.2 Remainder of the Dissertation

The remainder of this dissertation is organized as follows. In Section 2, the details of **task 1** are explained. This task is to develop a tentative framework for documenting location information of visual data, which are challenging to obtain when they are collected in indoor spaces of buildings during post-event building inspection. The methodology is to localize images and link them to locations on a structural drawing. Images gathered along the path with a compact camera is used to compute a relative location of each image in a 3D point cloud model, which is reconstructed using a visual odometry algorithm. By projecting the point cloud model to the structural drawing, the images can be overlaid onto the drawing, providing clear context information necessary to make use of the images captured with a compact camera and the inspection images. Additionally, components- or damage-of-interest captured in these images can be reconstructed in 3D, enabling detailed assessments having sufficient geospatial context. The

outcome of this task is a tool that provide locations of images on a structural drawing and local 3D textured models illustrating the environments around them.

In Section 3, the details of **task 2** are explained. This task is to expand the capability of the framework in task 1 to cope with a large scale inspection mission, and establish a tool to automatically render the functionalities. Here, large scale inspection mission refers to an inspection mission covering multiple buildings and each of them may contain multiple floors being inspected. Images are collected during the walk-through of these environments, they are focused to record the post-event building conditions. The integrated technique developed in this task requires data separation, VO, and clustering steps. Here, "data separation," which is driven by a convolutional neural network (hereafter, CNN) image classifier (LeCun & Bengio, 1995), refers to splitting the input data according to the building floors. After separation, PathImgs are used to reconstruct Path and associated point cloud model (hereafter, Pcl) through VO. We then formulate an optimization problem to automatically overlay Path and Pcl (hereafter, PathPcl) onto the structural drawings, and link PathImgs to their position on the structural drawings. In the end, the location of each of the InspImgs is provided.

In Section 4, the details of **task 3** are explained. This task focuses on the development of an automated technique to classify the overall damage state of a building based on a typical set of reconnaissance images collected from a single building in the field (outputs of Task 1 and Task 2). The motivation is the collection of data and classification of damage into broad categories, such as those needed for computing the Hassan index. The method adopts convolutional neural network-based image classifiers to extract initial classification information, a naïve Bayes fusion algorithm to combine the information, and an integrated sampling technique to reduce the computational time without compromising the quality of the results. Validation is performed using real reconnaissance images collected from several natural hazards in the past.

In Section 5, the conclusions are discussed and the lessons learned through this study are discussed, along with some future work.

1.3 References

[1] EERI. (2009). Earthquake Clearinghouse–Earthquake Engineering Research Institute. https://www.eeri.org/

[2] Datacenterhub. (2014). https://datacenterhub.org/

[3] PEER. (2013). Pacific Earthquake Engineering Research Center. https://peer.berkeley.edu/

[4] QuakeCoRE. (2016). http://www.quakecore.nz/

[5] Yeum, C. M., Dyke, S. J., Benes, B., Hacker, T., Ramirez, J., Lund, A., & Pujol, S. (2019). Post event reconnaissance image documentation using automated classification. Journal of Performance of Constructed Facilities,33(1), 04018103.

[6] Yeum, C.M.; Choi, J.; Dyke, S.J. Automated region-of-interest localization and classification for vision-based visual assessment of civil infrastructure. Struct. Health Monit. 2019, 18, 675–689.

[7] Choi, J.; Dyke, S.J. CrowdLIM: Crowdsourcing to Enable Lifecycle Infrastructure Management. Available online: http://arxiv.org/pdf/1607.02565.pdf (accessed on 5 January 2020).

[8] GeoSLAM. Available online: https://geoslam.com/ (accessed on 5 January 2020).

[9] Leica Geosystems. BLK360. Available online: https://shop.leica-geosystems.com/ca/learn/reality-capture/blk360 (accessed on 17 January 2020).

2. AUTOMATED INDOOR IMAGE LOCALIZATION TO SUPPORT A POST-EVENT BUILDING ASSESSMENT

The objective here is to develop a technique to localize inspection images onto the structural drawings and reconstruct a local 3D textured model of scenes-of-interest. A major opportunity enabled by this technique is to achieve these capabilities without adding extra effort to or interrupting the existing data collection process in the field or without utilizing an expensive 3D sensor. In addition to the images collected for inspection (hereafter, InspImgs (inspection images)), engineers must simply collect a steady stream of images using a compact camera, potentially mounted on their hard hat or chest (hereafter, PathImgs (path images)). By implementing a visual odometry algorithm using the PathImgs, the relative locations of PathImgs along the path taken through the building being inspected are estimated, and a 3D point cloud model of the scenes is reconstructed. Structural drawings of the building may also be automatically reconstructed from drawing images (hereafter, DrawImgs (partial drawing images)) using the technique already developed [1]. By transforming the point cloud model to the drawing coordinates, InspImgs, which are taken at the time when PathImgs are captured, can be mapped and localized on the reconstructed drawing. Additionally, since I collect a large number of PathImgs along the path, any useful scenes in InspImgs can be reconstructed in 3D including color surface texture, enabling their detailed inspection with sufficient spatial context. To demonstrate the capability of the proposed technique, I have conducted an experiment on a building, assuming that I follow the typical procedures performed during a post-earthquake reconnaissance mission.

The remainder of this task is organized as follows. Section 2.1 starts with a review of the state-of-the-art in path reconstruction techniques. In Section 2.2, the technical approach is described, including a detailed technical discussion of the image collection, path reconstruction, drawing reconstruction, and the path overlaid onto the drawing. In Section 2.3, the technique is demonstrated using the images collected by a human data collector walking through an actual building. This chapter is adapted from the published work of the author [2].

2.1 Literature Review of Path Reconstruction Techniques

A main technical challenge of this approach lies in how to recreate the path of an engineer walking through an indoor environment so that InspImgs can be mapped onto the reconstructed path. The use of GPS metadata in each image would be a logical solution, but they are extremely limited within an indoor environment [3]. Another possible technique for localizing image positions where no GPS signal exists is an indoor positioning system. Such systems adopt beacon-based methods of communicating with various signals like vision, infrared, ultrasound, Bluetooth, Wi-Fi, and radio-frequency identification (RFID) [4-10]. They are based on communications or measurements between mobile devices and fixed beacons that serve as landmarks. However, the preparation needed to place the necessary fixed devices before using such a system is an obvious limitation [11]. In reality, after significant disasters, buildings often have no electricity and telecommunication services are not available. Furthermore, time is tight and engineers want to collect data from several buildings each day. In typical reconnaissance procedures, it would be extremely unlikely to have the ability to set up the necessary indoor landmarks before gathering data. Therefore, beacon-based localization is infeasible in the field.

Alternatively, visual odometry (hereafter, VO) provides a potential solution that offers accurate positional output without having prior information about the environment and without relying on other sensors installed in the building [12,13]. Originating from visual-based navigation systems for mobile robots called simultaneous localization and mapping (SLAM), the technique performs the localization of a camera (including transposition and rotation movement) using a stream of still images as inputs. The accuracy and speed are improved by incorporating techniques, such as loop closure detection, map reuse, etc. [14-16]. In general, depending on how many cameras are engaged in the data collection process, VO can be categorized as either stereo or monocular VO [12,13]. Stereo VO, inspired by human eyes, constructs 3D depth images [17]. The main advantage of the stereo VO is that scale information of the scene can be obtained from the known distance between the lens, called, intra-axial distance. However, users need to purchase a stereo camera or manually calibrate two cameras to implement the algorithm. In contrast to stereo VO, monocular VO only uses the data from a low-cost single camera. The main disadvantage of monocular VO is that this method can only provide relative positions because there is no physical scale information available to the algorithm. In the technique described herein, the true (physical) scale information is not necessary because I only need to find the parameters needed for

transforming the reference coordinate for the camera path to the coordinate for the structure drawing. Thus, I selected monocular VO to estimate the position information by collecting and processing a stream of PathImgs.

Several successful monocular VO have been published by researchers. For instance, parallel tracking and mapping (PTAM) has improved mapping results using a new feature-based method [18], and oriented fast and rotated BRIEF SLAM (ORB-SLAM) has achieved accurate reconstruction results in a fast speed using the key frame notion [19]. Direct methods have been shown to rebuild the path and the map accurately as well, such as large-scale direct monocular SLAM (LSD-SLAM) [20]. Here, for path reconstruction, I adopted a state-of-art odometry technique called direct sparse odometry (DSO). DSO has been recognized as one of the best VO techniques with several advantages, including accuracy, processing and implementation times, etc. [21].

2.2 Technical Approach

An overview of the technical approach is shown in Figure 2.1. The technique consists of three main steps including image collection, data processing, and data visualization, and implements four algorithms (marked as A–D) into the process to generate two outcomes: a drawing overlaid with InspImgs and a local 3D textured model in Step 3. In Step 1, as the input for the proposed technique, engineers collect three types of images from the building including InspImgs, PathImgs, and DrawImgs. InspImgs are the data being collected in the field, aiming to capture buildings and their components for the purposes of visual assessment and documentation. Collected concurrently with InspImgs, PathImgs contain an image sequence recorded continuously in time in order to capture the scene in front of the engineer as he/she walks through the building and thus document the path taken. Thus, PathImgs are not selectively captured by focusing on specific objects or damage, like how InspImgs are captured. PathImgs and InspImgs are synchronized in time, used for later localization. DrawImgs are part of the metadata captured using images [1]. DrawImgs contain a portion of a physical drawing while preserving its details. In the proposed technique, in addition to InspImgs and DrawImgs, which are normally captured in a typical reconnaissance mission, engineers simply collect PathImgs by mounting an extra camera on a hard hat or body.

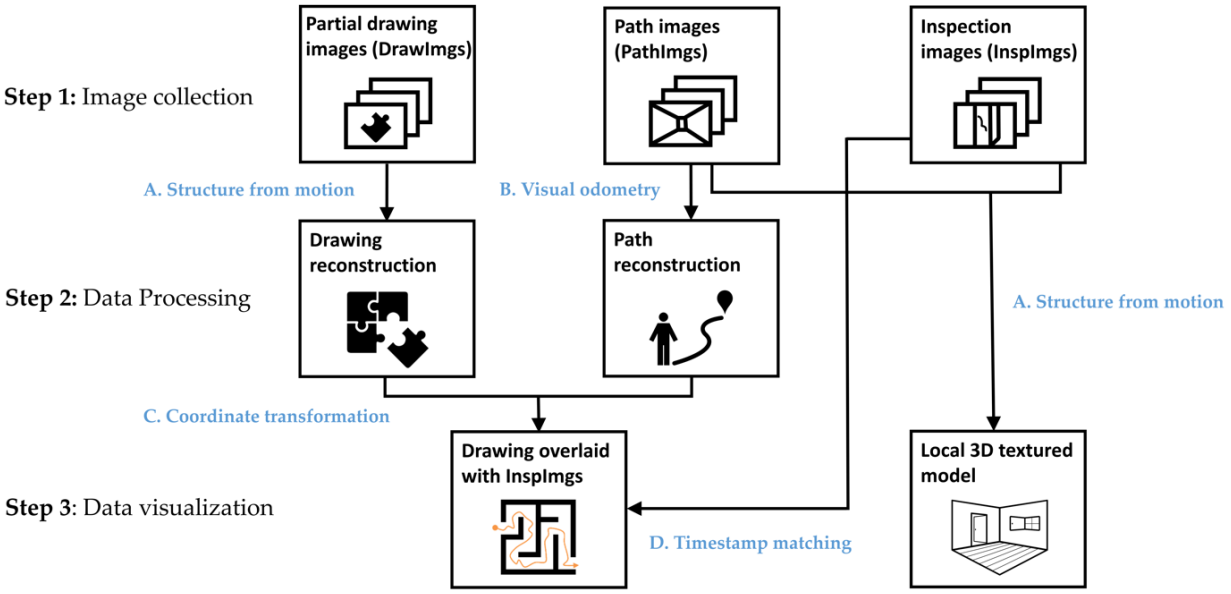


Figure 2.1. Overview of the technical approach.

After these images are gathered, in Step 2, the path of the engineer is reconstructed with the set of PathImgs using the DSO algorithm, which is one of the most popular VO algorithms (B). The 3D point cloud of the scenes included in PathImgs is also reconstructed. This process is to estimate relative locations between the PathImgs so that they are overlaid to the drawing, followed by localizing InspImgs. In this step, the structural drawing is also reconstructed from DrawImgs using the structure-from-motion algorithm (A).

Then, in Step 3, the path and 3D point cloud are overlaid onto the reconstructed drawing using a coordinate transformation (C). The transformation matrix is computed using an interactive tool by manually but rapidly finding the correspondence between a few images (around 10 images) in PathImgs (localized in the 3D point cloud) and their approximate capture locations on the drawing. Here, the purpose of mapping the path to the drawing is to localize InspImgs. Since the InspImgs are captured while PathImgs are continuously collected, approximate locations of InspImgs are easily identified from PathImgs having the closest timestamp (D). Additionally, PathImgs and InspImgs are used for reconstructing the local 3D textured model using the structure-from-motion algorithm (A). In the end, the user selects any InspImg and the proposed technique automatically informs its position on the reconstructed drawing and if needed, the scene on the selected InspImgs can be reconstructed in 3D. It is worth mentioning that the only manual process

required in this technique is to manually match a select group of PathImgs with their corresponding locations on the drawing using the interactive tool for obtaining the transformation matrix.

2.2.1 Reconnaissance Image Collection

2.2.1.1 Collecting InspImgs and DrawImgs

The procedure for acquiring InspImgs is governed primarily by typical protocols for obtaining reconnaissance images that are useful for documenting the perishable visual evidence of the hazard event. To gather high-quality images, InspImgs should be captured without blur. As the engineer approaches a building scene of interest, he/she aims to capture a few InspImgs of that specific scene from various perspectives and distances.

DrawImgs are independent of PathImgs and InspImgs and are meant to produce a high-resolution image of the structural drawing when that is not available in a digital or carriable form. Certain guidelines do need to be followed for taking DrawImgs, as explained in detail in [1]. In short, the physical drawing is placed in a flat and non-obstructed position; each DrawImg should have a large shared region (overlap), with adjacent DrawImgs; all contents of the drawing should be included in the complete set of DrawImgs; and, each DrawImg should be taken from a position in which the camera is pointing to the drawing while maintaining a similar distance from the drawing.

2.2.1.2 Collecting PathImgs

PathImgs are a sequence of images that are taken automatically by a compact and mountable camera that can be carried during the building walk-through. They record a stream of scenes that the field engineer observes along this path, and eventually, they are used for reconstructing the entire of the path using DSO, that the engineer walks through. Engineers do not need to take extra efforts to collect PathImgs if the camera is mounted on the hard hat or chest. Since PathImgs are collected mainly for this purpose, there is no need for the engineer to modify his or her motions or directions. Additionally, PathImgs are not the images that engineers will sift through for inspection or condition assessment. However, there are some considerations on the selection of the camera and its calibration before collecting images.

There are three factors to consider when the camera is selected. First, a global shutter camera is the best option for collecting PathImgs, while a rolling shutter camera would not be a suitable choice. This choice allows for avoiding the jello effect in PathImgs [22]. Second, motion blur in the PathImgs must be avoided. The selected camera should support a faster shutter speed with high ISO without dropping image quality. Third, the camera should support a high frame-per-second (fps) video or continuous shots. If the absolute motion between two consecutive PathImgs is too large, the DSO algorithm will produce large modeling errors and they are accumulated in the course of path reconstruction. These three considerations will guarantee the capture of valid PathImgs that can be used for accurate path reconstruction.

To use DSO, initial camera calibration is necessary. The intrinsic parameters of the compact camera must be determined accurately through the camera calibration process when they are not provided by the manufacturer. There are many ways to calibrate a digital camera, but the chessboard calibration method is widely used to find all these parameters [23,24]. In this method, a chessboard pattern having clean borders between black and white cells is placed on a flat table or attached to a wall, and the camera is used to take images from various angles with the full chessboard in view. Normally 10 to 30 calibration images are sufficient to perform geometric camera calibration [23,24]. This camera calibration is independent of the PathImg collection and should be done before the actual data collection.

2.2.2 Path Reconstruction

I adopted DSO to generate the path associated with data collection. The path was generated based on the stream of PathImgs gathered during the mission. Based on the performance evaluation described in the original DSO-based work and the preliminary tests, DSO has shown superior performance in terms of accuracy and speed among several monocular VOs. Additionally, DSO offers an easy-to-implement strategy and does not require special programming libraries and hardware.

To better understand the working principle of DSO, the workflow of the algorithm is summarized in Figure 2.2 when a new image is input (hereafter, NImg) [21]. Steps (a)–(h) in the procedure were repeated for each subsequent NImg until all images (PathImgs) were scanned (inputted). Basically, the point cloud and pose estimation were performed on a subset of images, called the sliding window (hereafter, SW) and the images in SW are called key frames (hereafter,

KeyFrames). The process in Figure 2.2 is to determine whether the NImg is eligible for KeyFrame and to perform the joint optimization over the KeyFrames to update the point cloud and the pose of the KeyFrames once the NImg is added to the SW.

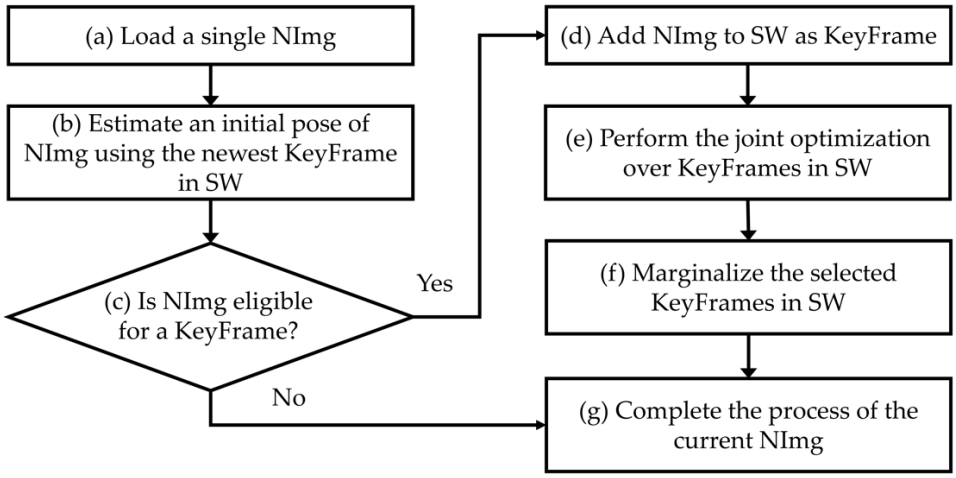


Figure 2.2. Flow chart of the direct sparse odometry (DSO) algorithm.

In Step (a), a single NImg is fed into the algorithm. Then, an initial pose of that NImg is roughly estimated by matching the points on the newest existing KeyFrame in SW. With the outcomes from Step (b), a series of strategies are applied to decide whether or not NImg can serve as a new KeyFrame in SW. This decision is based on, for example, i) whether or not the field of view has changed since the most recent KeyFrame, which is measured the mean square optical flow from the newest KeyFrame to NImg, ii) a camera translation has caused an occlusion or disocclusion, which is measured by the mean flow without rotation, or iii) the camera exposure time has changed significantly, which is measured by the relative brightness factor between the newest KeyFrame and NImg [21]. A condition for candidacy as a new KeyFrame is the image having a large relative movement from previous KeyFrames. In Step (c), if the NImg is not eligible for a new KeyFrame, the NImg only contributes to update the depth values of inactive points in SW, which are the points used for the future joint optimization. The NImg assigned as non-KeyFrame is not involved in the optimization process for point cloud and pose update. If NImg is qualified as the new KeyFrame, the NImg will be added into SW in Step (d). Subsequently, a joint optimization is performed, which is the core part of DSO in Step (e).

The technical details of this process are delineated in the following paragraph. In short, the intensity difference between the points on the newly added KeyFrame and the existing KeyFrames in SW is minimized using the Gauss-Newton method for parameter optimization. Once the new KeyFrame is added in SW and successfully registered in the existing model, in Step (f), I deactivated some KeyFrames in SW, which will not contribute to the upcoming match with NImg. This process is called marginalization and helps to maintain a consistent number of KeyFrames in the SW, thereby improving the efficiency of the optimization in DSO. Finally, in Step (g) the operations related to the current NImg ends, and the same process is then repeated for the next NImg.

In the optimization in Step (e), a cost function is designed to minimize the difference of intensity values between points in each KeyFrames with the projected points from all the other KeyFrames in SW. The formulation of the cost function starts with the difference between one point in the host KeyFrame and the projected point from a reference KeyFrame in SW, denoted E_{pij} as

$$E_{pij} \coloneqq \left\| I_j(p') - I_i(p) \right\|_2 \tag{2.1}$$

where the points from the host (i) and reference (j) KeyFrame are denoted as p and p', respectively. I stands for the pixel intensity and $I(p)$ is the intensity value at p. In addition, $\|\cdot\|_2$is the l^2-norm. Equation (2.1) computes the difference of intensity values between the point p and the projected point p'. The process of projection described in [21] is based on a pinhole camera geometry as

$$p' = \Pi_c(R \cdot \Pi_c^{-1}(p, d_p) + t) \tag{2.2}$$

where the movement from the position where the camera takes KeyFrame i to KeyFrame j is modeled as a rotation and translation. R is a rotation matrix, and t is the translation vector. d_p is depth value of point *p,* which is the perpendicular distance from the principal plane of the camera to the world point, which p represents in KeyFrame i. $\Pi_c()$ stands for the projection process of the camera from the corresponding world point to the image point p, and $\Pi_c^{-1}()$ is the inverse projection process.

The DSO algorithm considers an intensity calibration factor in Equation (2.1), expanding it to

$$E_{pij} := \sum_{p \in N_p} w_p \left\| I_{raw}(I_j(p')) - I_{raw}(I_i(p)) \right\|_r$$
$$= \sum_{p \in N_p} w_p \left\| (I_j(p') - b_j) - \frac{t_j e^{a_j}}{t_i e^{a_i}} (I_i(p) - b_i) \right\|_r \quad (2.3)$$

To compensate for unknown intensity calibration factors involved in the imaging process of a camera, a function is defined to reflect such a process as

$$I_{raw}(I(p)) = \frac{I(p) - b}{te^a} \quad (2.4)$$

This equation converts the intensity value of point p in one image to the raw intensity value that the camera should capture. Here, t is the exposure time of the image, a and b are constants regulating this converting process. These two parameters are taken as unknown values to be calibrated in the optimization. Note that if a camera does not record an accurate exposure time, the exposure time t is simply set to 1. Here, l^2-norm is replaced by the Huber norm $\|\cdot\|_r$ [25] to increase resistance to outliers. In addition, the weight term (w_p) in Equation (2.3) is used to accommodate points with different gradients. The weight term is defined in [21] as

$$w_p = \frac{c^2}{c^2 + ||\nabla I_i(p)||_2^2} \quad (2.5)$$

where $\nabla I_i(p)$ is the gradient vector at point p in KeyFrame i, and $||\nabla I_i(p)||_2^2$ is the square of its l^2-norm. A factor c is a constant regulator and is set to 0.75 in this task. To improve the robustness of the cost function, DSO utilizes eight neighborhood points to compute the intensity difference for point p including its nearby region. This set of points is denoted as N_p.

E_{pij} is summed up over the points in all host-reference KeyFrame combinations. E_{pij} is established in terms of point p in KeyFrame i, which is observed in KeyFrame j. $\sum_{j \in obs(p)}$ is the summation where j over all the KeyFrames in SW in which p is visible. $\sum_{p \in P_i}$ is where point p over all the points P_i in KeyFrame i. $\sum_{i \in F}$ indicates that i becomes all the KeyFrames in SW. Thus, the final cost function is defined as

$$E = \sum_{i \in F} \sum_{p \in P_i} \sum_{j \in obs(p)} E_{pij} \quad (2.6)$$

The Gauss-Newton method is used to find the global minimum of this cost function. The unknown parameters to be computed include the rotation matrix R, the translation vector t, the

depth value d_p of point p, parameters of the imaging process a and b, and the camera intrinsic parameters [21]. The camera intrinsic parameters are treated as variables in the optimization for fine-tuning from the initial estimates found in the camera calibration. Among these parameters, the camera position, R and t, are the desired output in this study.

2.2.3 Drawing Reconstruction

Images of structural drawings are often collected as a part of a building reconnaissance dataset to document the details of the structural system and design. When the digitized version of the drawings is not available, for example with older buildings, field engineers must take multiple photographs of the hard copy of the structural drawings to capture this information in a legible and complete form. To accomplish this task, they capture DrawImgs, because it is often difficult to include the entire view in one single photograph. A method is available to automatically organize these DrawImgs and restore a complete high-resolution drawing in a digital form from these DrawImgs by the coauthors. DrawImgs are first automatically filtered out from the entire building image collection using a convolutional neural network classifier. Then, the DrawImgs are grouped according to the original drawing that they belong to. After that, a full reconstruction of each page of the drawings is obtained. More details are provided in [1].

2.2.4 Overlaying the Path with the Drawing

The 3D point cloud is projected to a 2D plane in the gravity (height) direction. Thus, I could use two independent sets of 2D points to represent the path defined by the 3D point cloud and the reconstructed drawing. Then the path can be overlaid onto the drawing by finding the transformation matrix between these two sets of data. Based on the correspondence between the locations of some PathImgs and their locations on the drawing, the transformation matrix for projecting 3D points cloud and path onto the drawing can be computed. I used the absolute orientation method to find the optimal transformation matrix between these correspondences [26].

An interactive tool was developed to assist with this task. The objective of this tool is to rapidly match PathImgs with their locations on the reconstructed drawing. The tool shows a group of PathImgs to the engineers so that they can select and match PathImgs to the corresponding locations on the reconstructed drawing. If there is no suitable PathImg in the group, or if the

location is hard to recognize, the engineers can select PathImgs from another group. The tool supports the function of enlarging the drawing to improve selecting the location. I repeated this process until engineers have selected a sufficient number of image-location pairs. Around 10 pairings across the drawing are sufficient for obtaining the transformation matrix. The more pairings are given, the more accurate the projection result will be. Note that the selected points should be equally distributed over the entire path rather than gathered in a specific region of the drawing. Once the selection process is completed, the tool automatically computes the transformation matrix and conducts the coordinate transformation to overlay the path and 3D point cloud onto the drawing as the outcome. Figure 2.3 shows 10 PathImg-locations pairs used for experimental demonstration in Section 2.3. The images on the right are PathImgs selected and their corresponding locations are marked on the reconstructed drawing.

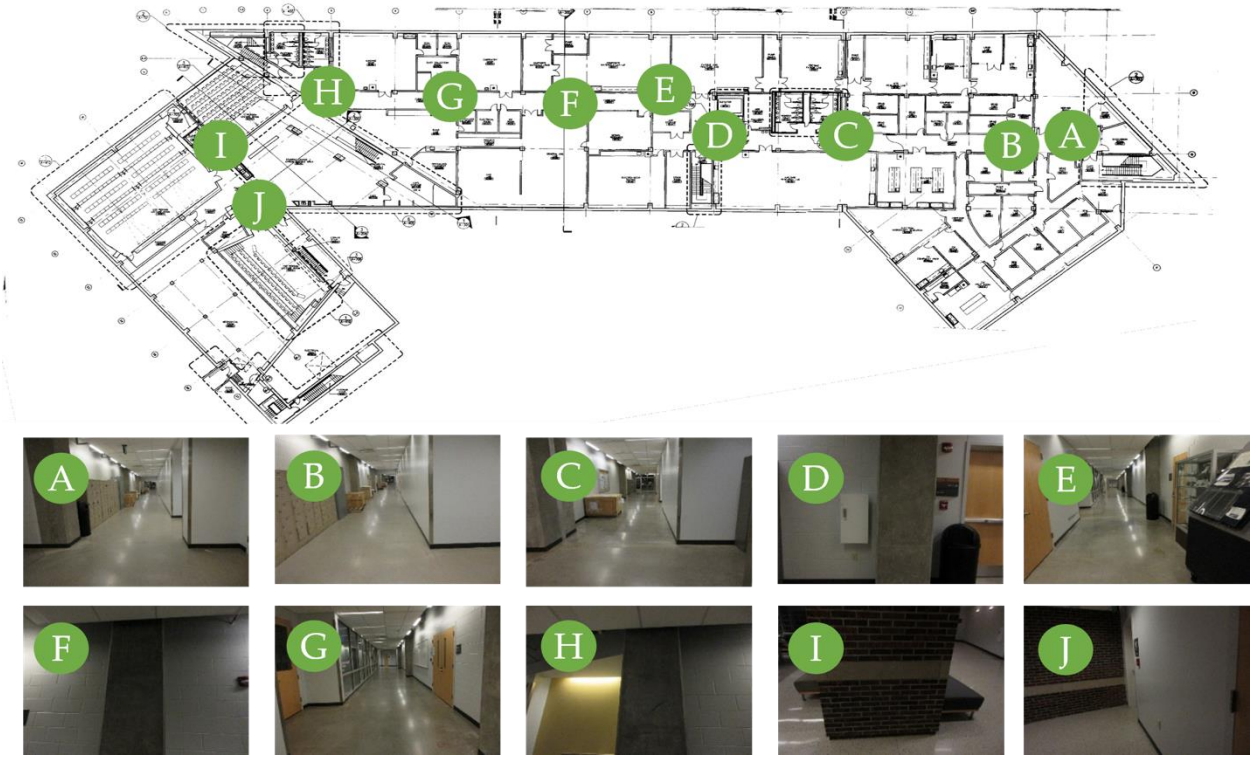


Figure 2.3. Path transformation: 10 images are matched with the corresponding locations on the reconstructed drawing.

2.3 Experimental Verification

2.3.1 Description of the Test Site

(**a**)

(**b**)

Figure 2.4. Experimental test site (Armstrong Hall at Purdue University, United States): (**a**) building overview and (**b**) basement floor plan: InspImgs and PathImgs are collected along the corridor highlighted as solid blue. Sample images corresponding to key spots in the corridor are provided.

Experimental verification was performed on an actual building. I chose the basement floor of Armstrong Hall on the Purdue University campus as a test site, shown in Figure 2.4a. The area of the basement floor was about 175 m × 60 m and its digital drawing including sample images of

key places is presented in Figure 2.4b. A long corridor of 175 m (long) by 3 m (wide) was located along the centerline of the floor, marked as solid blue in Figure 2.4b. Since the corridor was sufficiently long and had several turns to walk through, and the structural columns were exposed in the corridor (see Figure 2.4b), the scenes in the test site do represent the actual building environment that engineers would visit after earthquake events. I collected the necessary images, including InspImgs and PathImgs, by emulating the inspection and data collection steps that would be taken in a typical post-earthquake field reconnaissance mission.

2.3.2 Collection of the Image Data

I manually collected PathImgs using a compact camera (Canon 350HS), InspImgs and DrawImgs using a DSLR camera (Nikon D90). The size of the Canon 350HS was 3.92 inches × 0.9 inches × 2.28 inches, and the weight was 5.19 ounces. Overall, 3687 PathImgs, 232 InspImgs, and 44 DrawImgs were collected, and their resolutions were 2595 pixels × 1944 pixels, 4288 pixels × 2848 pixels, and 4288 pixels × 2848 pixels, respectively. Their sample images are shown in Figure 2.5. The compact camera was set to burst mode, which continuously took PathImgs at 7.8 fps. The focal length was fixed throughout the entire experiment because the initial calibration parameters remained unchanged. I avoided the jelly effect in this experiment by simply walking slowly, at about 1/3 of the normal walking speed of a human. By doing this, I did not notice an obvious jelly effect, and if there were, the errors did not influence the quality of the result. Additionally, the Canon 350HS is just one sample of a suitable camera to collect PathImgs and a baseline for choosing the camera device. By using cameras with faster shutter speeds with high ISO, one can avoid the need to walk slower. In this experiment, two people collect InspImgs and PathImgs at the same time. However, the actual collection of PathImgs is devised to be automated with a mountable camera (e.g., action camera) by a single engineer.

The DSLR camera was used to collect InspImgs and DrawImgs. A major distinction between InspImgs and PathImgs is that PathImgs represent a stream of image sequence without gazing at any specific objects or regions that the engineers would be interested in, while InspImgs are non-periodic image shots targeting objects-of-interest at various viewpoints and distances. For instance, assume that the engineer gazes at interesting objects (e.g., columns and walls) or certain evidence of damage (e.g., crack and spalling), PathImgs and InspImgs capture different information: PathImgs, as in Figure 2.5a, captures the views in front of the engineer, regardless of

whether the compact camera is directly facing any particular building components of interest. The scenes in these images will turn upwards or downwards following the gaze of the engineer. On the other hand, InspImgs, as in Figure 2.5b, were aimed at the objects and regions that the engineer found interesting and chose to document.

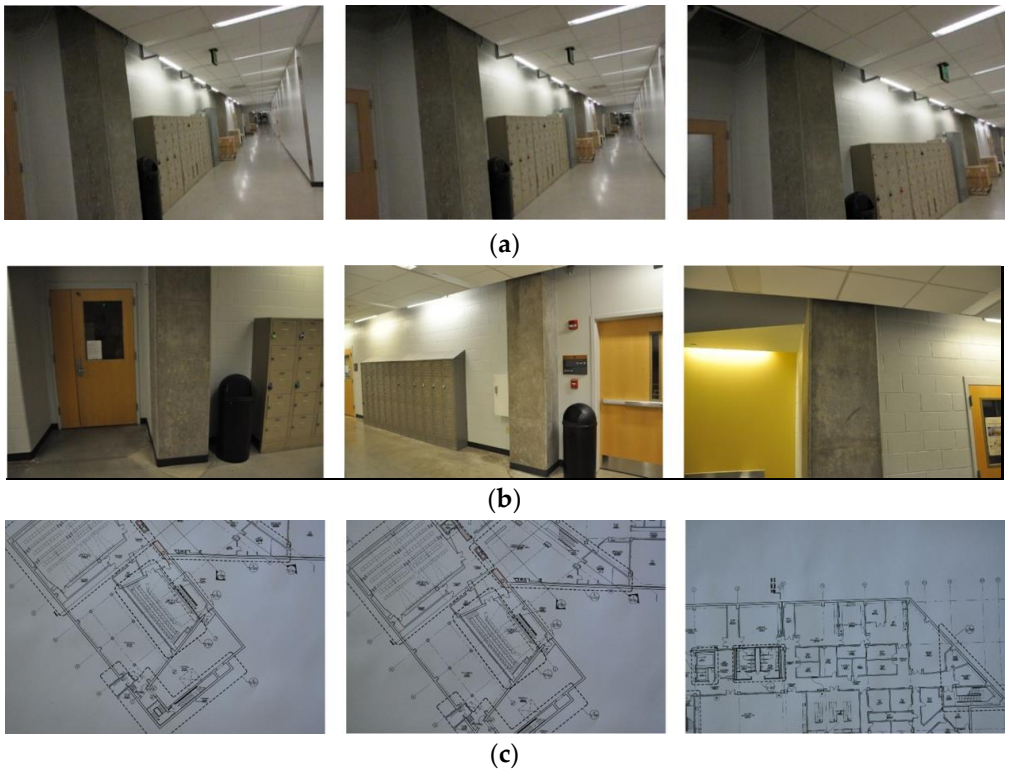


Figure 2.5. Sample images collected during the test: (a) PathImgs, (b) InspImgs, and (c) DrawImgs.

The image data collected was intended to cover the entire corridor area highlighted in Figure 2.4b. It followed the image collection guideline introduced in Section 2.1. The compact and DSLR cameras were set to have the same timestamp before conducting the experiment. It spent 11 min and 45 s to collect both InspImgs and PathImgs by walking through the entire corridor and performing inspection actions, such as observing structural conditions and taking more photos of structural elements, which is to emulate an actual post-earthquake reconnaissance mission.

Regarding DrawImgs, I assumed the situation where only a paper copy of the drawing was available to the engineers (although in this case I did have a digital drawing shown in Figure 2.4b). The digital drawing was printed on a large engineering paper (A1) and DrawImgs captured the

drawing, following the image collection guideline provided in the coauthor's paper [1]. The paper copy of the drawing was placed on a large flat table and images were taken at a suitable distance from the drawing in such a way that details of the drawing (e.g., line and number) were visible. The total number of images depends on the size and details of the drawings. In this task, I collected 44 DrawImgs from a single drawing of the basement to capture all the details and some sample images are in Figure 2.5c. Note that the original technique proposed by the coauthor also performs image classification and drawing matching techniques so that DrawImgs are automatically extracted from a set of images collected and individual drawing images are created from DrawImgs captured from multiple drawings. However, in this task, I only implemented the drawing image generation (stitching) technique using a set of DrawImgs collected from a single drawing.

I used a workstation with an Intel i9-7920x CPU, 32 Gb memory, and a NVIDIA GeForce RTX 2080Ti video card. The path reconstruction with 3687 PathImgs and drawing reconstruction with 44 DrawImgs took less than 20 min. Generating the local 3D surface model for one scene took about 2.5 h using 402 images, although the actual time would vary for each case depending on the number of images. All these processes were fully automated. The only manual task was to match PathImgs to the corresponding locations in the reconstructed drawing in order to compute the transformation matrix between the 3D point cloud and the reconstructed drawing image. However, this task took less than five minutes to match 10 PathImgs, shown in Figure 2.3.

2.3.3 Results

2.3.3.1 Path Reconstruction

The path was reconstructed using a stream of PathImgs. Video footage was also applicable after transferring video footage to images. I used the compliable source code [27], published by the DSO creators, to generate the DSO software. It is written in C++, run in Linux 14.04, and operated with Linux bash command lines. When DSO was applied to PathImgs, KeyFrames were automatically extracted and used to estimate the positional information. Here, 1428 images were identified as KeyFrames and their relative positions in the reference coordinate were estimated. I linearly interpolated between every two consecutive KeyFrames, each PathImg was assigned with a relative position for both KeyFrames and non-KeyFrames, excluding PathImgs that were dropped in the initialization process. The reconstructed path consisted of a set of 3607 discrete

points associated with PathImgs, and each discrete point had a 7-dimensional positional vector including a three-dimensional translation and a four dimensional quaternion to represent a 3D rotation with respect to the reference coordinate system (as mentioned in Section 2.2). Moreover, the 3D point cloud was also generated from the scenes including wall, doors, columns, of which scenes were contained in KeyFrames.

Figure 2.6 shows the reconstructed path and 3D point cloud, which were viewed (or projected) in the gravity (height) direction. Most PathImgs were normally captured while the gravity direction was aligned with the image height. Thus, I could easily compute the gravity direction of the reconstructed model for projection. In Figure 2.6, each red point indicates each PathImg location (although they look as if they were connected) and blue points were the point cloud (for a black and white figure, the line passing through the middle of the encompassed region was the set of red points.) A majority of points (blue points) were likely generated from the perpendicular features adjacent to the corridors like walls or doors. Thus, the blue points formed a layout of the walls along the corridor.

As mentioned in Sections 2.1 and 2.2, DSO was based on image collection using a monocular camera, which could not determine a real-world scale. Thus, the points in Figure 2.6 are represented in hypothetical units, which have no physical scale information. However, they are proportional to real-world units, facilitating mapping the path to the drawing image. Note that I manually rotated the reconstructed model in Figure 2.6 to be horizontal for better visualization.

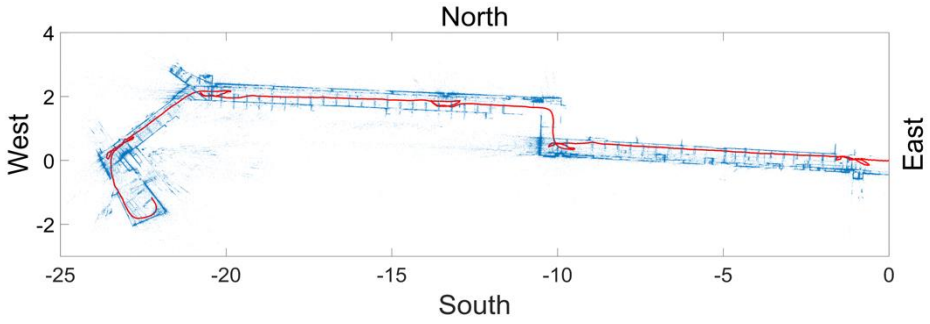


Figure 2.6. Reconstructed image collection path and a point cloud of the scenes on PathImgs: The points in blue represent the reconstructed point cloud and form a layout of the walls in the corridor. A set of red points passing through the corridor is the locations of PathImgs. The values are represented using a hypothetical unit in the initial reference coordinate system and have no physical scale information.

2.3.3.2 Drawing Reconstruction

The drawing reconstruction method [1] is implemented to the collected DrawImgs and the resulting reconstructed image of the structural drawing is present in Figure 2.7. The overall quality of the drawing was quite satisfactory, and as was clear from the enlarged areas next to the full drawing. All detailed were preserved, even small texts and thin lines. The color and orientation of the reconstructed drawing in Figure 2.7 were manually tuned for better visualization. It should be emphasized that this method possessed the ability to automatically restore multiple drawings from a mixed set of DrawImgs that included images of more than one drawing. However, in this experiment, I only reconstructed a drawing for a single basement floor using the corresponding DrawImgs. Additionally, if the digital drawing is available, such a drawing reconstruction step can be skipped. This digital drawing was used directly for the overlay step as in the next section, replacing the reconstructed drawing image.

Figure 2.7. Drawing image reconstructed using DrawImgs: The four images on the bottom are magnified areas corresponding to the boxes on the drawing on the left.

2.3.3.3 Path Overlay

Following the steps explained in Section 2.4, I selected 10 PathImgs and their corresponding positions in the reconstructed drawing for computing the transformation matrix. The transformation matrix was used to map the point cloud and PathImg locations (in Figure 2.6)

to the reconstructed drawing (in Figure 2.7). The reconstructed drawing overlaid with the point cloud is shown in Figure 2.8. The overall result was quite accurate. A majority of blue points were well aligned with the corridor wall boundary on the drawing. Note that I did not conduct a quantitative evaluation of the mapping result because it was sufficient to identify approximate locations of PathImgs for the purpose of documenting the path of the engineer and associating specific images with that reconstructed path and the structural drawing. Here, the reconstructed drawing image in Figure 2.8 was the binary image converted from the drawing image in Figure 2.7 so that the line and text in black were clearly legible.

In Figure 2.8, five regions on the drawing were enlarged. The walking (inspection) path (a set of red points) was not straight because I mimicked the actual inspection procedure, such as wandering around to see components-of-interest for close-up inspection. This type of walking is likely application-specific and certainly challenges this technique. Additionally, more blue points were generated near the scenes-of-interest because the inspector spent more time near those regions, collecting more PathImgs.

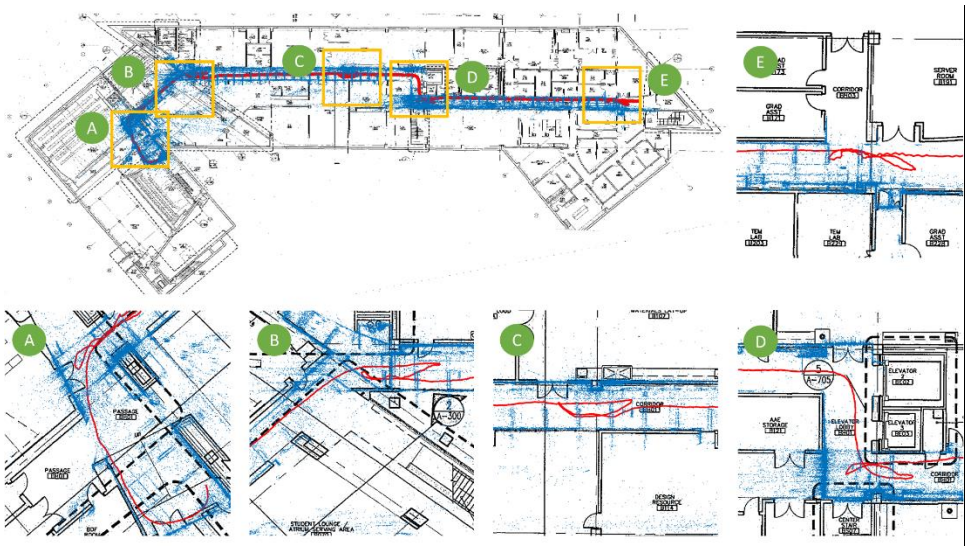


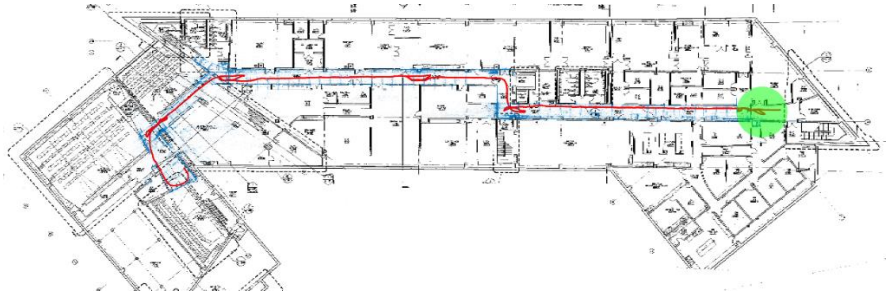
Figure 2.8. The reconstructed drawing in Figure 2.7 overlaid with the 3D point cloud and PathImgs locations in Figure 2.6.

2.3.3.4 Image Localization and Local 3D Textured Model Reconstruction

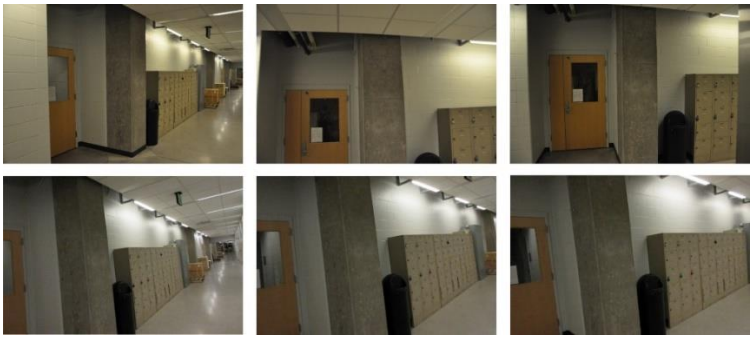
I made an in-house tool for reviewing InspImgs and their localization results using MATLAB. When the user selects a particular InspImg, the tool searches for the PathImg that is

captured around the same time when the selected InspImg was taken. Then, the location of the selected InspImg is automatically marked on the drawing. For example, Figure 2.9a shows a selected InspImg, which contains a reinforced concrete structural column. Figure 2.9b is the location of the corresponding image automatically marked as a circle on the reconstructed drawing.

(**a**)



(**b**)



(**c**)

(**d**)

Figure 2.9. Image localization and local 3D textured model generation: (**a**) selected InspImg, (**b**) its location on the reconstructed drawing, (**c**) InspImgs (first row) and PathImgs (second row) collected at a similar time when the selected InspImg is taken, and (**d**) reconstructed local 3D textured model.

In the method, I did not combine the tool with an SfM-based 3D texture modeler. Herein, I simply show the capability for generating a 3D textured model using SfM software. As mentioned in Section 2.4.2, the clocks of both the compact and DSLR cameras were synchronized. Thus, the selected InspImg could be readily paired with the PathImgs and InspImgs that have a nearby timestamp. I roughly set 30 s for PathImgs and 5 s for InspImgs, both before and after, to extract views containing the same scene on the selected InspImg. This set of extracted images became an input for the SfM software. Here, for this task only, I used the commercial software, Pix4D mapper 4.4.4. In Figure 2.9c, samples of identified PathImgs and InspImgs are shown. These samples were captured during a time range around the selected image in Figure 2.9a. The local 3D textured model generated using the images in Figure 2.9c is shown in Figure 2.9d. Once a user selects any InspImg, the full process, including its localization and local 3D textured model construction, is automated. Then, engineers can review the scenes on the selected InspImg with sufficient spatial context.

2.4 Published manuscript

Liu, X., Dyke, S. J., Yeum, C. M., Bilionis, I., Lenjani, A., & Choi, J. (2020). Automated Indoor Image Localization to Support a Post-Event Building Assessment. *Sensors*, *20*(6), 1610.

2.5 Author Contributions

Liu and Yeum generated the concept behind the work.
Liu was responsible for data collection, coding and data analysis to generate results.
Lenjani and Choi supported data collection and method verification.
Liu wrote the paper with support from Dyke, Yeum and Bilionis.
Dyke, Yeum and Bilionis provided supervision.

2.6 References

[1] Yeum, C.M.; Lund, A.; Dyke, S.J.; Ramirez, J. Automated Recovery of Structural Drawing Images Collected from Postdisaster Reconnaissance. *J. Comput. Civ. Eng.* **2018**, *33*, 04018056.

[2] Liu, X., Dyke, S. J., Yeum, C. M., Bilionis, I., Lenjani, A., & Choi, J. (2020). Automated Indoor Image Localization to Support a Post-Event Building Assessment. *Sensors*, *20*(6), 1610.

[3] Kos, T.; Markezic, I.; Pokrajcic, J. Effects of multipath reception on GPS positioning performance. In Proceedings of the ELMAR-2010, Zadar, Croatia, 15–17 September 2010.

[4] Want, R.; Hopper, A.; Falcao, V.; Gibbons, J. The active badge location system. *Trans. Inf. Syst.* (*TOIS*) **1992**, *10*, 91–102.

[5] Bahl, V.; Padmanabhan, V. *Enhancements to the RADAR User Location and Tracking System*; Microsoft Corporation: Redmond, WA, USA, 2000.

[6] Gutmann, J.-S.; Fong, P.; Chiu, L.; Munich, M.E. Challenges of designing a low-cost indoor localization system using active beacons. In Proceedings of the 2013 IEEE Conference on Technologies for Practical Robot Applications (TePRA), Woburn, MA, USA, 22–23 April 2013.

[7] Pierlot, V.; Droogenbroeck, M.V. A beacon-based angle measurement sensor for mobile robot positioning. *IEEE Trans. Robot.* **2014**, *30*, 533–549.

[8] Meng, W.; He, Y.; Deng, Z.; Li, C. Optimized access points deployment for WLAN indoor positioning system. In Proceedings of the 2012 IEEE Wireless Communications and Networking Conference (WCNC), Paris, France, 1–4 April 2012.

[9] Willis, S.; Helal, S. A passive RFID information grid for location and proximity sensing for the blind user. *Univ. Fla. Tech. Rep.* **2004**, *TR04–TR09*, 1–20.

[10] Shirehjini, A.A.N.; Yassine, A.; Shirmohammadi, S. An RFID-based position and orientation measurement system for mobile objects in intelligent environments. *IEEE Trans. Instrum. Meas.* **2012**, *61*, 1664–1675.

[11] Ruiz-López, T.; Garrido, J.L.; Benghazi, K.; Chung, L. A survey on indoor positioning systems: Foreseeing a quality design. In *Distributed Computing and Artificial Intelligence*: Springer: Berlin, Germany, 2010; pp. 373–380.

[12] Scaramuzza, D.; Fraundorfer, F. Visual odometry [tutorial]. *IEEE Robot. Autom. Mag.* **2011**, *18*, 80–92.

[13] Fraundorfer, F.; Scaramuzza, D. Visual odometry: Part II: Matching, robustness, optimization, and applications. *IEEE Robot. Autom. Mag.* **2012**, *19*, 78–90.

[14] Fuentes-Pacheco, J.; Ruiz-Ascencio, J.; Rendón-Mancha, J.M. Visual simultaneous localization and mapping: A survey. *Artif. Intell. Rev.* **2015**, *43*, 55–81.

[15] Aulinas, J.; Petillot, Y.R.; Salvi, J.; Lladó, X. The slam problem: A survey. *CCIA* **2008**, *184*, 363–371.

[16] Taketomi, T.; Uchiyama, H.; Ikeda, S. Applications. Visual SLAM algorithms: A survey from 2010 to 2016. *IPSJ Trans. Comput. Vis. Appl.* **2017**, *9*, 16.

[17] Nistér, D.; Naroditsky, O.; Bergen, J. Visual odometry. In Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2004), Washington, DC, USA, 27 June–2 July 2004.

[18] Klein, G.; Murray, D. Parallel tracking and mapping for small AR workspaces. In Proceedings of Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality, Nara, Japan, 13–16 November 2007.

[19] Mur-Artal, R.; Montiel, J.M.M.; Tardos, J.D. ORB-SLAM: A versatile and accurate monocular SLAM system. *IEEE Trans. Robot.* **2015**, *31*, 1147–1163.

[20] Engel, J.; Schöps, T.; Cremers, D. LSD-SLAM: Large-scale direct monocular SLAM. In Proceedings of the 13th European conference on computer vision (ECCV 2014), Zurich, Switzerland, 6–12 September 2014.

[21] Engel, J.; Koltun, V.; Cremers, D. Direct sparse odometry. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *40*, 611–625.

[22] Wing, D. TrueSNAP Shutter Freezes Fast-Moving Objects. Available online: http://ericfossum.com/Articles/Cumulative%20Articles%20about%20EF/truesnaparticle.pdf (accessed on 3 March 2017).

[23] Clarke, T.A.; Fryer, J.G. The development of camera calibration methods and models. *Photogramm. Rec.* **1998**, *16*, 51–66.

[24] Zhang, Z. A flexible new technique for camera calibration. *IEEE Trans. Pattern Anal. Mach. Intell.* **2000**, *22*, 1330–1334.

[25] Huber, P.J. Robust estimation of a location parameter. In *Breakthroughs in Statistics*; Springer, New York, NY, USA, 1992; pp. 492–518.

[26] Horn, B.K.; Hilden, H.M.; Negahdaripour, S. Closed-form solution of absolute orientation using orthonormal matrices. *JOSA A* **1988**, *5*, 1127–1135.

[27] Engel, J. DSO: Direct Sparse Odometry. Available online: https://github.com/JakobEngel/dso (accessed on 3 March 2020).

3. BUILDING RECONNAISSANCE IMAGE MAPPING AND LOCALIZATION FOR LARGE SCALE INSPECTION MISSION

Natural hazard events remain a significant challenge to the engineering of our buildings. To reduce losses and improve safety, engineers must exploit each natural hazard event as an opportunity to observe and learn about the built environment for the purpose of improving the standards and guidelines that regulate their design. Image collection plays an indispensable role in supporting these post-event reconnaissance activities. Perishable data about building performance must be collected as quickly as possible. Photos and video are the preferred method because they can be acquired rapidly in the field. Teams of engineers travel to the site, identify structures that are relevant to the scientific questions they are most interested in, and collect large quantities of image data as they walk through those buildings.

GPS metadata is a common approach to get spatial information for images. However, this method only works in outdoor environments. In an indoor environment, GPS cannot provide accurate indoor location data [1]. To address this issue, I have previously developed a technique to localize reconnaissance images on a single structural drawing [2]. I used visual odometry (hereafter, VO) to reconstruct the walking path and associate it with the visual data. In this task, the step of overlaying the reconstructed path onto a drawing required manual user input, which is not preferred [2]. To overcome this limitation, here I develop the ability to entirely automate these steps, and evolve the single-floor image localization process into a fully automated multi-building, multi-floor image localization capability. The technique developed herein has three distinct advantages over manual human data organization. First, automation will save considerable time and human effort, especially when the mission involves numerous buildings, each having several floors. Second, the final overlaid result will have greater consistency in quality and fewer errors as the user is removed from the process along with the potential for human error when it comes to such a tedious and repetitive task. Third, because I automatically separate the data floor by floor, and building by building, and link them to the respective structural drawings, the availability and use of such image data will be accelerated, empowering engineers to improve the safety of our built environment to disruptions caused by natural hazard events.

In this task, a fully automated technique is developed to provide indoor localization. This technique requires no prior information about the condition or spatial layout of the indoor

environment. Moreover, this technique only requires the data collector to wear an additional inexpensive motion camera, and does not require costly equipment. Thus, it does not add time or effort to the current rapid reconnaissance protocol. During the reconnaissance missions, the data collector is able to walk through multiple buildings for inspection and data collection. Each of the building may contain multiple floors being inspected. Along the path, a video is recorded in the same time collecting the images. Together with the structural drawings, the video and the images are the large scale inspection data for this task. The data is firstly separated based on the individual floor that they are covering. This is done by applying a series of strategies involving an indoor-outdoor classifier, an unsupervised cluster and a visual odometry algorithm. After separation, the data belonging to each floor are processed one after another. The video clip is used to build a 3D path and point cloud using visual odometry technique. The path and point cloud are overlaid onto the structural drawing by solving an optimization problem. Then images are localized onto the structural drawing through time comparing between the images and the video clip. In addition, 3D reconstructions are built centering each image containing components-of-interest. The outcome of this task is a tool that automatically provide the indoor localization and a local 3D texture model for each inspection image.

The remaining sections of this paper are organized as follows. Section 3.1 reviews the research relating to this task. Section 3.2 explains the technical approach and the key challenges encountered and overcome, mainly focusing on data separation and the automated overlay process. In Section 3.3, experimental validation of the individual components of the technique is performed, including indoor-outdoor image separation, multi-floor separation, and Path overlay. I then validate the entire technique with data collected over a large-scale area. This chapter is adapted from the published work of the author [3].

3.1 Literature Review

Although to date no research has focused on tackling this problem, here I summarize past research conducted to look at tasks that are somewhat similar to those that I brought together to solve this problem. Researchers have considered indoor localization, projecting Pcl onto 2D surfaces, and nature-inspired optimization algorithms for a variety of purposes. Studies on indoor localization have been focused on accessing and sometimes integrating data from various sensors. These sensors include infrared cameras, Bluetooth, Wi-Fi, and radio-frequency identification [4-

9]. However, as these approaches rely on measurements between mobile devices and fixed landmarks, these methods are not suitable for deployment in post event reconnaissance. Post event building reconnaissance teams must operate without local electric or telecommunication services. Researchers have also explored using feature matching to localize images. Li et al. [10] match newly collected images to images in a data set by reading their geotags to provide localization. Geo tags would be GPS coordinates in the outdoor environment, or location IDs (e.g., room IDs) in an indoor environment. This approach requires time and effort to prepare the data set with geotags before the actual mission, and thus is of very limited use in rapid reconnaissance missions. Potentially, indoor place recognition [11-13] could be adapted to support this problem. However, the main limitation is again that the recognized scenery alone cannot provide localization results, and still requires some prior reference information, for example, prerecorded geotags and beacons. Furthermore, this technique cannot uniquely localize indoor components with identical or similar appearance, which is of course quite common in buildings.

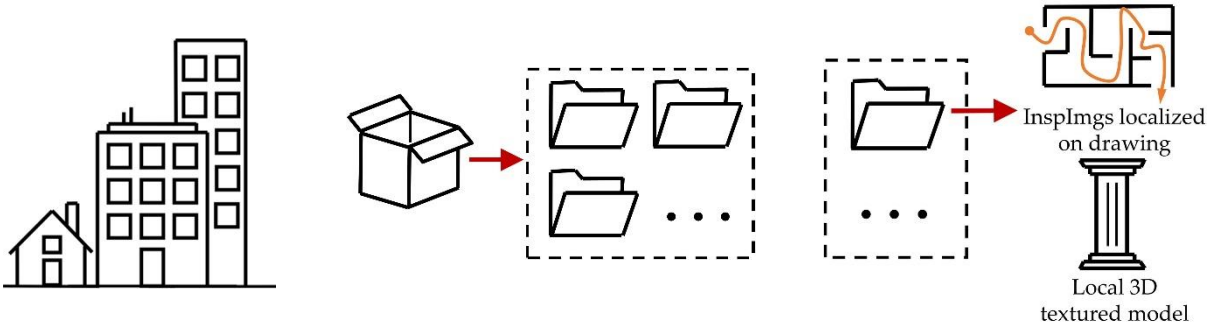
Linking or projecting a Pcl onto a 2D surface is sometimes needed. Work on this topic has mostly considered outdoor scenarios. Kaminsky et al. [14] formulated an optimization problem for carrying out this task using two cost functions based on the Pcl and Ray models, respectively. Then a grid search–based method was adopted to find the optimal overlay. This method may also leverage a GPS signal to improve performance. Because a Ray model is mainly for structure-from-motion (hereafter, SfM) models that span a limited region, it is not suitable for reconnaissance data collection where the environments normally consist of hallways and are visited just once. Based on these constraints, VO, or simultaneous localization and mapping (hereafter, SLAM) is chosen here over SfM because it is less time consuming for generating the Pcl and can directly provide Path results. Also, the grid-based search method developed can take time. While these factors limit the use of this particular method for reconnaissance data, the formulation of the Pcl cost function and coarse to fine search logic has inspired this work. In other past work, Ni et al. [15] use Hough transformation and scan match to perform the overlay of Pcl onto Google maps. They detect plane surfaces such as wall elements from the Pcl, and try to match them with lines on the map. This requirement inevitably limits the use of the method. In an indoor environment, wall elements are normally featureless, and through approaches such as VO/SLAM/SfM, walls are reconstructed as regions with no points or highly sparse points. This characteristic could lead to failure in detecting wall elements, and correspondingly, the implementation of this method. Furthermore, it would not

be useful for large open indoor spaces with no walls. Alternately, Zhang et al. [16] use edge detection to refine the overlay of building roof onto satellite images. This method is for improving existing results, not for achieving an initial overlay.

Another topic that has been considered by researchers is nature-inspired optimization algorithms. Genetic algorithms (hereafter, GA), introduced by John Holland in the 1970s [17] are inspired by the principles of genetics. Evolving over a number of generations, better genomes will survive over weaker ones and lead to optimal solutions for a given problem. Particle swarm optimization (hereafter, PSO) invented by Kennedy and Eberhart [18] in the 1990s is inspired by the motion of swarms of birds. It considers a group of randomly generated solutions and propagates them toward the optimal solution based on information shared by all members of the group. Other nature-inspired optimization algorithms have been developed, including Ant Colony Optimization inspired by foraging behavior of ants, Bat Algorithm inspired by the echolocation ability of bats, and Spider Monkey Algorithm inspired by the social behavior of a South American species [19-21]. Comparisons among these have already been made, and serve to guide researchers in choosing the most suitable algorithms. Hassan et al. [22] compared PSO and GA over eight benchmark problems, and drew the conclusion that PSO and GA yield the same level of solution quality, while PSO is generally more computationally efficient than GA. Tharwat and Schenck [23] also performed a comparison where a total of five algorithms are compared in terms of their performance on six benchmark problems. Based on a review of this past work, I adopt PSO for this overlay problem for its quality, robustness, computation efficiency, and widespread availability.

3.2 Technical Approach

3.3



Stage T2-1. Data collection

Stage T2-2. Data separation

Stage T2-3. Data processing

Figure 3.1. Overview of the technical approach

An overview of this automated technique is shown in Figure 3.1. The technique has three stages, including data collection, data separation, and data processing, each with its own challenges, which I will discuss here. The first stage is data collection. Engineers collect reconnaissance data over a large-scale area as the inputs to the technique. In a large-scale area, engineers will walk through multiple independent buildings to collect the visual data, and in each building, multiple floors may need to be visited for data collection. For example, the data collection may cover one floor in the first building, three floors in the second building, and so forth. There is no limitation regarding the number of floors, the number of buildings, or the order of the buildings to be covered in a given reconnaissance mission. The reconnaissance data include InspImgs, PathVideo, and structural drawings for all the floors in each building visited in the mission. InspImgs are the primary images collected during a mission, and are intended to document the structural condition of the building and the evidence of the consequences of the hazard event. At the same time, PathVideo is collected to store the scenes visible in front of the engineer as the mission takes place. Structural drawings may also be stored in advance as digital images with distinguishable file names, such as building1floor1, building2floor3, and so forth.

The second stage is data separation. I aim to separate the data according to the individual floor on which they were collected. After separation, the data belonging to a single floor will all be collected in one folder. This process is mainly driven by the separation of PathVideo, by exploiting indoor–outdoor classification, clustering, and Path reconstruction. After that, I will

obtain PathImgs according to the individual floor and put them in different folders. Following the separation of PathImgs, the InspImgs are put to the corresponding floors by timestamp matching between InspImgs and PathImgs. And structural drawings are simply arranged by their file names.

The third stage is to process the data that are stored in a single folder to generate the indoor locations of InspImgs and then localize them on structural drawings and repeat the process for each of the folders. Thus, using the data in one folder, I apply VO to PathVideo and create PathPcl. These results are automatically overlaid onto the structural drawing by solving an optimization problem. The locations of InspImgs are obtained by referring to their pairing with PathImgs, which are matched using timestamps. And a selection of InspImgs and PathImgs near any highly inspected location may be used to generate a local texture 3D model. In the end, the locations of InspImgs on the structural drawing and local texture 3D models are the output and provided to the engineers.

3.3.1 Data Separation

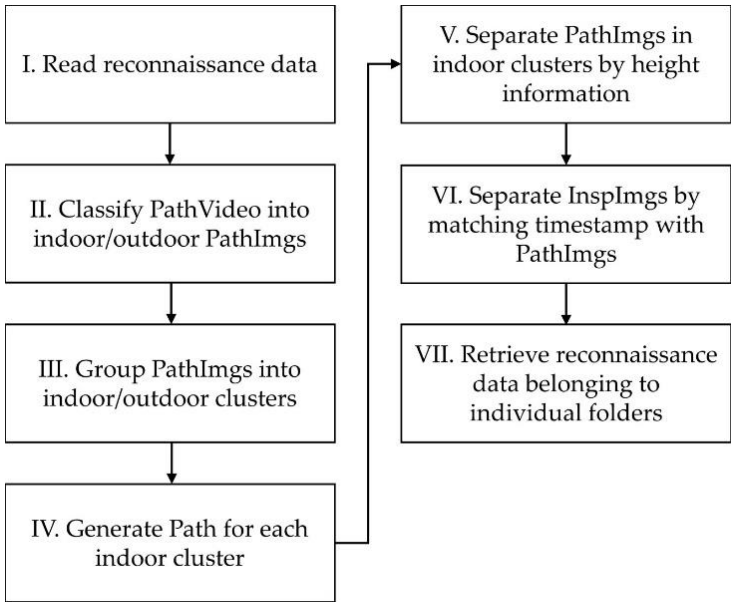


Figure 3.2. Workflow for completing the data separation stage

In this section, the details of this data separation method, Stage T2-2 in Figure 3.1, are explained. This stage is to separate the reconnaissance data into folders according to the individual

floor they belong to. All data belonging to one floor are collected in one folder. The procedure is shown in Figure 3.2. Note that all steps in this stage are completely automated. To begin with, the reconnaissance data are read in step T2-I, including InspImgs, PathVideo, and structural drawings. Frames of PathVideo are stored as PathImgs, and named with corresponding indices. Each PathImg is assigned with a timestamp by interpolating between the beginning time and the end time of PathVideo. InspImgs are also stored with their timestamp. The structural drawings are read as digital images, and named based on the building index and floor index, for example, as "building1floor1." With these file names assigned, they are directly put into the corresponding folders.

For step T2-II, a two-class image classifier is designed using CNNs. This classifier intends to distinguish indoor images from outdoor ones, and only needs to be trained just once before processing the data. Instead of generating labels, the classifier is used to assign each of the PathImgs with a probability ranging from 0 to 1, where a number closer to 0 indicates a higher chance of being an indoor image, while values closer to 1 are for outdoor images.

In step T2-III, I aim to group the PathImgs according to their indoor or outdoor labels or probabilities from step T2-II. Each PathImg is treated as a 2D point with the image index being the x coordinate and the probability being the y coordinate. After removing ambiguous points with probabilities between .1 and .9, the left points are grouped using an unsupervised cluster based on 2D Euclidean distances between each other. For any group, if all the images in it are enclosed in any other group based on the upper bound and lower bound of the image index, then it is absorbed by that clustered group. At this point, the indoor-outdoor separation is finished. Each remaining group represents PathImgs taken inside one particular building or taken during an outdoor passage between different buildings.

In step T2-IV, Path is generated for each indoor group using VO technique [24]. The path that the data collector takes through the building is rebuilt, including how she or he walks within floors and across a particular floor. Climbing between floors through stairwells is captured because it results in coordinate changes in the height dimension in 3D. This direction is recognized as being perpendicular to the ground surface in the 3D coordinate system.

In step T2-V, for images in each indoor group, Path is divided into segments based on the height information. Each segment thus corresponds to Path formed on one single floor. By tracing back to PathImgs through the indices, I obtain PathImgs taken at each floor level.

In step T2-VI, InspImgs are related to PathImgs by comparing their timestamp, and the images from both sides with the nearest timestamp are considered taken in the same physical location or the same floor. By addressing the corresponding PathImgs, InspImgs are localized to floors they are collected from.

Then in step T2-VII, I can retrieve the reconnaissance data that are already separated into a number of folders, and inside of each folder, it contains inspection data for a single floor, as PathImgs, InspImgs, and the structural drawing.

3.3.2 Path Overlay in Data Processing

Stage T2-3 in Figure 3.1 is data processing where I process data in each folder to generate the indoor locations of InspImgs and visualize them on the respective structural drawing. The process includes using VO to rebuild PathPcl, automatically overlay the reconstructed PathPcl onto the structural drawings, and perform timestamp matching. The details of Path overlay are discussed in this section, and PathPcl reconstruction and timestamp matching will be explained later in the validation section.

Here I develop a method to automatically carry out the overlay without any manual assistance. To achieve this goal, an optimization problem is defined such that the solution gives the optimal Path overlay on the structural drawing. The optimization problem is formulated by minimizing the value of a cost function to determine several unknown parameters that define the overlay position of PathPcl. The cost function encodes a quantitative representation of how well PathPcl is overlaid onto the structural drawing. The search process to obtain the optimal combination of these parameters is designed to be practical, in that, it does obtain a useful and valid result quickly. Formulating the overlay problem as an optimization problem requires that one take into account the complexities of structural drawings, as well as the overall goal and what type of result is acceptable. Because this task is dealing with multiple floors and multiple buildings, all of which need to be identified, and then the overlay of each of these must be achieved.

It is worth mentioning that without defining the cost function in the following section, an overlay result can only be evaluated by human judgment as to whether or not it is properly overlaid on the drawing. To do that, a human would simply focus their attention on meeting two goals. The first is matching the shape of Pcl with the markings that define the structural elements in the drawing. The second is to guarantee that the Path object falls in an empty area in the structural

drawing, specifically within the passage the engineers take in the hallways. These two goals inspire this approach and are thus encoded into the cost function discussed next.

3.2.2.1 Cost Function Formulation

The cost function is defined to quantitatively evaluate the quality of the overlay result. Thus, to define the cost function, I must model the overlay process. For generating the model, the markings on the structural drawing are considered to be fixed and impenetrable (i.e., the walking path cannot penetrate walls and columns). I must first transform PathPcl from its original arbitrary coordinate system to the coordinate system of the structural drawing. In the overlay model, PathPcl and the structural drawing are the known data. The unknown variables to solve for are the set of the parameters needed to perform a 2D affine transformation for PathPcl. The parameters include a translation in the x-direction, a translation in the y-direction, a rotation angle, and the scale. These are denoted as t_x,t_y, θ, and s, respectively. Collectively, I denote all the parameters to be tuned by $\phi = (t_x, t_y, \theta, s)$.

The coordinate transformation for a point in PathPcl is defined as

$$p'(p;\phi) \coloneqq \begin{bmatrix} s\cos(\theta)\, p_x - s\sin(\theta)\, p_y + t_x \\ s\sin(\theta)\, p_x + s\cos(\theta)\, p_y + t_y \end{bmatrix} \tag{3.1}$$

where p is a point from either Path or Pcl, p_x and p_y are its x and y coordinates in the original coordinate system, and $p'(p;\phi)$ is the transformed point with the corresponding coordinate in the structural drawing coordinate system.

Combining the input structural drawing and PathPcl with transformation parameters ϕ, I can define the cost function. The cost function is formed as a combination of two terms, based on Path and Pcl of PathPcl, respectively.

The term in the cost function related to Path is defined as

$$C_{\text{Path}}(\phi; D) = \frac{1}{N_{\text{Path}}} \sum_{p \in P_{\text{Path}}} B(p'(p;\phi), D) \tag{3.2}$$

where P_{Path} is the set of points in Path, p is a point in P_{Path}, and $p'(p;\phi)$ is the transformed point of p. D is the image of the drawings. $B(p', D)$ is the intensity value of binary D at the pixel whose 2D coordinates are the ones of point p'. If the value is 0, it means that point p' hits a white pixel on the binary image of the structural drawing, and 1 means point p' hits a black pixel. When Path

is optimally, or even acceptably, overlaid, all or most of the points along Path should encounter white pixels since Path must be placed in regions with no structural components and open to passage. And, N_{Path} is the number of the points in P_{Path}.

The second term in the cost function related to Pcl, and is defined as

$$C_{\text{Pcl}}(\phi; D) = \frac{1}{N_{\text{Pcl}}} \sum_{p \in P_{\text{Pcl}}} E(p'(p; \phi), D) \tag{3.3}$$

where, similarly, P_{Pcl} is the set of points in Pcl, p is a point in P_{Pcl}, and p' is the transformed point of p. $E(\,)$ is the Euclidean distance transform (hereafter, EDT) [25] of the structural drawing as a binary digital image. And $E(p')$ is the value of the EDT at the pixel whose 2D coordinates are associated with point p'. EDT is a mapping method for a digital image where, for each pixel, EDT stores the Euclidean distance from this pixel to the closest pixel measured by such distance. In this problem, EDT only serves as a query table for analyzing and storing the Euclidean distance between Pcl points and SDI pixels. EDT is computed just one time before the search is executed. During the search, I directly query EDT for the distance information instead of repeatedly visiting the SDI. This approach greatly boosts the speed required to solve the optimization problem. Again, if Pcl is optimally, or even acceptably, overlaid on the structural drawing, the EDT mapping for most of the points in Pcl should be 0. Here, N_{Pcl} is the number of points in P_{Pcl}.

The final cost function is defined as

$$C(\phi; D) = \begin{cases} \alpha \cdot C_{\text{Path}}(\phi; D) + C_{\text{Pcl}}(\phi; D), \text{if } \phi \in \Phi \\ C^1_{\text{penal}}, \text{if } \phi \notin \Phi \\ C^2_{\text{penal}}, \text{if } p'(p; \phi) \notin \Phi(t_x, t_y), \text{for any } p \in P_{\text{Path}} \end{cases} \tag{3.4}$$

where Φ is the set of parameters that are bounded to yield a reasonable overlay, and how to retrieve the exact Φ for a Path overlay will be discussed in Section 3.2.2.2. When ϕ belongs to Φ, the cost function, $C(\phi; D)$, equals the combination of the two cost function terms defined above, while ϕ falls outside of Φ, I simply set C^1_{penal} to $C(\phi; D)$, which is a penalty value set to 1100 in this task. And when a transformed Path point exists, which is out of the bounds of t_x and t_y (really, outside of the building plan), I set another C^2_{penal} to $C(\phi; D)$, which is a penalty value set to 2200 in this task. The reason to set two different penalty values is to simply keep track of the cases when a penalty is applied. α is a coefficient used to provide a relative weighting between the two terms. This coefficient is set as the ratio of the number of Path points to Pcl points to balance the

$C_{\mathrm{Path}}(\phi; D)$ and $C_{\mathrm{Pcl}}(\phi; D)$. Together with the factors including the VO algorithm used in this task, Pcl filtering as explained in the next section, and so forth, α is set as 0.1 in this task to provide the best overlay results, although the method is not sensitive to this parameter. By minimizing the value of C, I obtain the values of the variables in ϕ that correspond to the optimal overlay result. Note that I acknowledge the fact that it is possible that the hallways in a given structure will be wide enough that there are several adjacent positions for Path that are equally acceptable, and any of these would be an acceptable choice.

3.2.2.2 Search Strategy

To avoid being trapped by local minima, I seek to form a search strategy that is highly likely to yield the global minima. Given the design of this problem, it is guaranteed that at least one optimal solution exists for this optimization problem, which corresponds to the optimal overlay result, and thus I can find a set of the variables that give the optimal overlay result. This optimal result must exist within the range of the structural drawing, as PathPcl are overlaid onto the structural drawing. Thus, among the large but finite number of overlay results, I form a derivative-free method to search for the best values of the variables and to obtain such a result quickly. This search strategy is an adaptation of the original PSO method. Compared to PSO, which would be likely to become stuck in a local minimum in this problem, this method is able to achieve the global minimum with high robustness (this will be demonstrated in Section 3.3.1.3). In the next paragraphs, I explain this approach, and in the last paragraph of this section, I briefly discuss PSO and how PSO is integrated into this method.

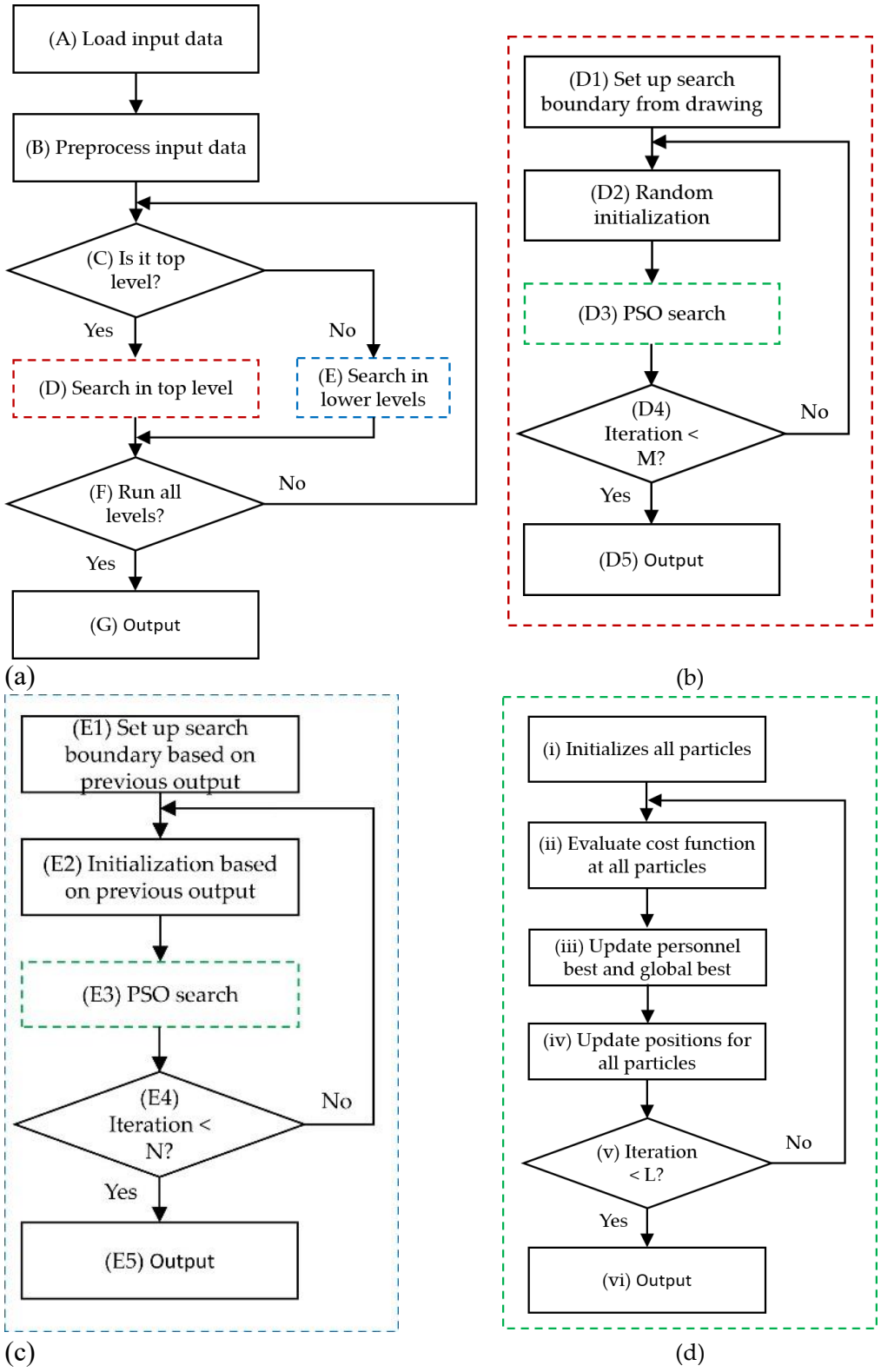


Figure 3.3. Workflow of the search strategy: (a) Overall workflow, (b) Detailed workflow of search in the top level, (c) Detailed workflow of search in lower levels, (d) Detailed workflow of PSO search.

The front-end workflow of the search strategy is shown in Figure 3.3a. Step (T2-A) is to load the input data of a structural floor, including PathPcl and SDI. A series of preprocessing procedures are applied in step (T2-B): (T2-B-1) transform SDI to a binary image, as binary SDI; (T2-B-2) generate a multi-level image pyramid of binary SDI [26]. The total level in the image pyramid is denoted as T (how to determine the value of T will be discussed in the validation section). This step is to obtain T copies of binary SDI with different sizes. Level 0 is the original binary SDI (the finest level), and level $T-1$ is the highest level (the coarsest level). Each level has half as many columns and rows of pixels as were in the previous level by smoothing the pixel intensities in the neighborhood. (T2-B-3) Filter the points in Pcl by their height coordinate values, the coordinate axis perpendicular to the SDI. Along the height coordinate, the points at the center area in the nearby region are kept. All of the preprocessing steps in step (T2-B) are meant to shorten the search process to a reasonable time. Then the following steps are used to perform the search over the image pyramid. The search starts in the highest level (level $T-1$) and moves down until reaching level 0 [14]. Step (T2-C) is a judgment of whether the search is going to be in the top level of the image pyramid, which has the smallest copy of binary SDI. If the answer is yes, it goes to step (T2-D), which applies the search at the top level. And if the answer is no, it goes to step (T2-E) to carry out the search at the lower levels. Then, step (T2-F) is to check whether the process has gone through all levels. If no, it will continue until it reaches level 0, and if yes, it proceeds to the final step, step (T2-G), where I obtain the output, the specific values of the variables ϕ yielding the optimal overlay result.

The details of step (T2-D), to search in the top level, are shown in Figure 3.3b. Data passed from the previous steps are preprocessed input data, with indices indicating that these data are for processing at the top level of the image pyramid. Step (T2-D1) is to set up the search boundary for all of the variables (t_x, t_y, θ, s). For each one of the four variables, a lower bound and an upper bound are generated. These two bounds govern the range of possible values for that variable, and when the value is outside of these bounds, a large penalty is applied in the cost function for that candidate (see Equation (3.4)). In this step, all bounds are set based on the top level in the image pyramid of binary SDI. If the binary SDI at the top level is treated as a 2D matrix, the indices of the far left and far right columns containing less than 1% black pixels are automatically set as the lower bound and the upper bound of t_x, respectively. In the same way, t_y are set up based on the indices of the rows. θ is simply 0 and 360 degrees. For s, I calculate the Euclidean distance of

each black pixel in binary SDI in the top level from the origin, and get the standard deviation of the distances, Std_{map}, and for all the points of Path of PathPcl, Std_{path}. The ratio between Std_{map} and Std_{path} is regarded as s_{initial}. Then, the lower bound is chosen as 50% of s_{initial}, and the upper bound is chosen as 120% of s_{initial}. Step (T2-D3) is to apply PSO search [18] to find the global minimum of the cost function, as designed in Section 3.2.2.1. And in step (T2-D4), a loop is carried out to repeat steps (T2-D2) to (T2-D3). The purpose of this loop is to compensate for the random initialization of PSO, and this approach is demonstrated to greatly increase the chance of generating the desired output. This procedure will be further discussed in Section 3.2.2.3. When the iteration meets its preset limit, M, the process goes to step (T2-D5), which ends the search at the top level and gives the output of the search at this level. The method used to determine M will be discussed in Section 3.2.2.3.

Once a combination of variables is obtained through the search at the higher levels, starting from the top level, the search at the lower levels focuses on a small region based on the available outputs, as in Figure 3.3c. Step (T2-E1) is to set up the search boundary based on the outputs from the previous level. In particular, I set 80% and 120% of the output value of each variable as the lower bound and upper bound for the current level, respectively. The search in the current level considers only options within these boundaries. Step (T2-E2) performs the initialization from the previous output, where it takes $(2 \cdot t_x', 2 \cdot t_y', \theta', 2 \cdot s')$ as the initialization in the current level, and $(t_x', t_y', \theta', s')$ are the outputs from the previous level. Then PSO search is used to search for the global minimum in step (T2-E3). And then step (T2-E4) is used to check whether the loop meets the iteration limit, N. Compared to M, N is a small number. In this task, N is set to 10. After N iterations, the process gives the output in step (T2-E5).

The PSO algorithm used in step (T2-D), search in the top level, and step (T2-E), search in the lower levels, is shown as in Figure 3.3d. In PSO, there are a group of candidates, which is referred to as a swarm of particles. All candidates will have their own initial guesses for the variables (or denoted as positions in PSO) simultaneously and independently. These guesses are not required to have the same values. Each candidate follows a unique trajectory of searching, including initializing the variables and updating them. To start, step (T2-i) is to initialize all variables for each particle by giving them some values. Unlike the traditional PSO, here these values are inherited from step (T2-D2), random initialization, or from step (T2-E2), initialization based on output from the previous level. In step (T2-ii), I evaluate the cost function at the positions

of each particle, yielding the corresponding values of the cost function. Then in step (T2-iii), I compare these values with the personnel best for each particle, and keep the smaller value of the cost function as the updated personnel best for that particle. I do the same comparison between these values and the global best, each time keeping the lowest one as the updated global best. The corresponding values of the variables are passed along with the personnel best and global best. Then in step (T2-iv), the positions of the particles are updated based on personnel best and global best from the previous step. More details on the update process are discussed in the next paragraph. In step (T2-v), I determine whether the iteration meets its limit, L, from step (T2-ii) to step (T2-iv). When the iteration limit is reached, the output is generated in step (T2-vi), which is the updated global best kept until now and its corresponding combination of variables. The values of the variables are the outputs.

In step (T2-iv) of Figure 3.3d, all particles update their positions. For instance, take a particle having the index i, with the total number of particles being P, and the iteration index is $k+1$ out of the iteration limit L. The updated formula is given as

$$V_{i,k+1} = w \cdot V_{i,k} + c_1 \cdot z_{i,k} \cdot \left(\hat{\phi}_{i,k} - \phi_{i,k}\right) + c_2 \cdot z_{i,k}^{+} \cdot \left(\hat{\phi}_{g,k} - \phi_{i,k}\right) \tag{3.5}$$

where $V_{i,k+1}$ is the update for this particle in the current iteration. It is a 4D vector reflecting the changes in the variables ϕ, and $V_{i,k}$ is the updated vector for the same particle from the previous iteration. Two random variables, $z_{i,k}, z_{i,k}^{+} \sim U(0,1)$. $\hat{\phi}_{i,k}$ is a variable vector corresponding to the personnel best of this particle up to iteration k, and $\hat{\phi}_{g,k}$ is the variable vector for the global best up to iteration k. $\phi_{i,k}$ is the current position of this particle; w, c_1, c_2 are three coefficients to balance different terms in the update. Then, the new position of this particle is calculated by adding the update vector to the previous position, as

$$\phi_{i,k+1} = \phi_{i,k} + V_{i,k+1} \tag{3.6}$$

The question yet remains as to how to determine the hyper-parameters used in the search, including M, L, P, w, c_1, c_2. This question will be discussed in Section 3.2.2.3.

3.2.2.3 Hyper-parameter Tuning

The search algorithm requires the hyper-parameters to be selected before the optimization is performed. The choice of the hyper-parameters may influence the reliability of the search

algorithm and its computation time. Thus, I use a simple method to tune the hyper-parameters and find appropriate values. I run the algorithm 100 times using data collected only on a single floor, and count the number of runs during which the algorithm reaches a threshold, which indicates that the algorithm yields acceptable results in one run. For each run, instead of deciding if the overlay result is acceptable or not through human effort, I compare the value of the cost function to a predetermined threshold. If the value obtained is larger than the threshold, it is considered as a failed run, otherwise, as a successful run.

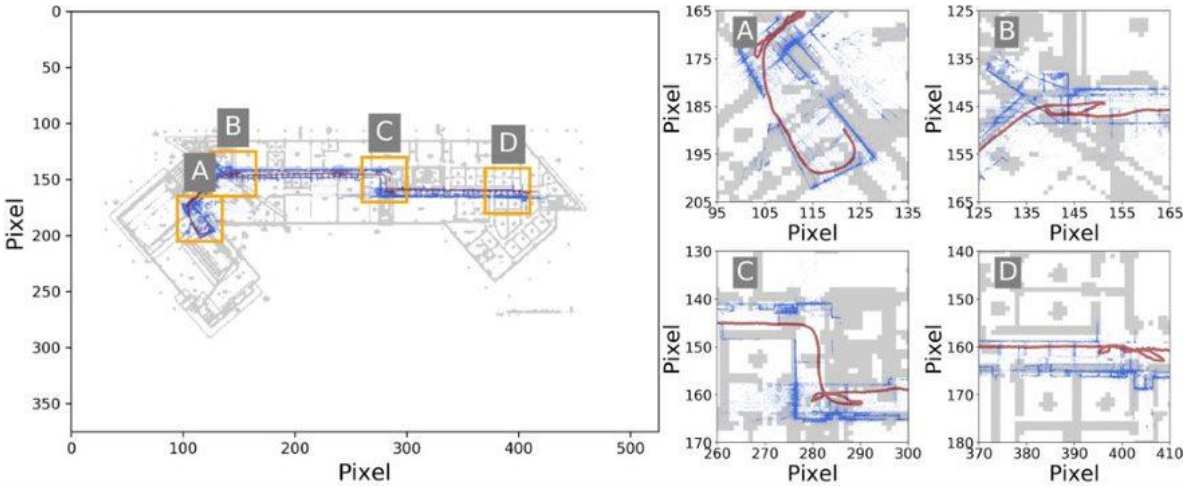


Figure 3.4. The overlay result with the global minimum of the cost function

I use the data collected in the underground floor in Armstrong Hall, Purdue, to tune the hyper-parameters. The data generate 1428 Path points and 489,930 Pcl points. To save time, I perform the hyper-parameter tuning only at the fourth level of the image pyramid. This adjustment shrinks the original image of the structural drawing from 8400 × 6000 pixels to 525 × 375 pixels. After using the algorithm (with a temporary hyper-parameter setting as $M = 50, L = 50, P = 50, w = 1.0, c_1 = 1.0, c_2 = 1.0$), the optimal overlay result at the fourth level is shown in Figure 3.4, which corresponds to the global minimum of the cost function. In this figure, the blue colored points correspond to the points in the Pcl, as the Pcl is rebuilding the visible environment along the path, including walls, doors, and so forth.

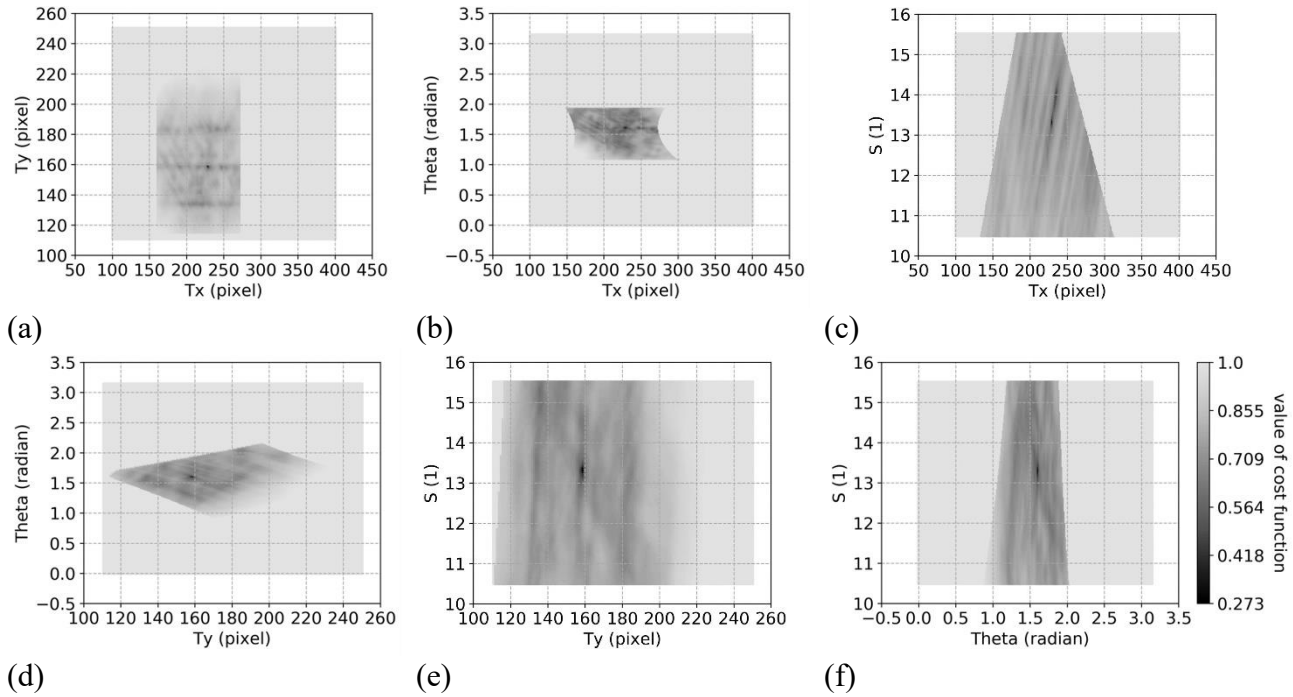


Figure 3.5. Results of validation by brute-force grid search

The next part of the process is to find the threshold. I use the brute-force grid search to evaluate the cost function with all possible values of unknown variables Φ. A brute-force grid search will examine all possible values of each variable within the search boundaries, as indicated in Section 3.2.2.2. Sample results from the brute-force grid search are shown in Figure 3.5. To illustrate the behavior of the cost function near the optimal overlay result, I plot the brute-force grid search over the values of any two of the variables while keeping the other two at the optimal values. Note that the results are plotted in log and rescaled for better visualization. As indicated by the color bar, if a point is plotted with a darker color, it means the cost function at that point is smaller and therefore, it is closer to the global minimum. Take Figure 3.5a as an example. For $t_x = 230, t_y = 158$, the point represents the global minimum. Notice in Figure 3.5a, the values of θ and s are set to achieve global minimum. This approach is simply for aiding visualization. As I move away from this point, the color of the points becomes brighter, which means that at those points, the cost function is becoming larger. It is obvious that in the region around the global minimum, there are scattered dark points compared to those in their neighborhood. These points represent local minima, and yield comparatively poor overlay results with respect to the global minimum. A key motivation for the development of this method is to avoid falling into a local

minimum. As mentioned in Section 3.2.2.2, any values of the variables that cause any portion of the PathPcl to stray out of the valid structural drawing boundary are not acceptable, and the cost function is accordingly assigned a large penalty value. These outcomes correspond to the gray background in each plot. It is easy to imagine that outside the gray region, the values of the variables lead to a cost function with such a penalty. So, there is no need to search in those areas. From these results, it is easily to see that the global minimum of the cost function corresponds to the overlay result. In addition, it is the sole point where the cost function reaches the global minimum. Thus, I use this global minimum and the corresponding value of the cost function (0.273) as the criteria in the evaluation discussed next.

Table 3.1. Candidate values of hyper-parameters

Hyper-parameter	Candidate values
M	10, 20, 30, 40, 50, 100, 150, 200, 250
L	10, 20, 30, 40, 50, 60, 70, 80
P	10, 20, 30, 40, 50, 60, 70, 80
aw	0.5, 0.8, 1, 1.2, 1.5
c_1	0.5, 0.8, 1, 1.2, 1.5
c_2	0.5, 0.8, 1, 1.2, 1.5

After determining the threshold, the hyper-parameter tuning can begin. Candidate values of the hyper-parameters are listed in Table 3.1. To save time, I perform two tunings. Since it is obvious that the accuracy rises when M is increased, the first tuning is performed at a fixed M, set at 10, while L, P, w, c_1, c_2 are tuned. After the first tuning finishes, I pick the hyper-parameters with the highest accuracy, and tune the value of M on top of that until a good accuracy is achieved. The results are shown in Figure 3.6. The results from the first tuning are plotted in blue color (with $M = 10, L = 50, P = 80, w = 0.5, c_1 = 1.5, c_2 = 1.5$). The results of the second tuning are then plotted in orange, and it is apparent that the accuracy grows along with the value of M. In the end, I choose the first set of hyper-parameter values that reach an accuracy of 100%, which is $M = 200, L = 50, P = 80, w = 0.5, c_1 = 1.5, c_2 = 1.5$, and these are the values used in the final validation of the entire technique. This result also demonstrates that with the selected hyper-parameter values, this search method has a high likelihood of obtaining the optimal overlay result.

Note that the objective here is to find an optimal overlay result for purposes of visualizing Path and linking it to the building locations visited by the field engineer when collecting data, instead of the optimal overlay result in the precise mathematical way.

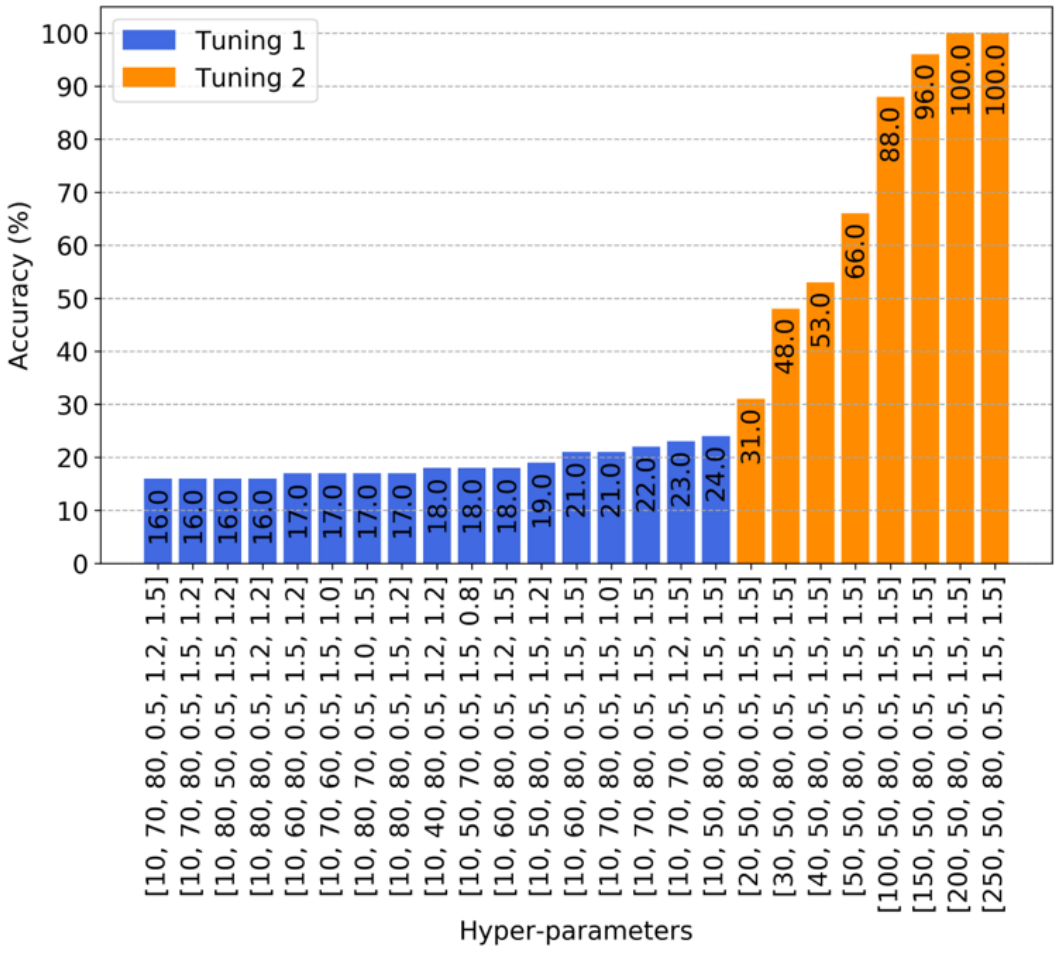


Figure 3.6. Results of hyper-parameter tuning

3.4 Experimental Validation

The validation is divided into two parts. First, I individually verify each of the major steps in this technique. These steps include indoor-outdoor separation, multi-floor separation, and Path overlay, which are tested separately with independent data collected from actual buildings. The focus of these sections is on explaining the implementation details, generalizing the methods for broad applicability, and verifying each with several sample datasets.

Next, to emulate a real reconnaissance mission, I collect image data covering a large-scale area. Data are collected using the recommended procedures while walking continuously through three buildings. Details are given for each of the buildings and floors used for this validation, as well as for the devices used, their configuration, and the lessons learned in this process. The results are provided to illustrate the method and type of results that are obtained.

3.4.1 Verification of Essential Steps

3.3.1.1 Verification of Indoor and Outdoor Separation

As explained in Figure 3.2, step T2-I to step T2-III, I must process PathVideo and separate PathImgs into indoor and outdoor groups. This function is the first key component of this technique. Here I discuss the design of the indoor-outdoor image classifier, and the verification of the indoor and outdoor separation with several test data.

3.3.1.1.1 Classifier Design

The indoor-outdoor classifier is designed to classify each PathImg as being either in the indoor or outdoor category. These two categories are defined as follows: (1) indoor images—the context of these images is indoor environments. Indoor objects are likely to be present in these images, for example, walls, doors, corridors, staircases; and (2) outdoor images—the context is outdoor environments, which are formed by elements, for example, pavement, trees, grass, façades of buildings, vehicles. Outdoor images are the negative of indoor images. Regarding the two categories, I build a training and testing data set based on the data set organized and labeled manually by the authors, as well as other published data sets. Some sample images and the number of images I used from each data set are listed in Figure 3.7. Data sets used in the training and testing include CDSE, SUN, DOIDE, and Indoor scene [27-30]. Images from the two classes are not equally selected from each data set. Instead, I choose images that are correctly labeled and with no ambiguous visual contents. However, the total number of images in both classes are balanced (indoor: 11,583 images, outdoor: 11,078 images). This approach will help to avoid misclassification.

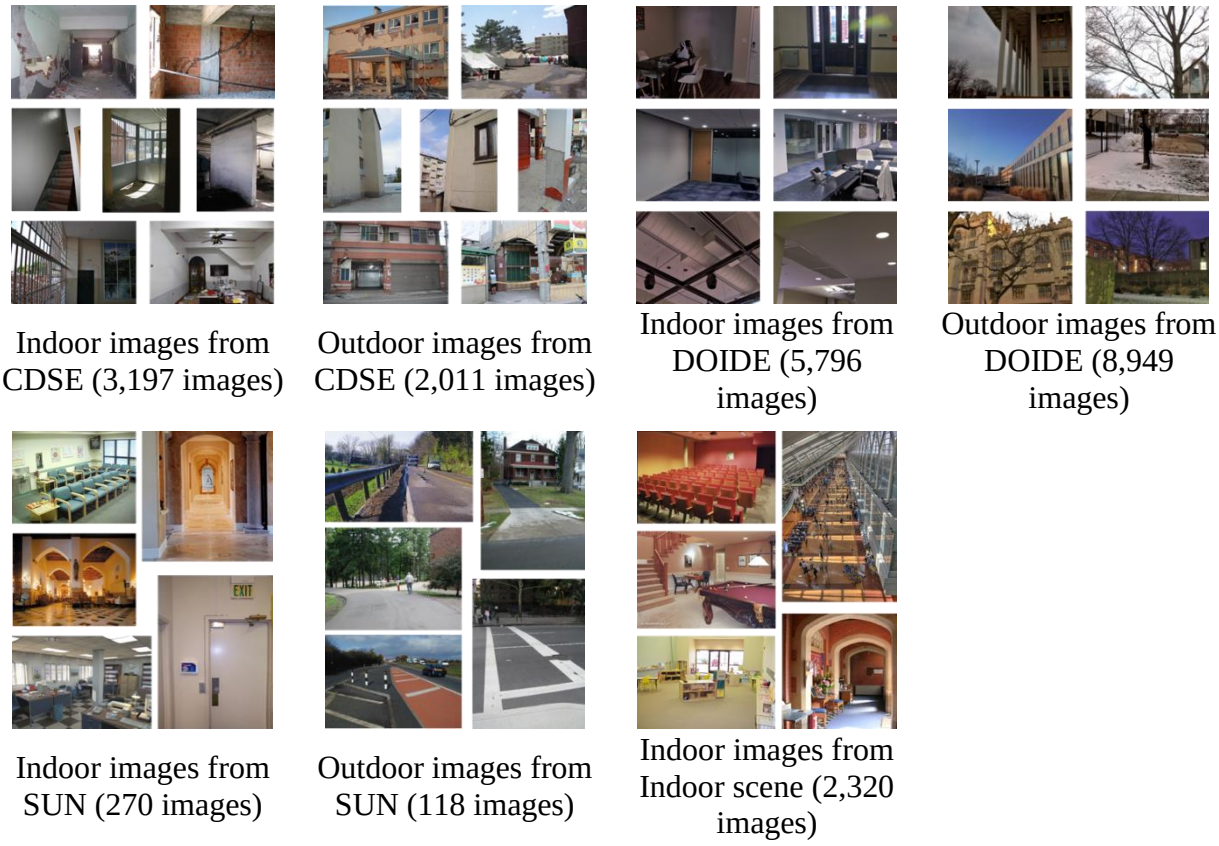
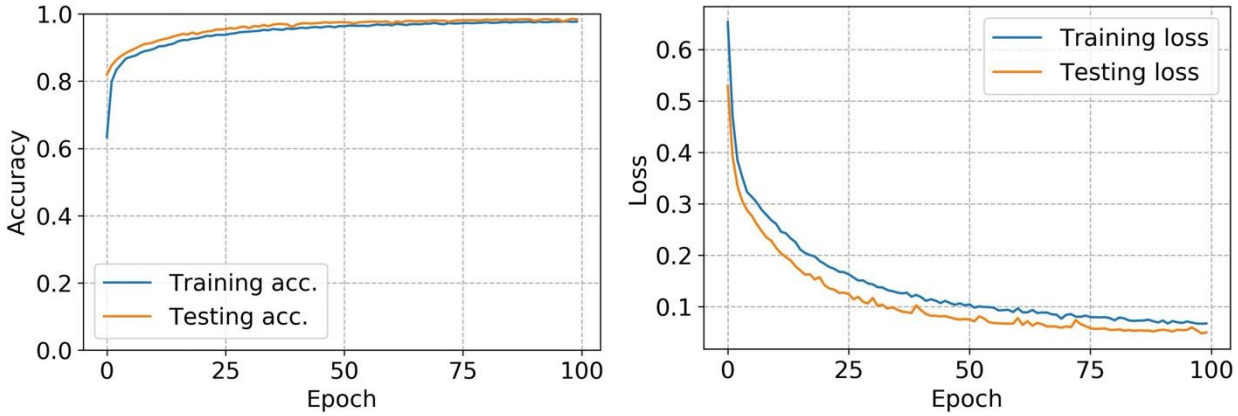


Figure 3.7. Sample images in the dataset [27-30]

The structure of the classifier is configured based on a popular CNN model, VGG16 [31]. This model performs among the best in the ImageNet competition in 2014, with high accuracy for classifying images into nearly 1000 classes. The five main convolutional blocks are kept, and the top block is replaced, since the output is binary, indoor or outdoor. To replace the original top block, a new top block is added after all of the convolutional blocks. This new block generates a probability between 0 and 1 for each image. A value closer to 0 means that the image has a high probability it is an indoor image. Otherwise, it is determined to be an outdoor image.

To train the classifier efficiently, I balance the training of new weights with the use of the pretrained network. I use the pretrained VGG16 weight trained with ImageNet data set [32]. During the training process, the weights of the first two convolutional blocks in VGG16 are fixed, and the latter three blocks are tuned. Together with the top block, the weights of the last three blocks are the only ones that are trained with the indoor and outdoor data set. The data set gathered and formed in the above is randomly separated into 80% for training and 20% for testing.

The training process is shown in Figure 3.8. The classifier is trained for 100 epochs. The accuracy of the classifier in Figure 3.8a rises rapidly to a high level within the first few epochs, then subsequently increases with a gentle trend. From the loss history in Figure 3.8b, the training process is clearly quite successful. Both the training loss and testing loss drop steadily in the first few epochs. The weights obtained after 100 epochs are used for the final classifier. The confusion matrix of this classifier on the testing data set is shown in Table 3.2. Clearly, this model achieves both high recall and precision in predicting indoor and outdoor classes.



(a) (b)

Figure 3.8. Training process of the indoor-outdoor classifier: (a) accuracy history, (b) loss history

Table 3.2. Confusion matrix of the classifier on the testing dataset

	Indoor pred.	Outdoor pred.	Recall
Indoor	2309	21	99.10%
Outdoor	53	2150	97.59%
Precision	97.76%	99.03%	

3.3.1.1.2 Results of Indoor and Outdoor Separation

To increase the confidence in assigning the images into indoor or outdoor categories, I also develop a method I call image separation. Indoor-outdoor image separation is needed to separate the indoor image groups from outdoor image groups, rather than entirely based on the classification result of every single image. To start the process, PathVideo is read and the frames are saved as PathImgs. The indoor-outdoor classifier then labels each PathImg, and the raw probabilities from the classifier are stored nstead of the labels. I remove PathImgs that have probabilities between .1 and .9, and leave all of those remaining. Then I apply an unsupervised cluster on the remaining

PathImgs. Here, I perform hierarchical clustering with the single linkage algorithm [33]. To perform the cluster, the data are treated as 2D points, and the clustering is based on the 2D Euclidean distance between the points. I scale the image indices by a scaling factor, which is roughly the total number of PathImgs. Here, I use 1000. This scaling factor is used to keep the values of probability and the image indices at about the same order of magnitude. The distance threshold for clustering is set to 0.1. Based on the raw clustering results, I remove any redundant clusters, which fall within other clusters. The remaining clusters are the final separation results.

I test this image separation approach on two data sets. They are collected with a motion camera, a GoPro HERO 8. The camera is set to 240 fps with all other options as default. Each frame is 1920 × 1080 pixels. Each of the two data sets is real footage recorded while the data collector is walking through one or more buildings. Both are mixed and contain indoor and outdoor passages. The first data set begins in a passage starting from the first floor of Knoy Hall on the Purdue campus, and then moves outside of the building, down in the alley between the ME building and the ECE building. To speed up the process, I use one PathImg from every 200; 1039 PathImgs are used. The raw probability results from the classifier are shown in Figure 3.9a. Here the x axis is the image index of PathImgs, and the y axis is the raw probability value. In the figure, each point corresponds to one PathImg. A few select PathImgs are shown in Figure 3.9c. From the plots, it is obvious that the basic trend of indoor and outdoor is captured. Most PathImgs are correctly labeled, as PathImgs with an index from 1 to 686 are PathImgs collected indoors, and after that, PathImgs are collected outdoors. Following the technical procedure mentioned previously, the final separation result is in Figure 3.9b. Different colors represent different clusters. There are two clusters in total, which is exactly as expected. By comparing the mean probability of each cluster with respect to .5, one readily associates the first cluster as indoors, and the second one as outdoors. Tracing from the clustered results back to PathImgs indices, I can easily determine the boundary between the two groups in the PathImgs. As shown in Figure 3.9c, the PathImg that begins each group is bounded by a green box (B), and the ending PathImg for each group is bounded by a red box (E). Notice that the ending PathImg for the indoor group and the beginning PathImg for the outdoor group are not immediately next to each other. This is because, to further avoid bad separation, I remove 10 PathImgs from the starts and the ends of each indoor or outdoor group to determine the best PathImgs to designate as starting and the ending except for the very first PathImg and the last PathImg.

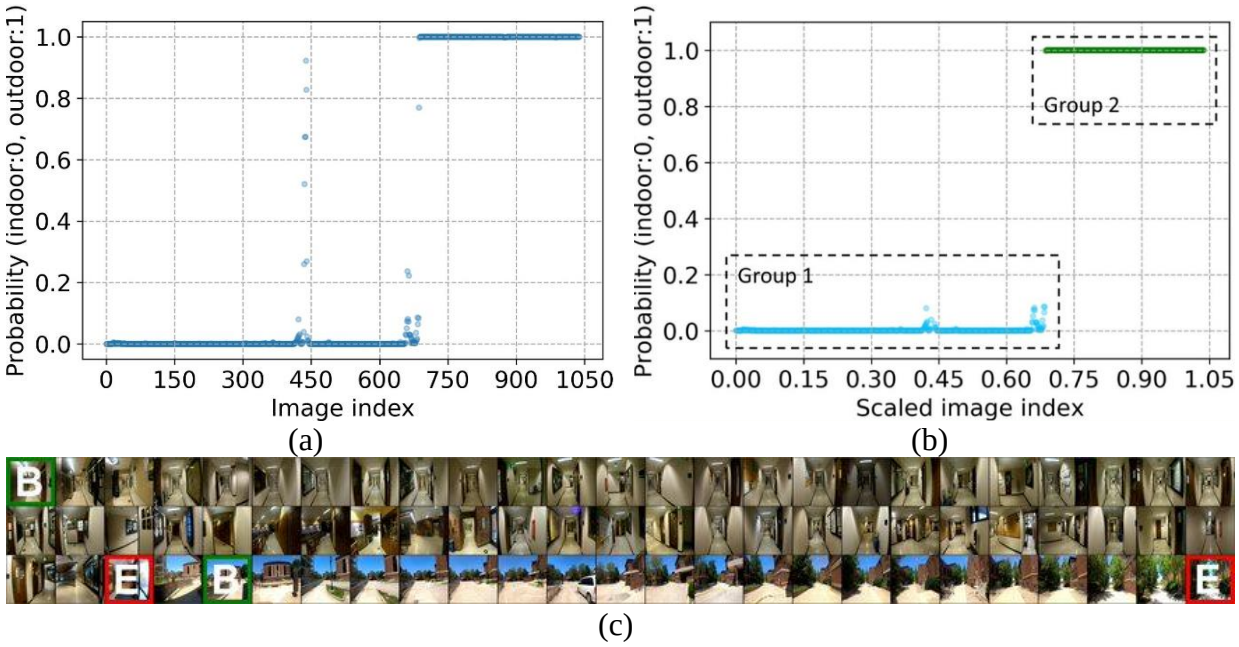


Figure 3.9. Results of indoor and outdoor separation with dataset 3.1: (a) probability, (b) final separation, (c) separation of PathImgs

Similarly, data set 3.2 starts in a passage on the second floor in the ME building on Purdue's campus, continues outside from the side door in the southeast direction of the building, then walks along the way besides Potter Center, and ends with an arrival inside the first floor of Knoy Hall; 1274 PathImgs are used. The results are shown in Figure 3.10, including the intermediate probabilities and final separation results. The data are successfully separated into three clusters. Also, the boundary PathImgs are marked as in Figure 3.10c.

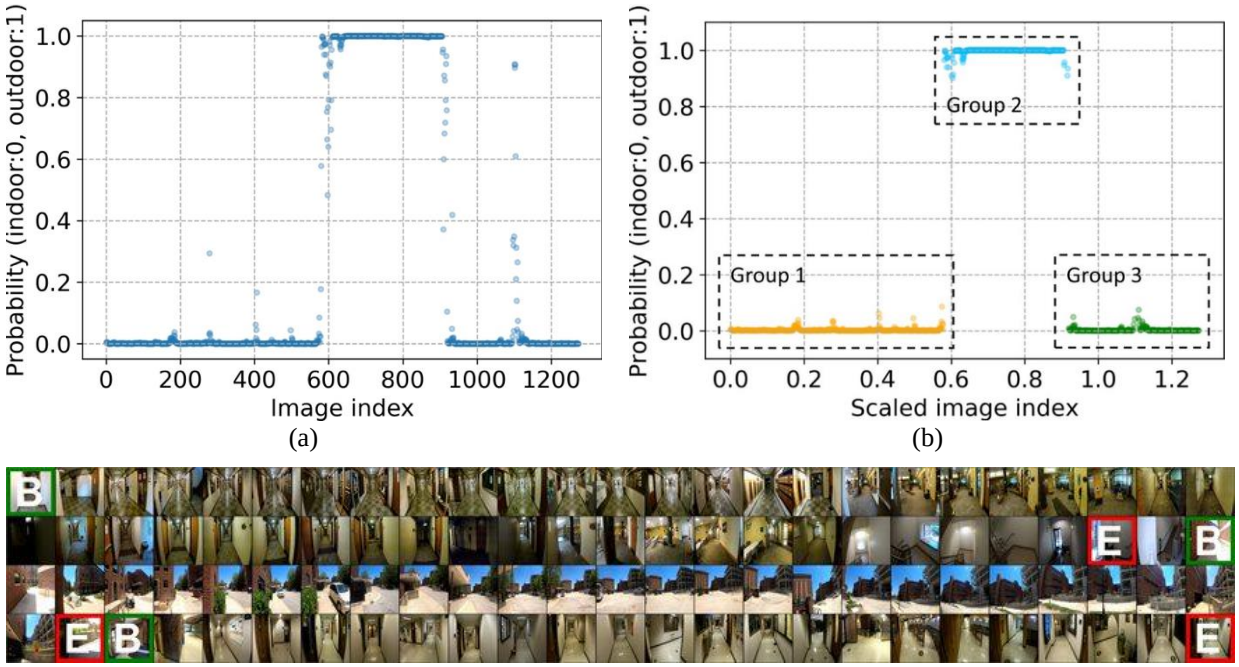
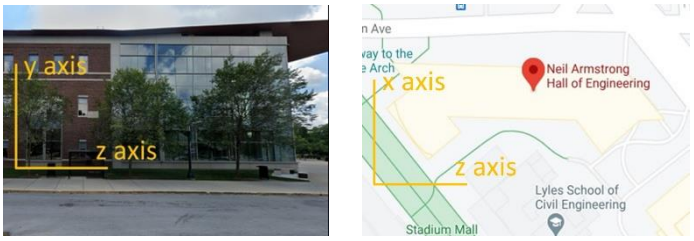


Figure 3.10. Results of indoor and outdoor separation with dataset 3.2: (a) probability, (b) final separation, (c) separation of PathImgs

3.3.1.2 Verification of Multi-Floor Separation

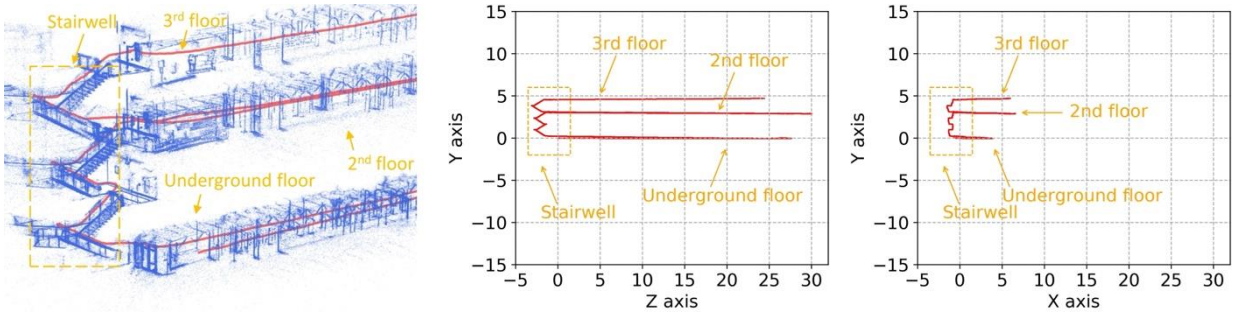
After indoor-outdoor image separation, I must process each indoor PathImg group. As in Figure 3.2, step T2-IV to step T2-V, if the PathImgs of one building contains data from multiple floors, I use the height information in the Path reconstruction to separate data collected at different floors. The multi-floor separation is tested with a PathVideo spanning three floors in Armstrong Hall on Purdue's campus. The passage begins from the underground floor, then the data collector climbs the stairwell to walk through part of the second floor, and then to the third floor. The first floor is skipped here to show that there is no need to collect the data from every floor to use this method, rather the data collector can choose to enter a given floor based on the need for data. The PathVideo is also collected with a motion camera, a GoPro HERO 8. All camera settings are the same as in the previous section.



(a) (b)

Figure 3.11. 3D coordinate system for the Path reconstruction: (a) $\boldsymbol{x-y}$ coordinate plane [34], (b) $\boldsymbol{x-z}$ coordinate plane [35]

Before demonstrating the results, I need to explain the 3D coordinate system used. The coordinate system is defined at the moment when the first PathImg is taken. As shown in Figure 3.11, the z axis is defined along the direction the data collector faces from backward to forward. The x axis similarly corresponds to the direction from the left to the right. The y axis is then perpendicular to the ground, from downward to upward.



(a) (b) (c)

Figure 3.12. 3D reconstruction of PathPcl of the data set: (a) 3D reconstruction, (b) Path reconstruction in $\boldsymbol{z-y}$ plane, and (c) Path reconstruction in $\boldsymbol{z-x}$ plane

I use VO to rebuild the 3D PathPcl using PathImgs. The 3D reconstruction is displayed in the z–y coordinate plane, as in Figure 3.12a [24]. There are 3387 points in Path and 1,336,847 points in Pcl. The red-colored lines correspond to the Path, the Path that the data collector takes when walking through the building. The blue-colored points correspond to Pcl. They are meant to capture the exposed infrastructure components. Clearly, the 3D reconstruction rebuilds all contents in the environment including the stairwell when the Path changes in the height direction or along the y axis. The Path reconstruction is plotted in the z–y coordinate plane as in Figure 3.12b, and in the x–y coordinate plane as in Figure 3.12c. Obviously, the height values along the y axis

separate the entire path into three parts, as the PathImgs belong to three floors. In Figure 3.12b,c, the unit of the x, y, and z axis is a hypothetical unit, which is determined by the VO algorithm. It is proportional to the corresponding real-world unit.

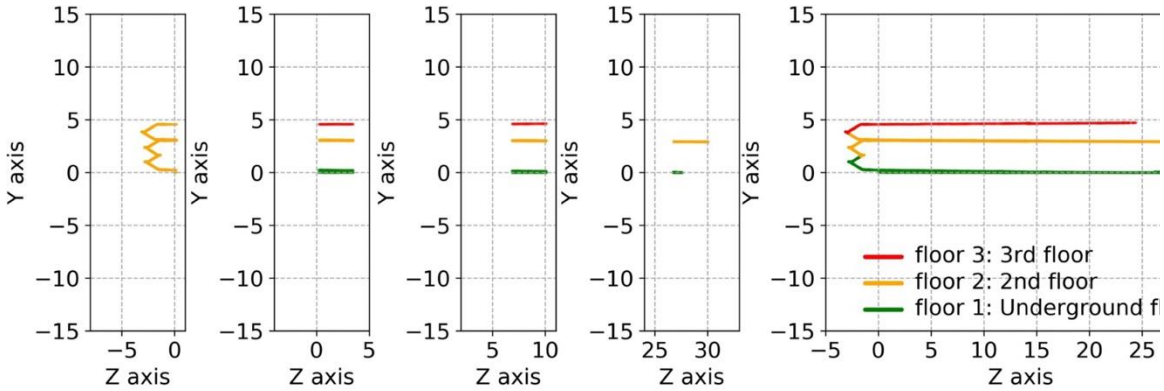


Figure 3.13. Clustering results of segments using unsupervised clustering and final separation results using supervised clustering: (a) cluster results of segment 1, (b) cluster results of segment 2, and (c) cluster results of segment 4, (d) cluster results of segment 10, (e) final separation results

It is obvious that the Path(s) of different floors are joined at the stairwell. Without any prior knowledge of where in the Path the stairwell is located, separating Path by associating it with the floors requires some assumptions. Because Path for the stairwell only exists over a limited range along the z axis, I first divide Path into a number of segments (here, the number is set to 10 by experience) along the z axis, and apply an unsupervised clustering method to each of the segments. As with the method in Section 3.3.1.1, I perform the hierarchical clustering with the single linkage algorithm [33] based on the 2D Euclidean distance between Path points. Representative results of some segments are shown in Figure 3.13. Although the first segment is clustered into one cluster corresponding to the location of the stairwell, as shown in Figure 3.13a, most other segments yield the correct number of floors, three, as shown in Figure 3.13b–d. Because segment 10 does not contain Path at floor 3, these two segments yield the number of the clusters which is 2. Thus, among all of the results given by the unsupervised clustering of each segment, those with the maximum number of clusters determine the correct number of floors. Using this number as the input for the number of clusters, I apply another supervised clustering method along the direction of y axis. Here, I adopt the K-Means clustering methods [36]. This process generates the final

results shown in Figure 3.13e. As denoted in the figure, different colors represent path points belonging to different floors. It should be mentioned that I can only determine the relative floor index, for instance, floor 1, floor 2, or floor 3, using the mean value of the coordinates cluster along y axis. Because the points of Path link to specific PathImgs, they are thus separated by referring to the separated Path. Thus, the multi-floor separation of PathImgs of one indoor group is complete. It should be pointed out that, in a multi-floor separation, for each floor, I also automatically cut a number of PathImgs (the number is set to 60 times step 200) from the beginning and the end after I apply multi-floor separation. This step is merely to remove Path in the stairwell and to avoid the possibility of including bad boundary PathImgs between floors.

3.3.1.3 Verification of Path Overlay

The Path overlay step is performed to automatically overlay the PathPcl of one floor onto the corresponding structural drawing. This step follows both the indoor-outdoor separation step, and the multi-floor separation step. PathPcl for each floor is reconstructed using VO [24], while structural drawings are saved as digital images. Prior to solving the optimization problem of Path overlay, I implement an automated step to rotate the skewed PathPcl to the nominal coordinate system of the SDI. A plane surface is fit to the reconstructed Path. Then, I find the transformation matrix by projecting the normal vector of this plane to the normal vector of the x–z plane. In this way, the skewness is corrected.

In the Path overlay step, I automate this overlay process. Hyper-parameters are tuned and set up before the validation, as discussed in Section 3.2.2.3. The only term that will vary with the specific structural drawing is the total level of the image pyramid, in Step T2-B-2 in Figure 3.3. Based on the experience, the total level should be chosen such that the top level has both a width and height that are larger than 350 pixels.

Here I use data collected from the underground floor in Armstrong Hall, which is part of the data used in Section 3.3.1.2. There are 1254 points in Path and 1,904,234 points in Pcl. In this case, the structural drawing is 8400 × 6000 pixels. Thus, the total level of the image pyramid is chosen to be 5, with the top level defined as level 4 to the origin level defined as level 0. As explained in Section 3.2.2.2, the search results are mainly determined by the search at the top level, in this case level 4. For instance, the cost function history at the top level is shown in Figure 3.14. The red-colored points are the minimum value of the cost function in each iteration during the

entire search process of this method. The orange-colored line corresponds to the global minimum value of the cost function with this method. Clearly, only one iteration or one PSO cannot guarantee reaching the global minimum. With this iterative scheme, the chance of reaching the global minimum is greatly increased. As a comparison, I apply the original PSO on the same data to search for the optimal results. The global minimum value of the cost function of PSO is plotted in blue color. PSO is also found to hit a stable minimum result. However, this is merely a local minimum and after several iterations the PSO remains at that result, while this method robustly finds the global minimum.

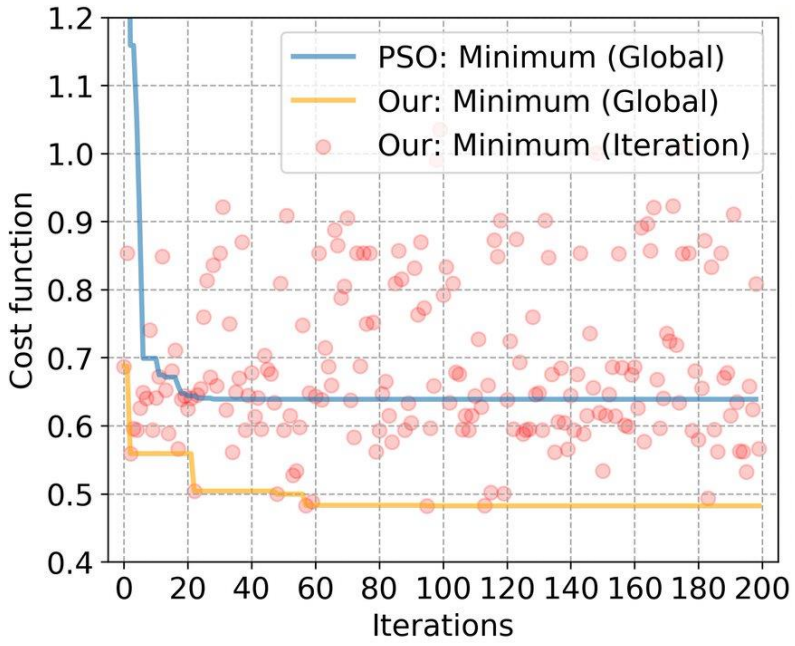
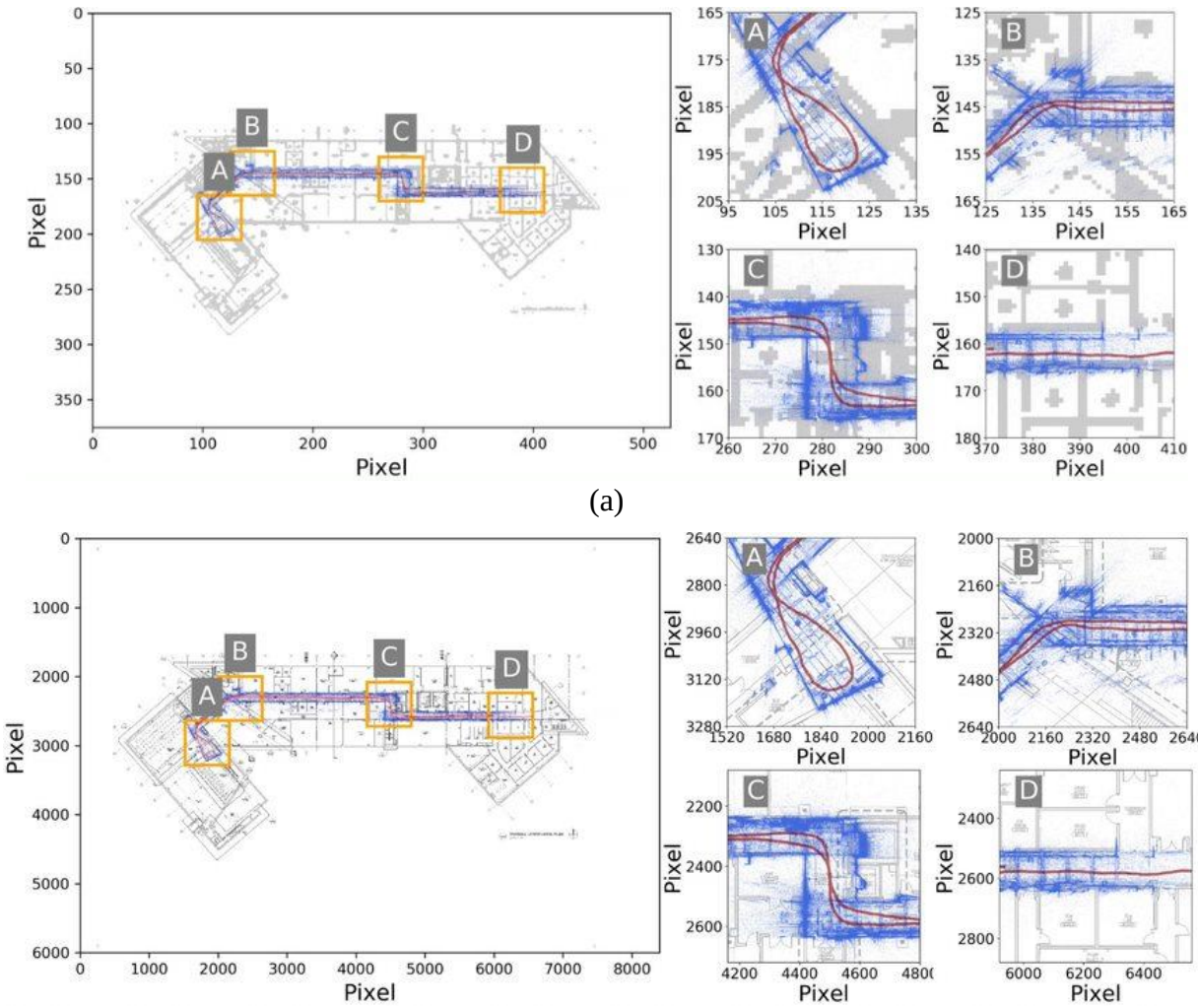


Figure 3.14. Cost function history at level 4

The results for level 4 and level 0 are shown in Figure 3.15, where both the overall view and the detailed view are shown. In the figures, the red-colored lines are the path taken by the data collector, and the blue colored points are the points in Pcl. Herein, the alignment and location of the blue points on the black lines of the structural drawing show that the automated overlay algorithm is quite successful.

Often photos are taken of paper drawings for older buildings, and I have addressed how to reassemble such photos into a drawing [37]. This stitched image can serve as the role of SDI in this task when a digital SDI is not available.



(a)

(b)

Figure 3.15. Results of automated overlay for underground floor of Armstrong Hall: (a) overlay results in level 4, (b) overlay results in level 0 (origin structural drawing)

3.4.2 Validation with Large Scale Reconnaissance Data

To assess the complete technique, I also perform an end-to-end validation. I collect continuous data from three buildings on Purdue's campus, starting from Armstrong Hall, to ME building, and ending after walking through Knoy Hall. The buildings are shown on the map in Figure 3.16, along with the walking route that the data collector takes between each building. The data collection route covers two floors in Armstrong Hall, the underground floor, and the second floor. It also includes the third floor in the ME building and the first floor in Knoy Hall. I continuously walk through all of the floors in each of these buildings to collect data, and also move

between these buildings without making any stops. In this way, the data collection aims to imitate a real reconnaissance mission.

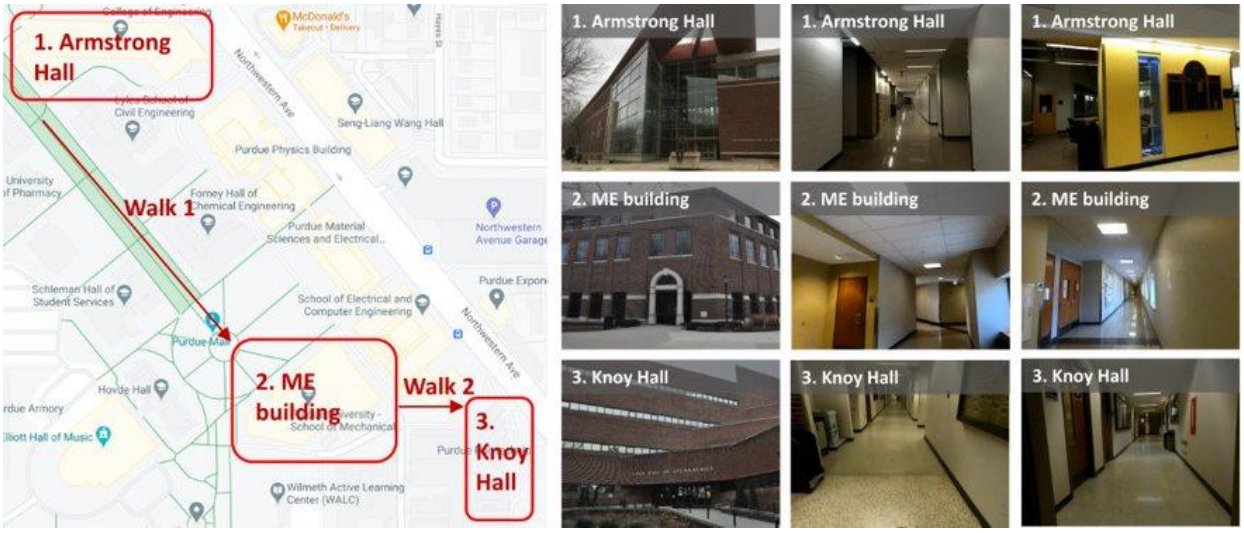


Figure 3.16. Buildings covered in the validation data [38]

In this experiment, I manually collect InspImgs using a DSLR camera (Nikon D90) and PathVideo using a motion camera (GoPro HERO 8). Before the data collection, the two cameras are set to have the same timestamp. In total, 811 InspImgs are collected along with a 53 min PathVideo at 240 fps. Each InspImg is 4288 × 2848 pixels, and each frame of the PathVideo is 1920 × 1080 pixels. The DSLR camera is set to fully automated mode for collecting InspImgs. The motion camera is set to 240 fps video mode and all other settings are set to their default values. In the experiment, two people work together to collect InspImgs and PathVideo at the same time. In practice, however, one person can perform the entire data collection by attaching the motion camera to one's body to record the PathVideo. Meanwhile, the person holds and operates the DSLR camera to collect InspImgs. InspImgs are select images targeting structural components, damage spots, and so forth that the data collector deems important to document with images, while the PathVideo is continuously recording the scenes in front of the data collector regardless of where that person directs their attention.

I use a workstation with an Intel i9-7920x CPU, 32 Gb memory, and an NVIDIA GeForce RTX 2080Ti video card to apply the technique to the collected data. The entire process is fully automated and the results of the main steps are given here. To start with, the indoor-outdoor

separation results are shown in Figure 3.17. Again, I use one PathImg from every 200, and here 3806 PathImgs are used. In Figure 3.17a,b, the probability and the final separation results are presented. The PathImgs of the PathVideo are successfully separated into five clusters. Starting from the left side to the right side of the plot in Figure 3.17b, one can see the first indoor group corresponding to Armstrong Hall, the first outdoor group corresponding to the passage from Armstrong Hall to the ME building, the second indoor group corresponding to the ME building, the second outdoor group corresponding to the passage from the ME building to Knoy Hall, and the third indoor group corresponding to Knoy Hall. Whether a group is located indoors or outdoors is determined by its mean probability value. The boundary PathImgs for each indoor or outdoor group are determined and marked in Figure 3.17c. Images with green-colored box correspond to the beginning PathImg for each group, while red-colored ones correspond to the ending PathImg.

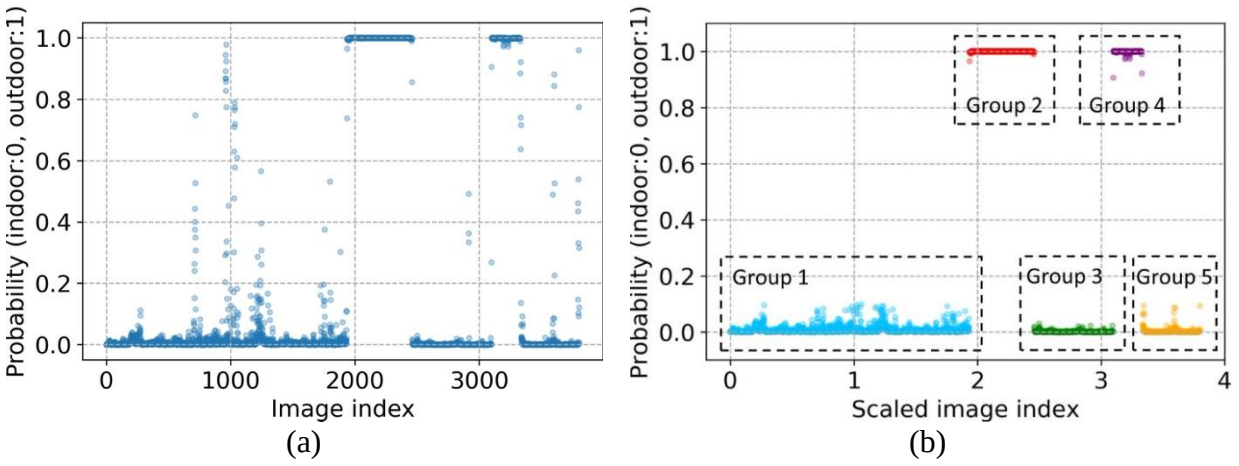
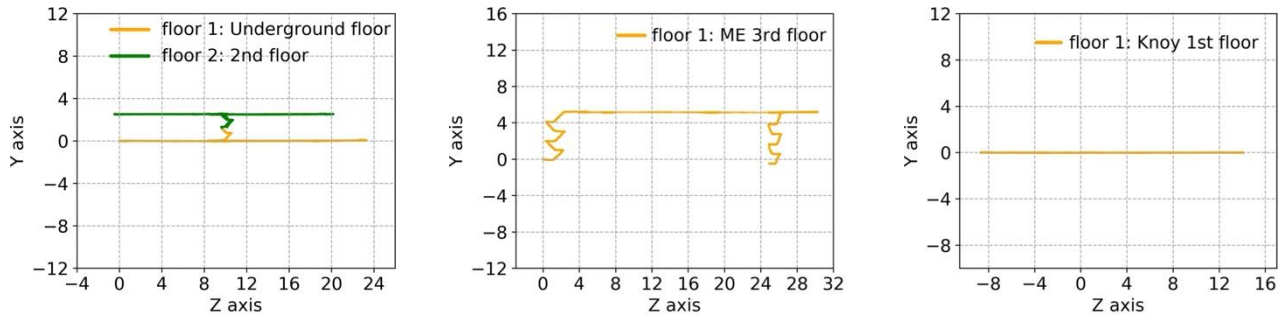


Figure 3.17. Results of indoor and outdoor separation: (a) probability, (b) final separation, (c) separation of PathImgs

After indoor–outdoor separation, I apply multi-floor separation for each indoor group. I use VO [24] to reconstruct the Path for each indoor group. There are 4135 points in Path of the first indoor group, 1785 points in Path of the second indoor group, and 1459 points in Path of the third indoor group. The multi-floor separation results are shown in Figure 3.18 in the same coordinate system as Figure 3.11, where the y axis is proportional to the height from the ground, while the x axis and z axis are determined by the orientation of the first PathImg collected. Note that I choose different stairwell as in Section 3.3.1.2 for the Armstrong Hall to justify multi-floor separation can deal with different cases. The first indoor group for Armstrong Hall is separated into two floors, while the second and the third indoor groups are identified as belonging on the first floor. By tracing back from the boundary points in each part of Path, I determine the index of the PathImgs that define the boundaries for the PathImg group for a single floor.

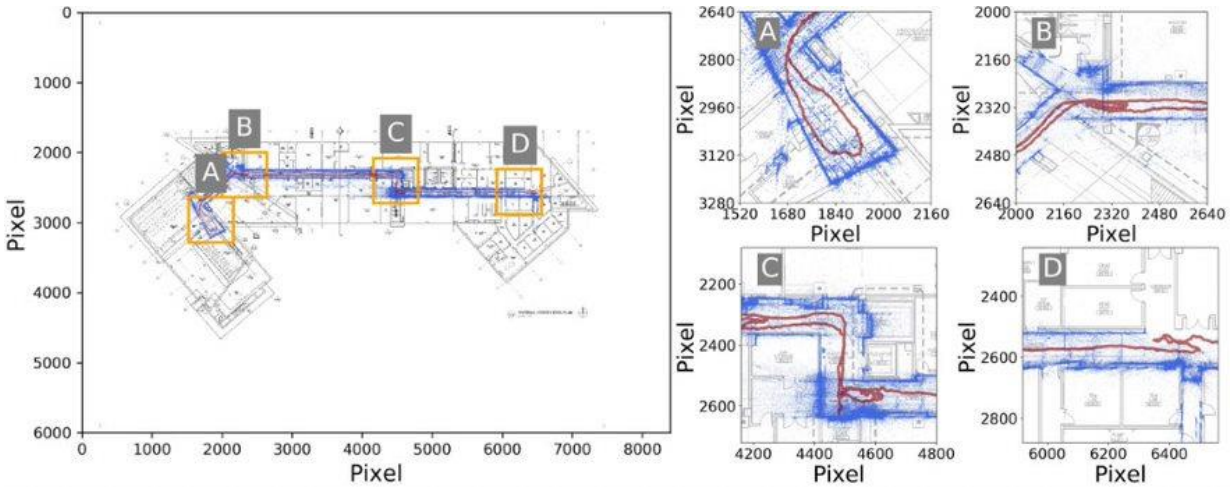


(a) (b) (c)

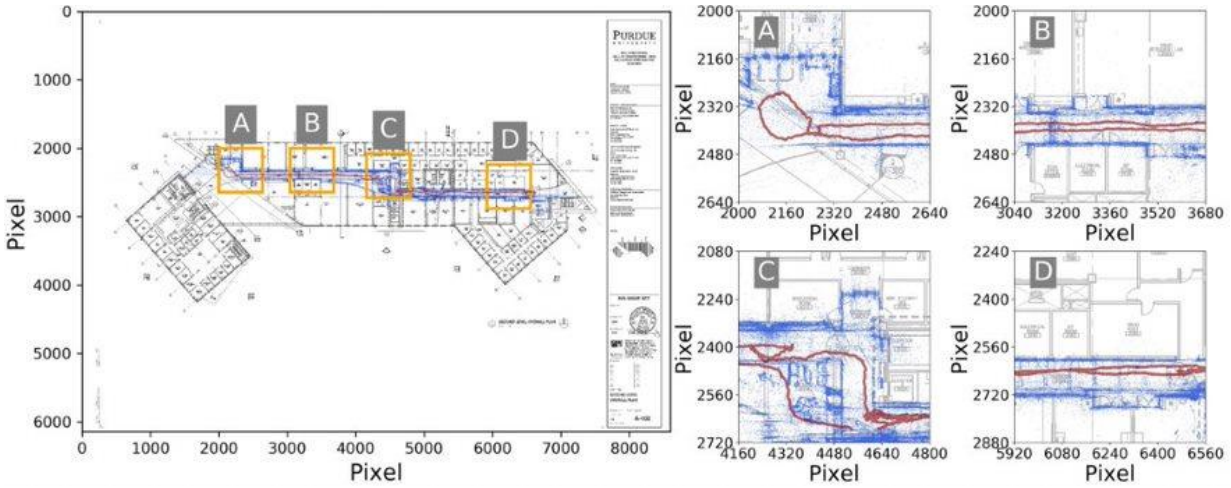
Figure 3.18. Results of multi-floor separation for each indoor group: (a) indoor group 1, (b) indoor group 2, and (c) indoor group 3

On each floor, VO [24] is used to reconstruct PathPcl using the local PImgs. PathPcl is then overlaid onto the structural drawing using the overlay algorithm. For the underground floor in Armstrong Hall, there are 1796 points in Path and 1,252,363 points in Pcl, and the structural drawing is 8400 × 6000 pixels. For the second floor in Armstrong Hall, there are 1671 points in Path and 624,747 points in Pcl, and structural drawing is 8600 × 6143 pixels. For the third floor in the ME building, there are 1785 points in Path and 858,257 points in Pcl, and the structural drawing is 656 × 570 pixels. For the first floor in Knoy Hall, there are 1459 points in Path and 522,189 points in Pcl, and the size of the structural drawing is 2533 × 1428 pixels. The overlay results for these cases are shown in Figure 3.19a–d. Again, the match between the shape formed by Pcl and the lines in the structural drawing demonstrates that the overlay is successful. Note that the PathPcl

of the ME building and Knoy building are overlaid onto floor plans. This demonstration is to illustrate that the overlay algorithm adapts to typical field scenarios other than using formal structural drawing.



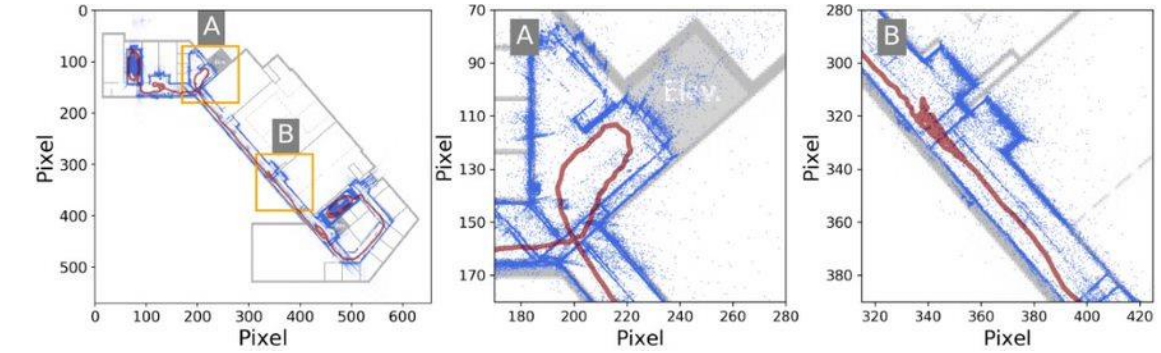
(a)



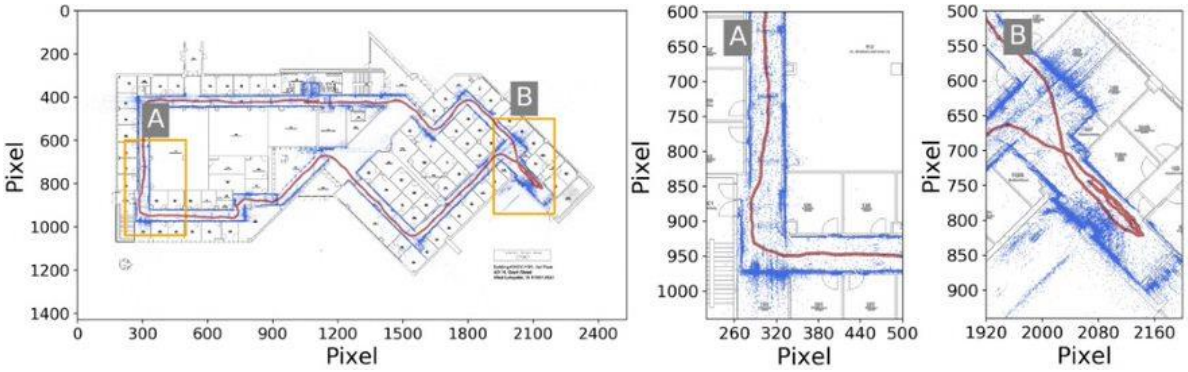
(b)

Figure 3.19. Overlay results of each floor: (a) results of underground floor in Armstrong Hall, (b) results of 2nd floor in Armstrong Hall, and (c) results of 3rd floor in the ME building, (d) results of 1st floor in Knoy Hall

Figure 3.19 continued



(c)



(d)

With these results, the locations of InspImgs on a structural drawing can be extracted and the images can be reviewed. Each InspImg links to a PathImg that has the closest timestamp to that of the InspImg, and that PathImg corresponds to a point of Path based on the image index. With those relationships computed, InspImgs can be automatically localized onto the structural drawing by tracing through the corresponding PathImgs. A representative result is shown in Figure 3.20. Using data for the second floor in Armstrong Hall, I illustrate how one InspImg and its location can be obtained and plotted on the overlaid structural drawing. The InspImg is shown in Figure 3.20a. The 3D textured model reconstruction at the selected InspImg is also shown in Figure 3.20b. This model is constructed using the InspImgs and PathImgs that are identified as being within a specific range of the selected InspImg. Here I define the range according to the timestamp, using 5 s for InspImgs and 30 s for PathImgs. This step in the 3D reconstruction is performed with commercial software, Pix4D mapper 4.4.4. In Figure 3.20c, the location of the InspImg is shown on the overlaid structural drawing as a green colored dot. In practice, a user can select any InspImgs for review, and the entire process will be performed automatically.

The total time to process the data and output the localization results consisted of PathVideo format changing (about 100 min); PathImgs undistortion (about 50 min); PathPcl reconstruction (about 35 min); indoor-outdoor separation, including classification, clustering, outputs (about 3 min); Path overlay (about 77 min); 3D reconstruction (about 50 min for one Pcl). Any steps that are not mentioned here normally require less than 1 min. In total, it takes about 4.5 h to process all of the data covering these three buildings (this is sufficient for rapid reconnaissance). Notice the video format changing step, which takes the major time is due to the special format of GoPro videos. Using motion cameras to output MP4 videos can avoid this step. An extra 50 min would be needed for generating a textured 3D reconstruction for one Pcl.

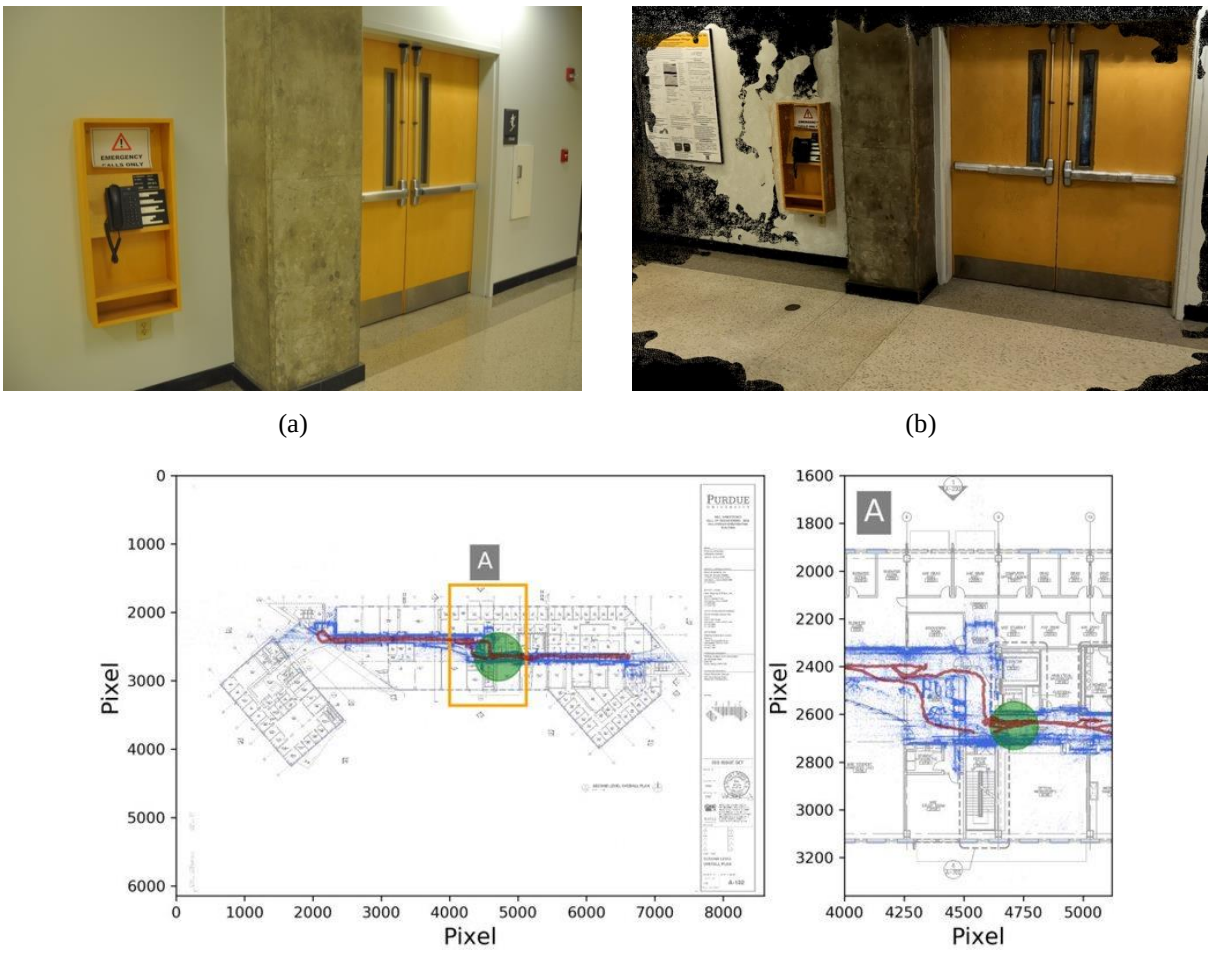


Figure 3.20. Representative results of image localization and local 3D textured model generation: (a) selected InspImg, (b) reconstructed local 3D textured model, and (c) its location on the structural drawing

3.4.3 Discussion of the Overall Results

The results illustrate that the integrated technique is successful in automating the process for general indoor environments. That is to say that the environment has to possess floors, walls, and ceilings (or most portions of them). When data are collected from such environments, this technique can rapidly and automatically process the data and provide the locations of the InspImgs to the user. With this capability, an engineer interested in reviewing the damage to a given building can easily browse through the InspImgs together with their indoor locations. This option adds value to the data collected, because the images can be automatically associated with their location in the building, which is necessary for interpretation of the damage. The added value also increases the value of these data to engineers that were not present when the data were collected, that is, their potential for re-use.

1. To reconstruct the PathPcl, the data need to be collected with sufficient lighting. If the indoor environment is not illuminated well, it is recommended that the data collector bring extra lights and use these to illuminate the scene captured by the motion camera.
2. The multi-floor separation is developed under the most common case that a building will be visited during a reconnaissance mission. This assumes that each floor is sufficiently visited, and typically the same stairwell is used to walk from floor to floor. In rare cases such as partial exploration of corridors and simultaneously using different stairwells, an alternate strategy to determine the correct number of total floors may need to be proposed.
3. For multi-floor separation, there may be ambiguity regarding how to determine whether an indoor group corresponds to a multi-floor or single floor situation. One can determine this using the height value, or simply use the number of structural drawing files input to the technique.
4. For indoor-outdoor separation, when the number of PathImgs is really large, an alternative is to break the entire set into several sets. Based on the experience, it is reasonable to use about 1000 PathImgs per group, then apply indoor-outdoor image separation, and join the individual results together to obtain the final separation results.
5. For successful Path overlay, PathPcl needs to cover at least 80% of the floor along one of the directions in the structural drawing. This requirement ensures that the automated overlay step will provide rational results. In an extreme case, one can imagine that if the inspection only takes place within a small portion of a floor (perhaps a small portion of a

hallway or just one room within a large building), the Path overlay is likely to fail to yield an acceptable result. This recommendation is used to define the search boundary for $Sinitial$ in Section 3.2.2.2.

3.5 Published manuscript

Liu, X., Dyke, S. J., Lenjani, A., Bilionis, I., Zhang, X., & Choi, J. (2022). Automated image localization to support rapid building reconnaissance in a large-scale area. *Computer-Aided Civil and Infrastructure Engineering*.

3.6 Author Contributions

Liu and Dyke generated the concept behind the work.
Liu was responsible for data collection, coding and data analysis to generate results.
Lenjani, Zhang and Choi supported data collection and method verification.
Liu wrote the paper with support from Dyke and Bilionis.
Dyke and Bilionis provided supervision.

3.7 References

[1] Kos, T., Markezic, I., & Pokrajcic, J. (2010, September). Effects of multipath reception on GPS positioning performance. In Proceedings ELMAR-2010 (pp. 399–402). IEEE.

[2] Liu, X., Dyke, S. J., Yeum, C. M., Bilionis, I., Lenjani, A., & Choi, J. (2020). Automated indoor image localization to support a postevent building assessment. Sensors, 20(6), 1610.

[3] Liu, X., Dyke, S. J., Lenjani, A., Bilionis, I., Zhang, X., & Choi, J. (2022). Automated image localization to support rapid building reconnaissance in a large-scale area. *Computer-Aided Civil and Infrastructure Engineering*.

[4] Bahl, V. & Padmanabhan, V. (2000). Enhancements to the RADAR user location and tracking system. [Technical Report MSR-TR2000-12], Microsoft Research, Redmond.

[5] Gutmann, J. S., Fong, P., Chiu, L., & Munich, M. E. (2013, April). Challenges of designing a low-cost indoor localization system using active beacons. In 2013 IEEE conference on technologies for practical robot applications (TePRA) (pp. 1–6). IEEE.

[6] Meng, W., He, Y., Deng, Z., & Li, C. (2012, April). Optimized access points deployment for WLAN indoor positioning system. In 2012 IEEE wireless communications and networking conference (WCNC) (pp. 2457–2461). IEEE.

[7] Pierlot, V., & Van Droogenbroeck, M. (2014). BeAMS: A beacon-based angle measurement sensor for mobile robot positioning. IEEE Transactions on Robotics, 30(3), 533–549.

[8] Want, R., Hopper, A., Falcao, V., & Gibbons, J. (1992). The active badge location system. ACM Transactions on Information Systems (TOIS), 10(1), 91–102.

[9] Willis, S., & Helal, S. (2004). A passive RFID information grid for location and proximity sensing for the blind user. University of Florida Technical Report, 1–20.

[10] Li, R., Yuan, Y., Zhang, W., & Yuan, Y. (2018). Unified vision-based methodology for simultaneous concrete defect detection and geolocalization. Computer-Aided Civil and Infrastructure Engineering, 33(7), 527–544.

[11] Espinace, P., Kollar, T., Soto, A., & Roy, N. (2010, May). Indoor scene recognition through object detection. In 2010 IEEE International conference on robotics and automation (pp. 1406–1413). IEEE.

[12] Gupta, S., Arbelaez, P., & Malik, J. (2013). Perceptual organization and recognition of indoor scenes from RGB-D images. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 564–571). IEEE.

[13] Quattoni, A., & Torralba, A. (2009, June). Recognizing indoor scenes. In 2009 IEEE conference on computer vision and pattern recognition (pp. 413–420). IEEE.

[14] Kaminsky, R. S., Snavely, N., Seitz, S. M., & Szeliski, R. (2009, June). Alignment of 3D point clouds to overhead images. In 2009 IEEE computer society conference on computer vision and pattern recognition workshops (pp. 63–70). IEEE.

[15] Ni, K., Armstrong-Crews, N., & Sawyer, S. (2013, May). Georegistering 3D point clouds to 2D maps with scan matching and the Hough Transform. In 2013 IEEE international conference on acoustics, speech and signal processing (pp. 1864–1868). IEEE.

[16] Zhang, X., Agam, G., & Chen, X. (2014). Alignment of 3d building models with satellite images using extended chamfer matching. In Proceedings of the IEEE conference on computer vision and pattern recognition workshops (pp. 732–739). IEEE.

[17] Holland, J. H. (1992). Genetic algorithms. Scientific American, 267(1), 66–73.

[18] Kennedy, J., & Eberhart, R. (1995, November). Particle swarm optimization. In Proceedings of ICNN'95-International Conference on Neural Networks (Vol., 4, pp. 1942–1948). IEEE.

[19] Akhand, M. A. H., Ayon, S. I., Shahriyar, S. A., Siddique, N., & Adeli, H. (2020). Discrete spider monkey optimization for travelling salesman problem. Applied Soft Computing, 86, 105887.

[20] Dorigo, M., Birattari, M., & Stutzle, T. (2006). Ant colony optimization. IEEE Computational Intelligence Magazine, 1(4), 28–39.

[21] Yang, X. S. (2020). Nature-inspired optimization algorithms. Academic Press.

[22] Hassan, R., Cohanim, B., De Weck, O., & Venter, G. (2005, April). A comparison of particle swarm optimization and the genetic algorithm. In 46th AIAA/ASME/ASCE/AHS/ASC structures, structural dynamics and materials conference (p. 1897).

[23] Tharwat, A., & Schenck, W. (2021). A conceptual and practical comparison of PSO-style optimization algorithms. Expert Systems with Applications, 167, 114430

[24] Engel, J., Koltun, V., & Cremers, D. (2017). Direct sparse odometry. IEEE Transactions on Pattern Analysis and Machine Intelligence, 40(3), 611–625.

[25] Breu, H., Gil, J., Kirkpatrick, D., & Werman, M. (1995). Linear time Euclidean distance transform algorithms. IEEE Transactions on Pattern Analysis and Machine Intelligence, 17(5), 529–533.

[26] Adelson, E. H., Anderson, C. H., Bergen, J. R., Burt, P. J., & Ogden, J. M. (1984). Pyramid methods in image processing. RCA Engineer, 29(6), 33–41.

[27] Quattoni, A., & Torralba, A. (2009, June). Recognizing indoor scenes. In 2009 IEEE conference on computer vision and pattern recognition (pp. 413–420). IEEE.

[28] Vasiljevic, I., Kolkin, N., Zhang, S., Luo, R., Wang, H., Dai, F. Z., Daniele, A. F., Mostajabi, M., Basart, S., Walter, M. R. & Shakhnarovich, G. (2019). Diode: A dense indoor and outdoor depth dataset. arXiv preprint arXiv:1908.00463.

[29] Xiao, J., Hays, J., Ehinger, K. A., Oliva, A., & Torralba, A. (2010, June). Sun database: Large-scale scene recognition from abbey to zoo. In 2010 IEEE computer society conference on computer vision and pattern recognition (pp. 3485–3492). IEEE.

[30] Yeum, C. M., Dyke, S. J., Benes, B., Hacker, T., Gaillard, M., & Liu, X. (2019). CDS&E: Enabling time-critical decision-support for disaster response and structural engineering through automated visual data analytics. DesignSafe-CI, https://doi.org/10.17603/ds2-r0tv-sv29.

[31] Simonyan, K. & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

[32] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. Advances in Neural Information Processing Systems, 25, 1097–1105.

[33] Gower, J. C., & Ross, G. J. (1969). Minimum spanning trees and single linkage cluster analysis. Journal of the Royal Statistical Society: Series C (Applied Statistics), 18(1), 54–64.

[34] Google Street View. (n.d.). Google street view of Neil Armstrong Hall of Engineering, West Lafayette, IN, US. https://goo.gl/maps/uKz726SeX99xQLJm9

[35] Google Maps. (2021b). Google maps of Neil Armstrong Hall of Engineering, West Lafayette, IN, US. https://goo.gl/maps/QEfvDMwJpyDePmH57

[36] Hartigan, J. A. (1975). Clustering algorithms (probability & mathematical statistics). John Wiley & Sons Inc.

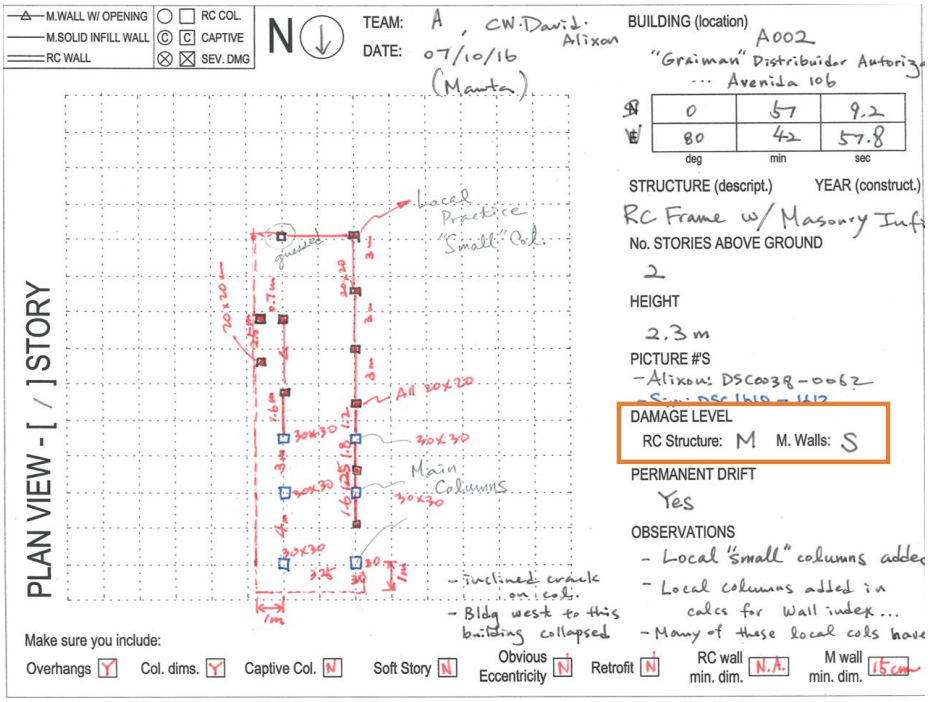
[37] Yeum, C. M., Lund, A., Dyke, S. J., & Ramirez, J. (2019). Automated recovery of structural drawing images collected from postdisaster reconnaissance. Journal of Computing in Civil Engineering, 33(1), 04018056.

[38] Google Maps. (2021a). [Google Maps of Purdue West Lafayette, West Lafayette, IN, US]. Retrieved Sep 10, 2021, from https://goo.gl/maps/R8Mu5xvuJ2ZVwZXj9

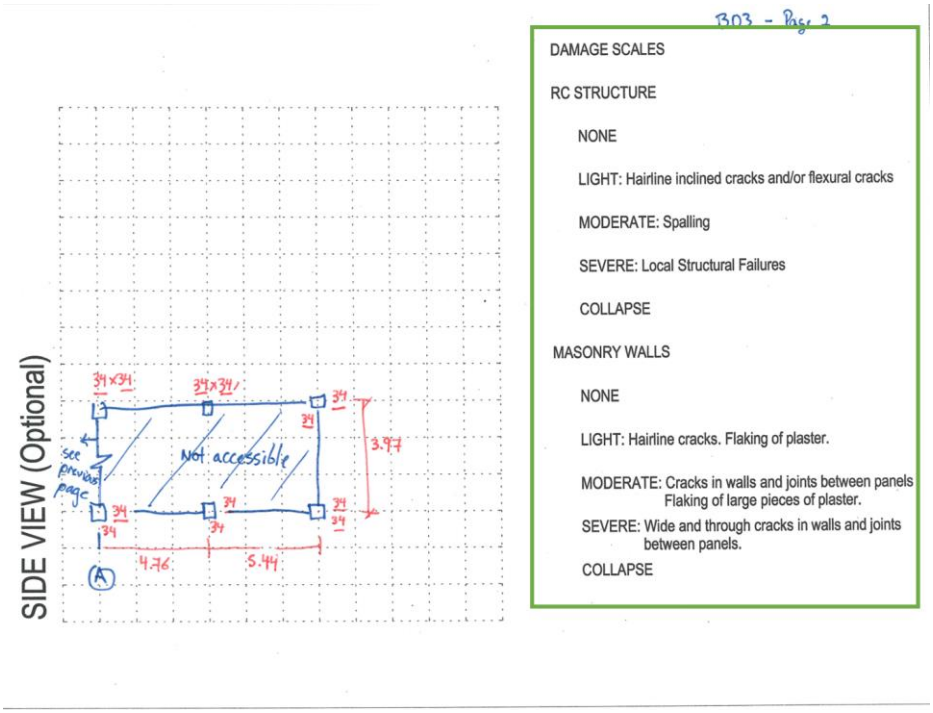
4. INFORMATION FUSION TO AUTOMATICALLY CLASSIFY POST-EVENT BUILDING DAMAGE STATE

In the field, images are collected to document evidence of the post-event condition of the structure for each building. Such image sets have been growing in size, and in recent missions the teams typically gather about 100-200 images per building. Here I will refer to this visual data collected from a given building as the set of images (SOI, hereafter). The SOI contains images with scenes focused on structural components and nonstructural components, either exhibiting damage or showing undamaged views of damaged components, and also containing various undamaged components. The SOI also contains images of other objects, such as measurements and GPS devices, and other less important objects. The SOI for a single building is certainly not comprehensive, and sometimes only representative damage to components is captured rather than collecting repetitive images.

During the data collection, an important task for reconnaissance teams is to classify the overall state of damage for buildings using general categories such as severe, moderate, etc. The classification, along with other reconnaissance information, is normally documented in a type of form which I refer as the building survey form. This damage classification task is performed separately for RC components and M components, as is evident from the information highlighted in the orange box in Figure 4.1a. These classes are assigned manually in the field following the guideline shown in the green box in Figure 4.1b. The guideline supports five states of damage each for both RC and M, including: none, light, moderate, severe, and collapse. Classifying the overall state of damage is just one example of the type of reconnaissance tasks that can be supported by automation and computer vision. Samples of images collected in the field are shown in Figure 4.1c.



(a)



(b)

Figure 4.1. Representative sample of the building survey forms used in the field: (a) the building survey form, (b) the guideline and (c) samples of images [1]

Figure 4.1 continued

(c)

Assessing the post-event condition of buildings is complex and diverse, and in some cases, unsafe for reconnaissance teams. Villalobos (2016) showed that after the 2016 Ecuador earthquake, 45% of the buildings surveyed presented severe damage, 24% presented moderate damage and 31% light damage [2]. Image data is certainly collected from the exterior of the building as well. There is an interest in using drones to perform such data collection tasks in the future, although the tremendous number of images collected would require significant time and computational power to sort and analyze as well. Efforts have also been devoted to developing methods for post-event building condition assessment using such data. Computer vision techniques have been utilized to detect various types of damage in buildings such as cracks and spalling [3-5]. Yeum et al. [6] designed clear definitions and associated image classifiers to classify images of buildings into ‘collapsed’ or ‘non-collapsed’ based on images of the building overview (overview image). Satellite images also have been used to provide such information [7-9]. However, to date the research has focused on generating in-formation from a single image. Techniques that can consider all of the images collected from a single building and produce a comprehensive output is lacking. Fusing the information from more than one images to support humans in making decisions has been developed for houses in hurricane surveys [10]. This task adopts a Bayesian-based method to fuse multiple overview images and make a decision regarding the damage level of a house. This approach provides a basis for the method developed in this paper. A barrier to that approach when dealing with more complex structures is that the computational time increases considerably when the number of images grows, for instance when dozens of images are collected from a single building. This challenge is addressed in this paper.

In this task, I develop and validate an automated technique to process the visual data to classify the damage documented in the SOI for a building. The information collected during past reconnaissance missions and published in public repositories such as DataCenterhub [11] and Design-Safe [12] is used as the basis of the technique, and also as the ground truth for its evaluation.

To establish this technique, the images are classified using convolutional neural network (CNN)-based image classifiers [13]. Two probability lists are formed, one for each category: for RC damage and M damage. Then, information fusion is performed to classify the overall RC damage state and the overall M damage state observed for the building. The main merit of this technique is that automation can assist survey teams by classifying the damage state of the building to support data organization and building-level classification. Such classification, into several broad categories based on damage state, will make useful data easier to search for in large reconnaissance data sets and serve the basis for a more targeted detailed assessment of particular structures.

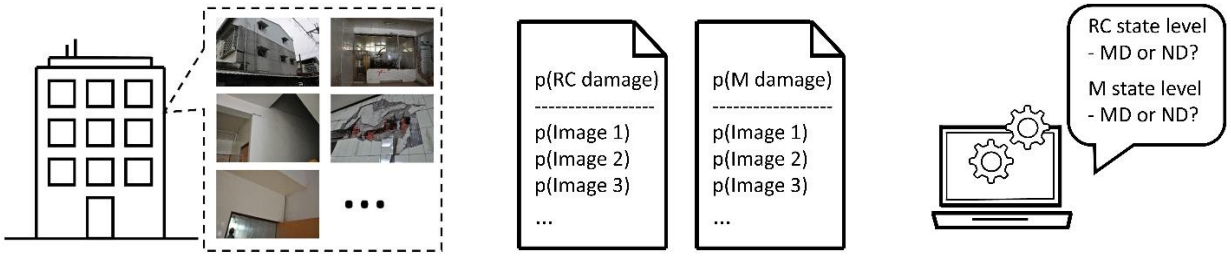
The contents of the task are organized as follows. Section 4.1 explains the methodology, including the schema designed for the image classifiers, and how to fuse the information to determine the overall damage states of the data set. Section 4.2 is the validation section, and describes the real-world dataset used for training and testing the image classifiers, and for validating the entire technique. A discussion of the results of this technique for earthquake induced structural damage, including pre-existing structural damage such as corrosion, is also included in this section. This chapter is adapted from the published work of the author [14].

4.1 Methodology

The overall workflow of the technique is shown in Figure 4.2. The input to the technique is an SOI collected from a single building during a reconnaissance mission and stored in a digital format. The output of the fully automated technique is a classification of the overall RC and M damage state present in the building based entirely on the scenes in the images collected. Thus, the technique must make these predictions of the damage state based entirely on the available SOI.

To explain the technique, I divide it into three steps. Step T3-1 is to read the reconnaissance images that comprise one SOI. Based on the observations of building survey forms and datasets from past reconnaissance mission, these images can target building components with various types of damage, or they can contain no damage at all. Images of irrelevant objects can also be included in the SOI collected for a given building; they can be automatically filtered out with image classifiers. Metadata for the SOI are not needed, although sometimes information is available, including the time and date when the images were collected, GPS coordinates, etc. The approach requires a reasonable level of quality in the images, in terms of both visual content and standards. For such purposes, the images need to have a resolution larger than 299 by 299 pixels. Beyond

that, the resolution of the images can vary in scale. The visual content of the images must be distinguishable, i.e., the damage should not take up of the entire image nor too small to be barely to be visible. Additionally, the images should not contain blur.



Step T3-1. Read reconnaissance images in an SOI

Step T3-2. Generate the damage state probability lists

Step T3-3. Use information fusion to predict the overall damage state for the building based on an SOI

Figure 4.2. Overall workflow of the approach

Step T3-2 is to generate values for the damage state in each image for populating two probability lists, one for RC damage and another for M damage. Taking RC damage as an example, each value in the list is a scalar probability between 0 and 1, and representing the probability of the corresponding image exhibiting RC damage state. This damage state probability is the raw prediction assigned by the respective image classifier to each image in the SOI. This classifier is applied after any irrelevant images are first filtered out automatically, which can be done using image classifiers. Irrelevant images are defined here as those for which the image classifier cannot generate a decision about the existence of RC damage, or for which no RC damage is present in the image. Detailed definitions of each of the classes used in the technique will be discussed in Section 4.1.1. A similar process is used for the M damage probability list. The generation of the two lists takes place in parallel, but they are entirely independent.

Then, in step T3-3, I use information fusion to determine the overall damage state for the SOI. The information fusion process is based on the naïve Bayesian method. For either RC damage or M damage classification, the process takes a probability list from step T3-2 as an input and generates a single probability value as the output. After performing information fusion, the output probability value is utilized to yield a damage state decision for the SOI corresponding to a particular building. A decision is made for the SOIs corresponding to each of the two types of

damage, RC damage and M damage, respectively. For each type of damage, the decision will be determined as one of two states, either **moderate-to-collapse damage** (MD) or **none-or-light damage** (ND), indicating the overall damage state as determined from the SOI. The definitions for MD and ND will descripted in detail in Section 4.1.1. The decisions for RC damage and M damage are derived independently throughout the entire process.

Note that although this technique is developed based on a selection of data from past reconnaissance missions, the data used here are from many locations around the world and are quite broad. Thus, I anticipate that the classifiers will be robust to variations in architecture and construction; they can be applied without any retraining. If architectural styles and construction were to vary significantly in some location or future mission from those used herein to develop the technique, the classifiers could readily be updated.

4.1.1 Schema for the Image Classifiers

To support the technique, four independent image classifiers are designed for use in step T3-2 in the overall workflow shown in Figure 4.2. All image classifiers used in this step are binary classifiers. The schema for the classifiers is shown in Figure 4.3. To make the classification result consistent and to avoid ambiguity, it is important to ensure that each classifier has a clear definition and a distinguishable boundary between positive and negative results. The definitions are provided here, and then used for labelling a training and testing dataset later. These definitions are built based on the guideline as described in the beginning of this task.

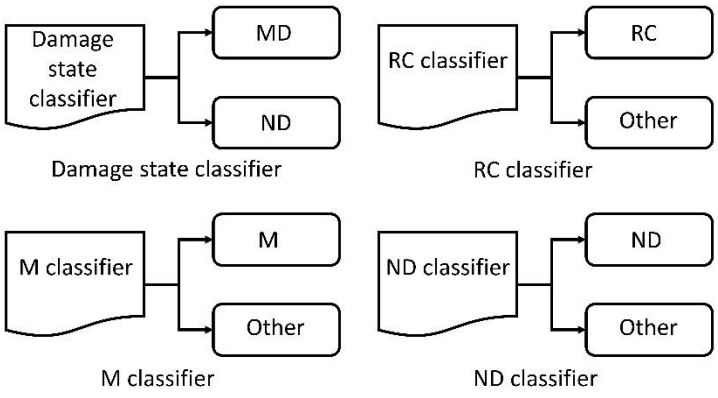


Figure 4.3. Schema designed for the image classifiers

The classifiers and the schema are as follows:

First, the **damage state classifier** classifies an image into either MD, or ND. Here, a single classifier applies to both RC damage and M damage. This approach takes into account the fact that RC damage and M damage are likely to be correlated with each other in post-event buildings, that is, if RC and M structural components both exist in that building, i.e., when the building is a reinforced concrete building that contains nonstructural masonry walls. And, more importantly, RC damage and M damage may frequently be present in a single image. Thus, this classifier is not meant to capture all types of damage but to focus on RC damage and/or M damage.

- **Moderate-to-collapse damage (MD)**: Image that contains building components having considerable damage. To be specific, the damage includes damage scales ranging from moderate, to severe, to collapse, as defined in the guideline. Moreover, it should be noted that damage in an image should be easily observed and identified, i.e., a significant part of the scene in the image should include the damage. Based on the past experience with similar classifiers, if the damage is extremely small in size as compared to the size of the image contents, it would be inappropriate to classify that image as an MD image. To quantify this relationship, I estimate that, to be classified as an MD image, the damaged region should take at least 30% of the entire area of that image without cropping.
- **None-or-light damage (ND)**: Image that contains building components having minor damage or no damage at all. This class includes damage scales from none to light, as defined in the guideline. This class is determined based entirely on the visual contents in the image, not the actual state of the building component. Thus, if the component is seriously damaged, an image capturing a healthy side of the component would also be considered as an ND image. Furthermore, an image with MD damage only in the background or damage that is hard for a human to distinguish would also be a valid ND image. To quantify this relationship, if the region with MD damage takes up no more than about a few percent (less than 5%) of the entire area of a single image without cropping, I still expect that image to be classified as a ND image.

Second, the **RC classifier** classifies an image into either RC damage, or other.

- **RC damage (RC)**: Image that contains RC damage. This class includes damage scales of moderate, severe, and collapse with respect to the RC components as defined in the

guideline. The damage should be visible on the RC structural components. The RC component should be easily recognized from the image, with visible concrete, rebar, etc. Images classified as RC should be a subset of the images classified as MD.

- **Other**: Image that is irrelevant to the condition classification of the building. Two types of images are included in this class. The first type is an image that contains no visible signs of MD damage to the building or the components. The image should not contain either RC damage or M damage, as defined above. Furthermore, the damage scale of moderate, severe and collapse are the target images that should be excluded from this class. The second type is the image that does not have the evidence to classify the building as ND, as defined in the above. An image belonging to ND can show no signs of damage, but it suggests that the building component captured in the image is in ND condition, therefore, it contributes to the decision of overall damage state to the building based on the SOI in the later process. Thus, ND images should be excluded from 'Other' class. Specifically, this class includes images about everyday objects, e.g., I have observed GPS, watches, people, vehicle, natural scenes, scenery other than infrastructure, etc. Some images with damage are also included in this class, if the scene includes irrelevant subjects such as people, papers, vehicles, etc. that represent at least 2/3 of the area of the damaged region in the image, making the damage hard to identify from the image.

Third, **M classifier** classifies images into either M damage, or other.

- **M damage (M)**: Image that contains M damage. This class includes damage scale of moderate, severe, and collapse with respect to the masonry components, as defined in the guideline. Similar to the definition of RC, the damage should be visible in the M components of the structure. The M component should be easily recognized from the scenery, with visible bricks, mortar, stones, etc. Images classified as M should be a subset of images classified as MD.
- **Other**: this class is defined in the same way as 'Other' in the RC classifier.

Fourth, the **ND classifier** classifies an image into either ND, or other.

- **ND**: this class is defined in the same way as 'ND' is defined in the damage state classifier.
- **Other**: this class is defined in the same way as 'Other' in the RC classifier.

4.1.2 Use Image Classifiers to Generate Probability Lists

In this section, the details of step T3-2 in the overall workflow, as in Figure 4.2, are explained. Using the schema for the image classifiers defined in section 4.1.1, I developed a process to generate two probability lists, one for RC damage and one for M damage. The process takes each image in the SOI as the input, and loops through each image in the SOI until it finishes.

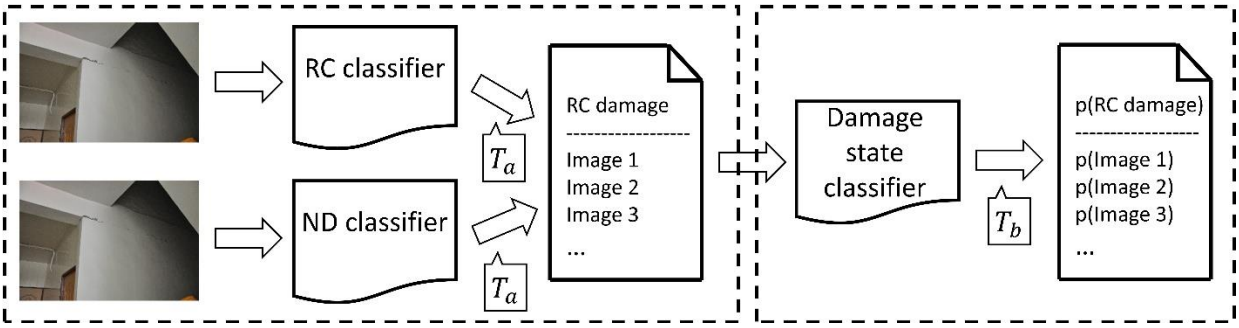
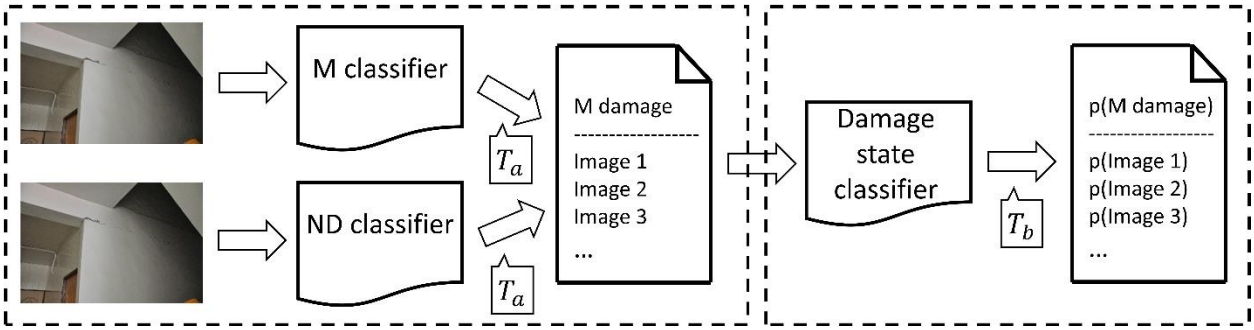


Figure 4.4. Detailed process to form the RC probability list

Figure 4.4 illustrates the process for predicting RC damage. I divide the process into two steps. Step T3-1a is to form the RC image list. First, each input image will be put through the RC classifier. The classification result determines whether or not the current input image should be included in the RC image list, i.e., if it contains RC damage with a sufficiently high probability. The decision is made by comparing the raw probability to a threshold, T_a. This approach is taken because the raw probability represents the confidence that the classifier should assign the corresponding label to that image. The closer the value is to 0 (or to 1), the more confident the classifier will be. Specifically, if the raw probability is larger than $1 - T_a$, I consider it to be valid to classify the image as RC damage, and it will be appended to the RC image list. The reason to use $1 - T_a$ is to have the threshold parameter is a region easy to visualize in the later steps. Simultaneously, I implement the ND classifier on the input image, and follow the same procedure. The image is added to the RC image list if the probability exceeds the corresponding threshold. To simplify the method, I use the same threshold parameter for each case, and it will take the same value in the process. In this way, I identify all of the images in the SOI that can contribute to derive a decision about the condition of the building components. These include images that are highly

likely to focus on RC components, and thus add evidence that the building's SOI is to be classified as a given MD state, and similarly for the ND classification.

Step T3-1b is to form the corresponding RC probability list. For each entry in the RC image list, each image will pass through the damage state classifier. This classifier assigns a probability to the image representing its likelihood of being either ND or MD. After comparing that value with a chosen threshold, T_b, the probability value will be appended to the RC probability list. It should be noted that I include images with a probability larger than $1 - T_b$ which is inclined toward MD, and images with a probability smaller than T_b which corresponds to ND. The RC probability list serves as part of the inputs to step T3-3 (from Figure 4.2) for generating the overall RC damage state for the SOI.



Step T3-2a. Form the M image list Step T3-2b. Form the M probability list

Figure 4.5. Detailed process to form the M probability list

A similar process is adopted for predicting M damage, as shown in Figure 4.5. I use the M classifier and the ND classifier to select images that should be appended to the M image list in step T3-2a, then use damage state classifier to generate the M probability list in step T3-2b. The thresholds in the process, T_a and T_b, are chosen to be the same parameter in the process for RC damage, as in Figure 4.4. They will be tuned simultaneously in Section 4.2.3. Also, I should point out that neither the RC image list and M image list, nor the RC probability list and M probability list, are mutually exclusive because an image can contain both RC damage and M damage at the same time. In that case, the image will be included in both lists, and measured by the damage state classifier in two separate processes.

4.1.3 Information Fusion

After acquiring the RC probability list and the M probability list, information fusion is used to fuse each of the probability lists into a single probability value, as in Step T3-3 in Figure 4.2. A single probability value is used to represent the damage state of either the RC components or M components of the building based on the SOI. In the following sections, I will explain the details of the information fusion algorithm. Subsequently, I will introduce a concern regarding the computational time of the algorithm. To address this concern, I integrate a sampling method to speed up the computations and the entire procedure will be explained.

4.1.3.1 Details of the Information Fusion Algorithm

I use naïve Bayesian fusion to fuse each probability list with the goal to arrive at the fused probability indicating the damage state of the building based on an SOI [10,15]. This procedure is applied separately to generate a damage state for both RC and M components. Let $x_1, \dots, x_n$ represent each image associated with the probability list, and $p_1(x_1), \dots, p_n(x_n)$ represent the damage state probability of each image. Using these values, the probability of the building based on an SOI is written as $p(D = d|\, x_1, \dots, x_n)$. And it is expressed as

$$
\begin{aligned}
&p(D = d|\, x_1, \dots, x_n) \\
&= \sum\nolimits_{d_1,\dots,d_n \in \mathcal{D}} p(D = d|D_1 = d_1, \dots, D_n = d_n, x_1, \dots, x_n) p(D_1 = d_1, \dots, D_n = d_n|x_1, \dots, x_n) \\
&= \sum\nolimits_{d_1,\dots,d_n \in \mathcal{D}} p(D = d|D_1 = d_1, \dots, D_n = d_n) p(D_1 = d_1, \dots, D_n = d_n|x_1, \dots, x_n) \\
&= \sum\nolimits_{d_1,\dots,d_n \in \mathcal{D}} p(D = d|D_1 = d_1, \dots, D_n = d_n) \prod\nolimits_{i=1}^{n} p(D_i = d_i|x_1, \dots, x_n) \\
&= \sum\nolimits_{d_1,\dots,d_n \in \mathcal{D}} p(D = d|D_1 = d_1, \dots, D_n = d_n) \prod\nolimits_{i=1}^{n} p(D_i = d_i|x_i)
\end{aligned}
\tag{4.1}
$$

To start with, $x_1, \dots, x_n$ are treated as the prior for the fused probability, since $p(x_i)$ only relates to x_i. n is the total number of the images in the probability list. Then, I define D as the random variable indicating the damage state of the building based on an SOI. And d will be a realization of the numerical value, as either 0 or 1, 0 denotes the damage state ND, and 1 denotes MD. Following that, I use the sum rule of probability to expand $p(D = d|x_1, \dots, x_n)$ to all the possible scenarios that each x has a chance being classified as ND or MD. Similar to the damage state of the building based on an SOI, D_i is the random variable for x_i, and d_i is its numerical value.

Then, $p(D_1 = d_1, \dots, D_n = d_n | x_1, \dots, x_n)$ is written as the product of the probability for each x, this is because I consider the chance for each x being classified as ND or MD are independent from each other. In the end, $p(D_i = d_i \,|x_i)$ is the probability for x_i being classified as d_i. And $\mathcal{D}$ is the set consisting of all the possible combinations of $d_i = \{0, 1\}$.

$$p(D = d | D_1 = d_1, \dots, D_n = d_n) = \begin{cases} \dfrac{\sum_{i=1}^{n} d_i}{n}, \forall\, d_i = 1, p(D_i = d_i | x_i) < 0.5 \\ \left\lceil \dfrac{\sum_{i=1}^{n} d_i}{n} \right\rceil, \exists\, d_i = 1, p(D_i = d_i | x_i) \geq 0.5 \end{cases} \tag{4.2}$$

where $\forall\, d_i = 1, p(D_i = d_i | x_i) < 0.5$ means for all $d_i = 1$, $p(D_i = d_i | x_i) < 0.5$ or all x are classified as more likely to ND over MD. In such case, I use the ratio of sum of d_i to n as the conditional probability. On the second case, $\exists\, d_i = 1, p(D_i = d_i | x_i) \geq 0.5$ means there exists $d_i = 1, p(D_i = d_i | x_i) \geq 0.5$ or at least one of x is classified as more likely to MD over ND. In such case, I use $\lceil \cdot \rceil$ of the ratio to compute the conditional probability. $\lceil \cdot \rceil$ is the mathematical ceiling of the argument. This indicates if at least one x is classified as MD or $d_i = 1$, then the conditional probability is 1.

4.1.3.2 Use of Sampling to Speed up the Process of Information Fusion

There are two characteristics this task is looking for in an information fusion algorithm. Without a doubt, the first one is 'accuracy'. The algorithm should be designed to reflect the damage state of the image set as much as possible. An evaluation of accuracy will be carried out in Section 4.2. Aside from accuracy, this task is interested in computational efficiency to get the fused result. To illustrate why this is important, an example of how to perform information fusion is provided in Table 4.1. The input, the probability list, is chosen to have four elements, with values $[0.0115, 0.1635, 0.6988, 0.1226]$. It should be noted that this hypothetical example pertains to step T3-3 in Figure 4.2, which means this is explaining what happens after all the image classification and filtering. The resulting list of probabilities is put through the information fusion algorithm. As explained in Section 4.1.3.1, the fused probability is formed by the sum rule. Thus, the algorithm must consider all possible combinations of the input list to compute the associated products and add them together. That will yield the fused probability. However, the total number of combinations is $C_{total} = \binom{N}{1} + \binom{N}{2} + \cdots + \binom{N}{N}$, where N is the number of elements in the

input list. Using the example in Table 4.1, $C_{total} = 16$, and it consumes a computation time of 0.9975 milliseconds in total. While this computation time is acceptable for a four-element list, C_{total}, will grow drastically as N increases. When N is 10, C_{total} will be 1024. When N reaches 20, C_{total} will be $1{,}048{,}576$. And when N approaches 25, C_{total} will be a whopping $33{,}554{,}432$. Since the computational time for each combination varies with the number of elements in the input list, consider that a single combination requires 0.06234 milliseconds (roughly the average time taken in the example), then, N of 25 will be about 34.86 minutes. This value is a comparatively long time to endure for this technique to assess one building. Given the fact that an SOI will easily contain tens or hundreds of images, potential large computational times will inevitably limit the value of this technique for larger image sets. This remark is based on the assumption that N will increase as the total number of images in an SOI increases.

Table 4.1. Example of the time required for the conventional information fusion algorithm

	Combinations	Product
1	[]	0.000000
2	[1]	0.000636
3	[2]	0.010678
4	[3]	0.506983
5	[4]	0.007634
6	[1, 2]	0.000248
7	[1, 3]	0.005898
8	[1, 4]	0.000178
9	[2, 3]	0.099093
10	[2, 4]	0.002984
11	[3, 4]	0.070841
12	[1, 2, 3]	0.001153
13	[1, 2, 4]	0.000052
14	[1, 3, 4]	0.000824
15	[2, 3, 4]	0.013846
16	[1, 2, 3, 4]	0.000161
Input and output		
Input: probability list, [0.0115, 0.1635, 0.6988, 0.1226]		
Output: fused probability, 0.721209		

To address this issue, I adopt a sampling method to speed up the fusion process. The basic idea is to sample a smaller number of elements from the input list, and iteratively perform the information fusion using the sampled list. Then, the process is repeated until the result converges.

The entire implementation is shown in Algorithm 4.1. To start with, I have the input probability list, A, and I define an empty list $p\_history$ for keeping track of the temporary fused probability, p_temp, which is the fused probability computed at each iteration, an empty list $e\_history$ for holding all the e, which are the errors. Before the iteration, I first check whether or not $length(A)$, the number of the elements in the input list, is larger than 5; if not, I simply compute the fused probability with A and return the fused probability as the output. The function *fuse_probability*() applies the original fusion algorithm as introduced in Section 4.1.3.1. However, if the answer is yes, the process moves to the iteration steps. I define two stop conditions, either of which will stop the iterations: one is e reaching $e\_threshold$ which is set to 0.01, since the maximum possible value of a probability is 1, $e\_threshold$ can be regarded as 1% of the maximum value, $e\_threshold$ is chosen for practical reasons so that the algorithm will reach a relatively accurate result in a reasonable iterations; and the other is reaching the $iteration\_limit$ which I set to $1{,}000$. Based on experience developed during the present study, I define the number of the elements in the sampled list, N_sample, as 5. Inside each iteration, the process moves to the sampling steps.

To fairly represent A with the sampled list, B, I adopt the proportional stratified sampling strategy [16]. This strategy is typically used when the sampling group (here, A) can be divided into several subgroups. This strategy samples from each of the subgroups independently. If I consider the procedure in step T3-2 from Figure 4.2, the RC probability list and M probability list are filtered by $threshold\_b$ to select candidates that fall into their respective classes with high confidence. This approach offers the chance to cluster A into two subgroups, one associated with probability values smaller than $threshold\_b$ which is defined in Section 4.1.2 and its value will be discussed and assigned in Section 4.2.3.2, and the other associated with probability values larger than $1 - threshold\_b$. I use proportional allocation to determine the number of elements to sample from each subgroup. Simply, the two sampled lists, denoted A_low_sample and A_up_sample, are sized to be proportional to the ratio between the size of the two subgroups, and they must add up to N_sample. Then, A_low_sample and A_up_sample form B. This sampling process is shown in lines 8 to 14 in Algorithm 4.1.

After sampling, p_temp is computed from B with the fusion algorithm. After appending p_temp to $p\_history$, I calculate e which is defined as the absolute difference between the current p_temp and the mean of $p\_history$. When either e is less than or equal to $e\_threshold$ or the

process reaches 1000 iterations, the iteration stops. When the iterations stop, if the total number of iterations is smaller than $iteration\_limit$, I take the last p_temp as the fused probability, p. Otherwise, I take the mean of $p\_history$ as p. This case applies in the rare cases in which the process does not converge early and the maximum iterations is reached. In my experience, this case has a very small chance of occurring. When it does happen, the modified process using sampling is still able to fulfill the goal of capturing the damage state of the image set, assuming that the input probability list is correctly provided. This approach works because I design the entire technique to predict a building based on an SOI as either MD or ND, rather than aiming to provide an exact probability value.

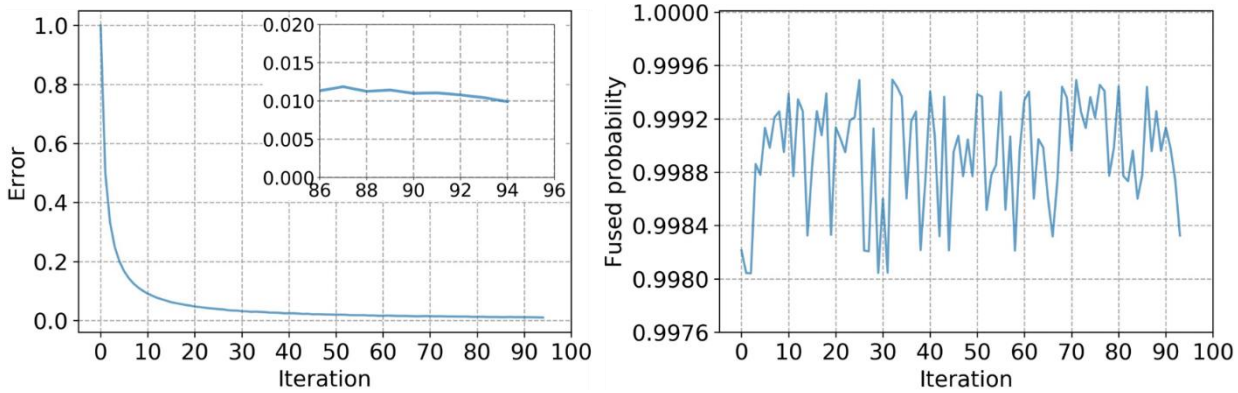
Algorithm 4.1. Implementation of the modified information fusion algorithm

Algorithm 1:

Input: probability list, A
Output: fused probability, p

```
p_history = [], e_history = [], e_threshold = 0.01, iteration_limit = 1000, N_sample = 5
if length(A) <= N_sample
        return p = fuse_probability(A)
else
        e = 1, iteration =0, e_history = [1]
        while e > e_threshold and iteration < iteration_limit
                B = []
                A_low = [element for element in A if element < threshold_b]
                A_up = [element for element in A if element > 1-threshold_b]
                Number_low = round(N_sample*length(A_low)/length(A))
                Number_up = N_sample – Number_low
                A_low_sample = random(A_low, Number_low) # randomly sampling Number_low of elements from A_low
                A_up_sample = random(A_up, Number_up) # randomly sampling Number_up of elements from A_up
                append elements of A_low_sample, A_up_sample to B
                p_temp = fuse_probability(B)
                append p_temp to p_history
                e = abs(p_temp – mean(p_history))
                append e to e_history
                iteration = iteration+1
        if iteration < iteration_limit
                return p = p_history[end]
        else
                return p = mean(p_history)
```

I examine the modified information fusion method with sample data consisting of a probability list with 27 elements, as $[0.9630, 0.9594, \dots, 0.03351]$. The process stops at the 94^{th} iteration where it reaches the stopping criterion when e meets $e\_threshold$ which is set to 0.01. The results are shown in Fig. 4.6. The error history is plotted in Figure 4.6a. Clearly, e decreases as the process proceeds. For a detailed view of the 94^{th} iteration when $e\_threshold$ is reached, the error history from iterations 86 to 96 is shown in the upper-right corner of Figure 4.6a. The history of fused probability, $p\_history$, is shown in Figure 4.6b. The final outcome of the modified algorithm is 0.9983. As a comparison, the fused probability for the original fusion algorithm is 0.9999, and the number of combinations for the original algorithm would be $134{,}217{,}728$. Meanwhile, the modified process drops this number to $94 * C_{total}(N = 5) = 3008$. The actual computation time for the modified process is 0.0728 seconds, while the original algorithm requires 3.15 hours.



(a) (b)

Figure 4.6. Results of the modified information fusion algorithm: (a) error history, (b) fused probability history

4.2 Validation of the Technique

4.2.1 Validation Dataset

I validated the technique with real world datasets from reconnaissance missions. The datasets were collected from the reconnaissance missions after several earthquakes, including Bingöl, Turkey in 2003; Haiti in 2010; Nepal in 2015; Taiwan in 2016; Ecuador in 2016; and Mexico City, Mexico in 2017 [1,17-21]. In these missions, 33,248 reconnaissance images were

collected from 800 buildings. The images cover a various of structural components with different health conditions. And they are taken from both inside and outside of buildings. Some sample images are shown in Figure 4.7.

Haiti earthquake in 2007 (129 buildings, 3,129 images)

Ecuador earthquake in 2016 (171 buildings, 6442 images)

Taiwan earthquake in 2016 (119 buildings, 9164 images)

Mexico City earthquake in 2017 (81 buildings, 3245 images)

Figure 4.7. Sample images from the reconnaissance image database [1,17-21]

During the missions, the reconnaissance teams walk through each of the buildings and manually collect each of the images in the datasets. For this task, I organized the datasets according to the building that they were collected from. I do not specifically make use of the event itself. For each building, the datasets tend to include a number of reconnaissance images and a building survey form, as shown in the sample in Figure 4.1. It should be noted that the images and the building survey forms in the original datasets do not exactly correspond to each other perfectly. Some buildings have images but lack of building survey forms, while some lack the images instead. Also, some building survey forms are empty or not legible for various reasons. As this technique aims to evaluate a building based on an SOI instead of single images, thus, I only use the data that has both an SOI and a valid building survey form for the same building. After examination of the data, there are 29,543 images and 720 buildings left for use in the following validation.

To fully test the technique, I divide the full dataset mentioned above into two parts, validation dataset 4.1 and validation dataset 4.2. The detailed assignments for the dataset are shown in Table 4.2. From validation dataset 4.1, I select some of the images to form the training set and the testing set for each of the classifiers used in this technique. The total number of images in validation dataset 1 is 26,298, and I select 5119 of them for this purpose. Next, validation dataset

4.1 will be used to tune the thresholds. In the end, validation dataset 4.2 will only be used for validating the technique. Since the process to develop the technique has not seen any of the data from validation dataset 4.2, using it for validation of the method is intended to represent an assessment of the performance of the technique on newly collected data. To form the two validation datasets, the events are randomly split as 90% (as 5) for validation dataset 1 and 10% (as 1) for validation dataset 4.2.

Table 4.2. Details of validation datasets

		Bingöl	**Ecuador**	**Haiti**	**Nepal**	**Taiwan**	**Total**
Validation dataset 4.1	RC: MD – ND	36 – 19	118 - 53	76 – 53	83 – 82	32 – 87	345 - 294
	M: MD – ND	49 – 6	133 - 38	91 - 38	119 – 46	33 – 86	425 - 214
	Total	55	171	129	165	119	639
Validation dataset 4.2		**Mexico City**					
	RC: MD – ND	33 – 48					
	M: MD – ND	46 - 35					
	Total	81					(unit: SOIs)

Also, as explained in the beginning of this task, reconnaissance teams manually evaluate the RC damage state and M damage state of each target building and document them in the building survey form. The RC damage state or M damage state is given in the building survey form as one of five possible states, based on the following set of options: {none, low, moderate, severe, collapse}. The guidelines used by the reconnaissance teams were consistent across all the different datasets used in this validation section. As discussed in Section 4.1.1, I merge the five states specified in the guidelines into two states, designated MD and ND. The number of building SOIs that include the corresponding ground truth are also listed in Table 4.2.

4.2.2 Classifier Design

To train the classifiers, I manually select images from validation dataset 4.1 and label them based on the guidelines used by the reconnaissance teams. Several sample images from each class are shown in Figure 4.8. In general, I label four categories of images, including RC damage, M

damage, ND and other. For training and testing the classifiers, RC and other form the dataset for the RC classifier, M and other form the dataset for the M classifier, ND and other form the dataset for the ND classifier, and RC and M form MD, together with ND, they form the dataset for the overall damage state classifier. Note that RC and M images are not mutually exclusive, as I discussed in Section 4.1.1. Also, RC and M damage can occur simultaneously and be captured in one image. The detailed number of images labelled and used are also listed in Figure 4.8. The number of images in each class are not uniformly selected. Instead, I select images with ambiguous visual contents, and manually label them strictly by the definitions formed in Section 4.1.1. This approach results in a more robust classifier that can correctly classify the more challenging scenes. In total, 5119 images are used here, as compared to the total number of images in validation dataset 4.1.

Image samples labelled as RC damage (843 images)

Image samples labelled as M damage (887 images)

Image samples labelled as ND (2363 images)

Image samples labelled as other (1026 images)

Figure 4.8. Labelled image samples for each class [1,17-20]

I use the same model for building all the classifiers. VGG16 is selected to be the base model of the classifiers, as its performance is one of the best in the ImageNet competition in 2014 [22]. The 5 main convolutional blocks are kept, and a new top block is attached to replace the original top block. The new top block generates a probability from 0 to 1 for each image, representing one of the binary categories of each classifier. During the training process, the pre-trained VGG16 weights, trained with ImageNet dataset, is used. The weights of the first two convolutional blocks in VGG16 are fixed, and the latter three blocks are allowed to be tuned. Together with the top block, the weights of the last three blocks are the only ones that are trained on the datasets. Since the training datasets for each classifier are not balanced, I set class weights to compensate for the imbalanced dataset in the training process.

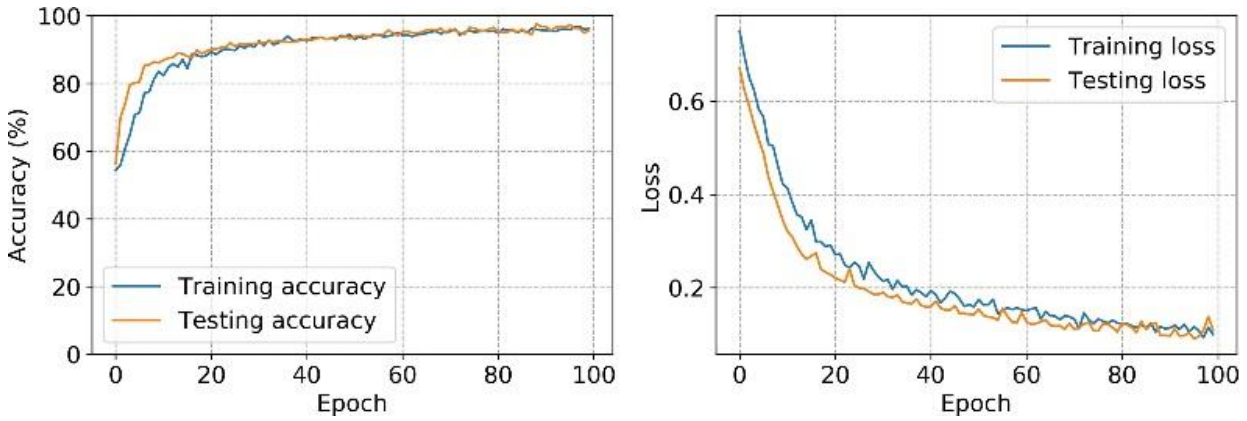


Figure 4.9. Training and testing of the RC classifier

Each classifier is trained for 100 epochs, and I use the final weights as the ones in the following test. As an example, the training and testing history for the RC classifier is shown in Figure 4.9. In the first 20 epochs, the loss drops quickly while the accuracy rises, then both histories change gradually. The training accuracy and testing accuracy for each classifier that occur in the final epoch are listed in Table 4.3. The overall performance of the classifiers is acceptable. I observe the scenes in damage state classifier are more complex as compared to the scenes for the other three classifiers, thus attribute its slightly lower accuracy to this fact.

Table 4.3. Final metrics of all the classifiers

	Trained epochs	**Final training accuracy**	**Final testing accuracy**
Damage state classifier	100	94.28%	92.88%
RC classifier	100	96.17%	95.70%
M classifier	100	98.29%	98.97%
ND classifier	100	95.86%	94.38%

4.2.3 Threshold Tuning

In this section, I tune T_a and T_b to find the values that yield the best performance of the overall technique. As mentioned in Section 4.1.2, the results of the RC classifier, M classifier and damage state classifier are filtered using the corresponding thresholds before moving to the next step in the process. Conceptually, the filters remove the portion of the results based on the

confidence with which the categories are assigned to the images. To carry out the tuning, I implement the technique with validation dataset 4.1 using a range of values of T_a and T_b. After generating the RC probability list and the M probability list, I fuse each probability list using the method explained in Section 4.1.3 to form the two overall probability values, RC fused probability (RCFP) and M fused probability (MFP). Then, I simply use a threshold of 0.5 to decide whether the building based on an SOI should be classified as ND (< 0.5) or as MD (> 0.5). The result is evaluated by the metrics of recall and precision on the entirety of validation dataset 4.1.

4.2.3.1 Metrics for Evaluating the Technique and on an Imbalanced Dataset

First, I explain the metrics used for evaluating the results [23]. Then for demonstrating how to interpret the metrics, I generate hypothetical data, and the associated results are shown with the confusion matrix in Table 4.4. The main items in the confusion matrix are denoted as follows: for the predicted damage state classification, the damage type (RC or M) followed by a "--", the prediction (MD or ND), followed by a "--", and true (or false) of the prediction, e.g., RC-MD-True means the RCFP prediction is RC MD and it is True. Similarly, RC-MD-False means the RCFP prediction is RC MD and it is False. The latter indicates that the ground truth for the classification is RC ND; for the column of total, the damage type (RC or M)-the ground truth (MD or ND)-Total; for metrics, the damage type (RC or M)-the ground truth (MD or ND)-recall, and the damage type (RC or M)-the prediction (MD or ND)-precision. After introducing the main items, the metrics are defined as follows:

$$\begin{aligned} \text{RC-MD-recall} &= \frac{\text{RC-MD-True}}{\text{RC-MD-True+RC-ND-False}} \\ \text{RC-MD-precision} &= \frac{\text{RC-MD-True}}{\text{RC-MD-True+RC-MD-False}} \end{aligned} \tag{4.3}$$

The ND and M related confusion matrix and metrics follow this same pattern. Nevertheless, when the dataset is imbalanced, there is an issue regarding the metrics shown in Table 4.4. The recall values for RC-MD and RC-ND are both pretty high, and this means the technique is quite successful in retrieving overall damage classification, both those classified as MD and ND. However, the precision values vary considerably; RC-MD-precision is 100% and RC-ND-precision is merely 1%. This outcome indicates that in the results that are predicted as ND, only 1% of them is True. The reason for this biased indication brought by the metrics is the imbalanced

dataset. Because the total number of samples with a ground truth of RC MD is 101,000 while the number with RC ND is only 10, no matter how well the technique performs, RC-ND-precision will always struggle and have a relatively low value [24].

Table 4.4. Hypothetical data and results of the demonstration

RC (Hypothetical data and results)				
Ground truth\prediction	**MD**	**ND**	**Total**	
MD	RC-MD-True: 100,000	RC-ND-False: 1000	RC-MD-Total: 101,000	RC-MD-Recall: 99%
ND	RC-MD-False: 0	RC-ND-True: 10	RC-ND-Total: 10	RC-ND-Recall: 100%
	RC-MD-Precision: 100%	RC-ND-Precision: 1%		

To compensate for this imbalance, I use a similar idea to the one adopted in Section 4.1.3.2 for accelerating the information fusion process. For the imbalanced dataset, I use a sampling method and sample from the categories with a larger number of SOIs. With these sampled results I compute the metrics, and then repeat the process until the metrics converge. The number of samples used is chosen as the number of SOIs in the smaller category, e.g., for the hypothetical data in Table 4.4, I simply sample 10 SOIs, which is the total number of RC-ND from the 101,000 SOIs as RC-MD, and use the 10 samples from RC-MD together with all of those in RC-ND to compute the metrics. I define the error to be

$$\text{Error} = \sum_{i \in metrics} |\text{i-mean(i\_history)}| \tag{4.4}$$

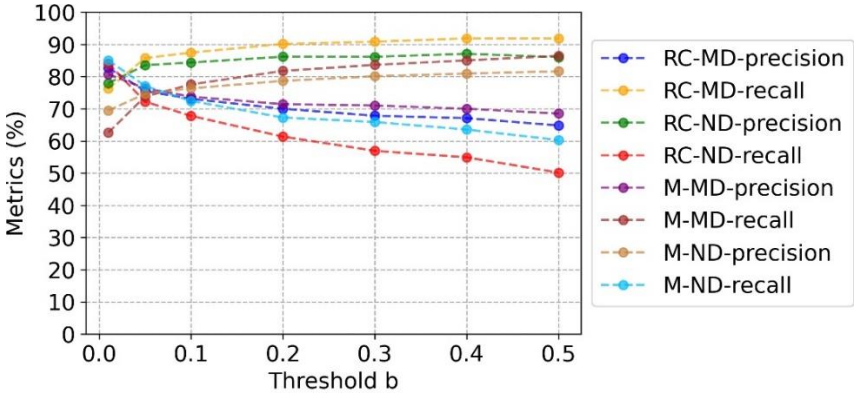
where i is one of the metrics, including: RC-MD-recall, RC-MD-precision, RC-ND-recall, RC-ND-precision, M-MD-recall, M-MD-precision, M-ND-recall, or M-ND-precision. $|\cdot|$ is the absolute value, and $i\_history$ is the history of the metrics, and *mean()* is its expected value. Similar to Algorithm 1, I define the stopping criteria of the iteration as 0.01, as the maximum possible value of each metric is 1, or 100%. When Error ≤ 0.01 or the iteration exceeds a predefined number (here, I set it to be 10,000), the iterations stop. If the number of iterations is smaller than the pre-defined limit, I use the last computed metrics; if the limit of iterations is reached, the mean of the history is used.

It is worth noting that even though similar sampling methods are utilized in both Section 4.1.3.2 and this section, the reasons for choosing to use them are fundamentally different. For the information fusion algorithm in Section 4.1.3.2, the sampling method is used to reduce the computation time that would be needed for the conventional method as much as possible. However, the goal for introducing the sampling method in this section is to overcome the issue caused by the imbalanced dataset. As the metrics are only computed one time after implementing the technique on the entire dataset, and, furthermore it will not be computed when the technique is actually implemented to classify the SOIs, the computation time is not of concern here.

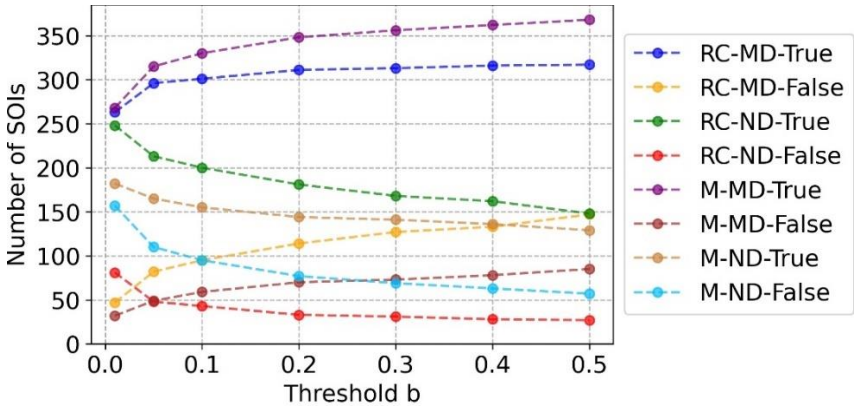
4.2.3.2 Detailed Procedure for Threshold Tuning

For tuning the thresholds, I begin by proposing candidate values of T_a as 0.01, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5 and T_b as 0.01, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5. The technique is run with each combination of these candidate values. Thus, with these candidates I perform 49 trials of the technique. For each trial, I classify all buildings in validation dataset 4.1 based on their respective SOI. The metrics are generated by the method introduced in section 4.2.3.1, and for each run 8 metrics are generated.

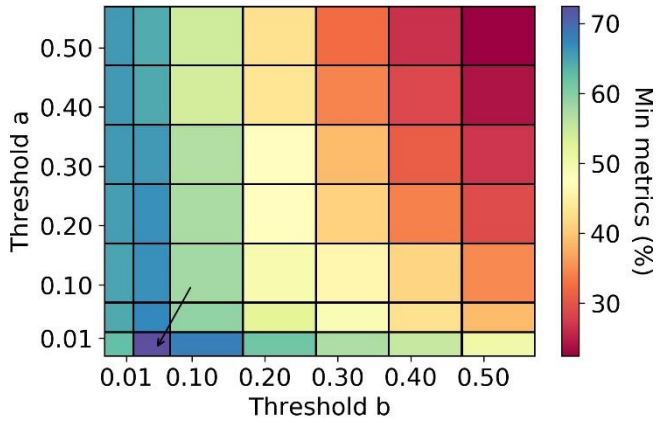
To illustrate the tuning, I plot a portion of the results for when T_a is fixed as 0.01 and T_b cycles through all its candidate values. In Figure 4.10a, I plot the resulting metrics as a function of the values of T_b. As mentioned earlier, there are 8 metrics in total. As shown in the plot, these metrics drastically change with T_b. For example, RC-ND-recall drops from 84.07% to 50.17%, as T_b changes from 0.01 to 0.5. Meanwhile, M-MD-recall increases from 62.62% to 86.45%. To understand the reason behind why some metrics are larger when T_b is large, while other metrics have the opposite behavior, I plot the number of SOIs being predicted in Figure 4.10b. As shown in the plot, all the values related to MD are increasing as T_b increases, and all the terms related to ND are decreasing. This trend occurs because when T_b is getting larger, or $1 - T_b$ is getting smaller, the filters allowing images to be classified as MD and ND are getting less strict, allowing more images to be passed to the next stage of the process as MD and ND images. Because the MD images tend to dominate in the information fusion process to classify the damage state of the SOI as MD, this outcome results in an increasing number of SOIs being classified as MD, or equivalently, a reduced number of SOIs that are classified as ND. The consequence of this behavior is the significant changes in the values of the metrics.



(a)



(b)



(c)

Figure 4.10. Thresholds tuning results: (a) results of metrics, (b) results of number of SOIs, (c) overall results

To select an optimal combination of thresholds, I simply use the minimum metrics in each case as the indicator. For instance, I use 64.95% to identify the case in which T_a is 0.1 and T_b is 0.01. From all the indicators, I select the highest, which represents the thresholds that yield the approach with the best performance. The results are shown in Figure 4.10c. In this figure, I show the minimum metrics for each combination of different values. The most appropriate one is selected to be T_a as 0.01 and T_b as 0.05 corresponding to an indicator of 72.45%, which is pointed out by the arrow in Figure 4.10c.

4.2.4 Validation Results

4.2.4.1 Validation Results on an SOI Example

In this section, I demonstrate the technique using an SOI. Several sample images from the SOI are shown in Figure 4.11 [20]. The SOI is from the Taiwan dataset, and contains 129 images. First, I walk through the workflow to demonstrate how the RC list is updated for one image. Updating the M list will be similar. The details relating to step 1a in Figure 4.4 are provided in Algorithm 4.2. The input is an image "$\boldsymbol{m}$" and the output is the updated RC list, RC_list. To begin, I obtain p_RC by applying the RC classifier and p_ND by applying the ND classifier on $\boldsymbol{m}$, respectively. Then, the following decision is made: p_RC or p_ND is larger than $1 - threshold\_a$, the RC list is updated by appending the image $\boldsymbol{m}$ to the RC list; otherwise, the RC list stays the same. This terminates the process of step T3-1a in Figure 4.4 for $\boldsymbol{m}$.

By going through this process, the algorithm avoids the extreme case of having an image classified as both RC and ND at the same time (a high probability from both the RC classifier and the ND classifier is an indication of misclassification). Take the forth sample image in Figure 4.11 as $p\_RC = 0.8831$, $p\_ND = 0.9778$, $1 - threshold\_a = 0.99$, this means $p\_RC < 1 - threshold\_a$ and $p\_ND < 1 - threshold\_a$, thus, this image will not be put in, thus, this image will not be appended to the RC list. For the second sample image in Figure 4.11, as $p\_ND > 1 - threshold\_a$, then, the image will be appended to the RC list.

Algorithm 4.2. Updating of the RC list with one image

Algorithm 2:

Input: m # image
Output: RC_list # the updated RC list

```
p_RC = RC_classifier(m)
p_ND = ND_classifier(m)
if p_RC > 1-threshold_a or p_ND > 1-threshold_a
        append m to RC_list
```

For the SOI example, the ground truth is MD for RC and MD for M. The resulting probabilities are 0.9999 for RC and 0.9999 for M. Thus, the prediction is MD for both RC and M, which agrees with the ground truth. Notice that I design approach, with a separate damage state classifier and category classifier, increases the robustness of the method to correctly predict the damage state for each image. Evidence of this robustness is found here with the forth sample image where the RC classifier assigns 0.8331 to the image indicating the image can be classified as RC-MD while the damage state classifier assigns 0.1365 indicating low damage state as the true condition of the image. The time required to generate this decision is 9.27 seconds, including the time for both image classification and information fusion.

Figure 4.11. Sample images from a Taiwan mission SOI including testing results [20]

4.2.4.2 Validation Results on the Validation Datasets

As demonstrated in Section 4.2.3.2, the technique achieves good performance using the pre-determined thresholds with all metrics being above 72%. The detailed results corresponding to validation dataset 4.1 are shown in Table 4.5. I provide the results as a confusion matrix grouped by the buildings surveyed during each event, and also provide the results over all events. For this task and similar implementations of classification methods, recall plays a more important role than precision. In particular, in this application it is critical to successfully identify as many buildings

as possible in each class, without neglecting classes that happen to contain a smaller number of buildings [6]. Thus, I only calculate and show the recall for each category. Recall values are calculated directly, since the imbalance in the dataset only significantly affects the precision values, as shown in Section 4.2.3.1.

In general, the performance is good. The results do vary somewhat with the specific event. In most cases, the performance is above or close to the overall metrics, for instance, the Bingöl, Ecuador, and Haiti datasets. However, in a couple of cases the performance is noticeably lower, including the M-MD for the Bingöl dataset, and M-MD of the Taiwan dataset, etc. I believe this outcome is mainly because the misclassification of images occurs more frequently in certain datasets. A possible solution is to collect more images containing a variety of damage conditions and architectural styles to add to the overall dataset. The variety of the training dataset is generally a strong indicator of the robustness of the classifiers trained. Also, adding more SOIs to the datasets will also reduce the likelihood of outliers in the metrics. For instance, the M-ND-recall of the Bingöl dataset is 100%. Here the M-ND-Total is only 6, and thus the high recall value does not necessarily reflect the technique. It is reasonable to expect that datasets containing more SOIs in M-ND, the recall will drop to a level closer to the overall performance.

Table 4.5. Results for validation dataset 4.1

Ground truth\prediction			MD	ND	Total	Recall
Bingöl	RC	MD	RC-MD-True: 27	RC-ND-False: 9	RC-MD-Total: 36	75.00%
		ND	RC-MD-False: 3	RC-ND-True: 16	RC-ND-Total: 19	84.21%
	M	MD	M-MD-True: 30	M-ND-False: 19	M-MD-Total: 49	61.22%
		ND	M-MD-False: 0	M-ND-True: 6	M-ND-Total: 6	100%
Ecuador	RC	MD	RC-MD-True: 102	RC-ND-False: 16	RC-MD-Total: 118	86.44%
		ND	RC-MD-False: 17	RC-ND-True: 36	RC-ND-Total: 53	67.92%
	M	MD	M-MD-True: 107	M-ND-False: 26	M-MD-Total: 133	80.45%
		ND	M-MD-False: 13	M-ND-True: 25	M-ND-Total: 38	65.79%
Haiti	RC	MD	RC-MD-True: 69	RC-ND-False: 7	RC-MD-Total: 76	90.79%
		ND	RC-MD-False: 18	RC-ND-True: 35	RC-ND-Total: 53	66.04%
	M	MD	M-MD-True: 72	M-ND-False: 19	M-MD-Total: 91	79.12%
		ND	M-MD-False: 9	M-ND-True: 29	M-ND-Total: 38	76.32%
Nepal	RC	MD	RC-MD-True: 73	RC-ND-False: 10	RC-MD-Total: 83	87.95%
		ND	RC-MD-False: 25	RC-ND-True: 57	RC-ND-Total: 82	69.51%
	M	MD	M-MD-True: 86	M-ND-False: 33	M-MD-Total: 119	72.27%
		ND	M-MD-False: 12	M-ND-True: 34	M-ND-Total: 46	73.91%
Taiwan	RC	MD	RC-MD-True: 26	RC-ND-False: 6	RC-MD-Total: 32	81.25%
		ND	RC-MD-False: 18	RC-ND-True: 69	RC-ND-Total: 87	79.31%
	M	MD	M-MD-True: 20	M-ND-False: 13	M-MD-Total: 33	60.61%
		ND	M-MD-False: 15	M-ND-True: 71	M-ND-Total: 86	82.56%
Total	RC	MD	RC-MD-True: 297	RC-ND-False: 48	RC-MD-Total: 345	86.09%
		ND	RC-MD-False: 81	RC-ND-True: 213	RC-ND-Total: 294	72.45%
	M	MD	M-MD-True: 315	M-ND-False: 110	M-MD-Total: 425	74.12%
		ND	M-MD-False: 49	M-ND-True: 165	M-ND-Total: 214	77.10%

The results for validation dataset 4.2 are shown in Table 4.6. Here it is clear that the approach also achieves good performance, especially considering the technique has not seen any images in validation dataset 4.2 before this test. Note that M-ND-recall is higher here than in the results for validation dataset 4.1. The reason for this outcome is possibly the limited number of SOIs. Increasing the number of SOIs in validation dataset 4.2 can lead to a more representative result.

Table 4.6. Results for validation dataset 4.2

Ground truth\prediction		MD	ND	Total	Recall
Mexico City	RC	MD	RC-MD-True: 26	RC-ND-False: 7	RC-MD-Total: 33	78.79%
		ND	RC-MD-False: 16	RC-ND-True: 32	RC-ND-Total: 48	66.67%
	M	MD	M-MD-True: 30	M-ND-False: 16	M-MD-Total: 46	65.22%
		ND	M-MD-False: 6	M-ND-True: 29	M-ND-Total: 35	82.86%

4.2.4.3 Influence of Corrosion and Other Types of Nonstructural Damage

As I have mentioned before, the data collection procedures do play a major role in the success of this technique. For instance, note that some of the damage visible in the images collected during the reconnaissance missions already existed prior to the seismic event. Additionally, some of the damage to concrete components was to nonstructural components. The presence of these images does bias the performance of the technique and can yield false predictions. To explore these as possible reasons for false predictions, I consider the influence of these images on the overall results. I manually remove two types of images, those with: pre-existing damage, which is evident by the level of corrosion visible, and nonstructural damage, for instance to components such as balconies or parapets.

During a reconnaissance mission, such evidence of distress in the building does not participate in the decision process because the human engineer is able to disregard this information. However, the computer is not yet able to distinguish between such cases. The design of new classifiers to filter out such data would be a viable option, however, I first must understand the role these images play in the overall success of the technique. I noticed that these situations are especially evident in the Ecuador dataset [1]. Thus, to examine the influence of these images, I manually remove such images (those with corrosion, indicating pre-existing damage; and with major nonstructural damage) from the Ecuador dataset. Then I re-run the technique on the reduced dataset and compare the results.

Several sample images that were removed because they contain corrosion are shown in Figure 4.12. In total, 16 images from 5 SOIs are removed to examine their influence on the technique. As shown in the figure, they would be classified as MD images with varying probabilities. However, when the SOIs include these images, the predictions are likely to be MD, which does not match the ground truth and thus will reduce the associated metrics. The results of the Ecuador dataset without these images are shown in Table 4.7. It is obvious that the performance

in RC-ND and M-ND improves, while RC-MD stay the same and a decrease happens in M-MD. One additional SOI is falsely evaluated as compared with the original predictions shown in Table 4.5. It is likely, with the tuned thresholds, that the removed images contribute to the MD prediction in this particular SOI. The improvement in the metrics agrees with the number of SOIs being altered. Because images with pre-existing damage are in 4 SOIs, RC-ND-True and M-ND-True increase by 2 and 2, respectively. Removing these images from the SOI, or not collecting them in the first place, would improve the results of the technique. This observation will be important for improving the data collection procedures.

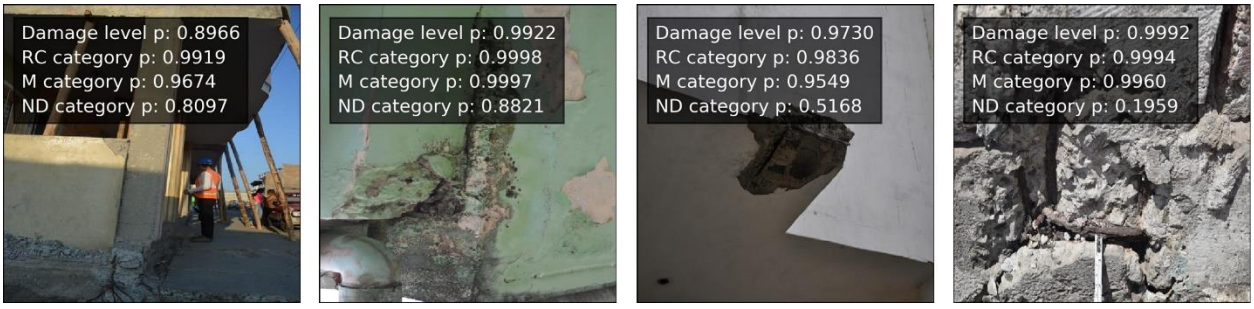


Figure 4.12. Sample images with corrosion as evidence of pre-existing damage from the Ecuador dataset [1]

Table 4.7. Results for the Ecuador dataset without corrosion images

Ground truth\prediction			**MD**	**ND**	**Total**	**Recall**	**Original recall**
Ecuador w/ corrosion	RC	MD	RC-MD-True: 102	RC-ND-False: 16	RC-MD-Total: 118	86.44%	86.44%
		ND	RC-MD-False: 15	RC-ND-True: 38	RC-ND-Total: 53	71.70%	67.92%
	M	MD	M-MD-True: 106	M-ND-False: 27	M-MD-Total: 133	79.70%	80.45%
		ND	M-MD-False: 11	M-ND-True: 27	M-ND-Total: 38	71.05%	65.79%

A similar situation is considered for images with purely nonstructural damage. Several sample images of this case are shown in Figure 4.13. In total, 49 images from 10 SOIs are removed and the predictions are repeated. The results for the Ecuador dataset without these images are shown in Table 4.8. Two categories see improved metrics, raising the number of true predictions,

while RC-MD stay the same and M-MD-True decrease by 1 likely due to the same reason in the corrosion case. Based on the sample here, it is clear that the data collection process does bias the results of the technique. These images, containing corroded components with pre-existing damage and damage to nonstructural components, contribute to the number of false predictions made by the technique. This sample case motivates the need for either new classifiers that can automatically filter out these images, or guidelines that discourage teams in the field from taking such images. The performance of such techniques will be improved with awareness about the overall process.

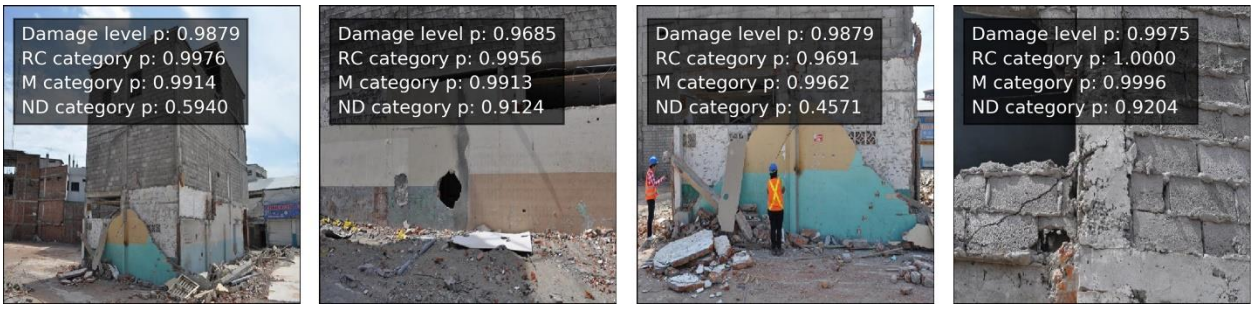


Figure 4.13. Sample images with nonstructural damage from the Ecuador dataset [1]

Table 4.8. Results for the Ecuador dataset without images of nonstructural damage

Ground truth\prediction		**MD**	**ND**	**Total**	**Recall**	**Original recall**
Ecuador w/ nonstructural damage	RC	MD	RC-MD-True: 102	RC-ND-False: 16	RC-MD-Total: 118	86.44 %	86.44%
		ND	RC-MD-False: 11	RC-ND-True: 42	RC-ND-Total: 53	79.25 %	67.92%
	M	MD	M-MD-True: 106	M-ND-False: 27	M-MD-Total: 133	79.70 %	80.45%
		ND	M-MD-False: 9	M-ND-True: 29	M-ND-Total: 38	76.32 %	65.79%

4.3 Published manuscript

Liu, X., Iturburu, L., Dyke, S. J., Lenjani, A., Ramirez, J., & Zhang, X. (2022). Information fusion to automatically classify post-event building damage state. *Engineering Structures*, *253*, 113765.

4.4 Author Contributions

Liu, Iturburu, Dyke and Ramirez generated the concept behind the work.

Liu was responsible for data collection, coding and data analysis to generate results.

Iturburu, Zhang and Lenjani supported data collection and method verification.

Liu wrote the text with support from Iturburu, Dyke and Ramirez.

Dyke and Ramirez provided supervision.

4.5 References

[1] Chungwook Sim, Enrique Villalobos, Jhon Paul Smith, Pedro Rojas, Aishwarya Y Puranam, Lucas Laughery, Santiago Pujol (2016), "2016 Ecuador Earthquake," https://datacenterhub.org/deedsdv/publications/view/535.

[2] Villalobos E, Sim C, Smith-Pardo JP, Rojas P, Pujol S, Kreger ME. The 16 April 2016 Ecuador Earthquake Damage Assessment Survey. Earthquake Spectra. 2018;34(3):1201-1217. doi:10.1193/060217EQS106M

[3] Jahanshahi, M. R., Masri, S. F., Padgett, C. W., & Sukhatme, G. S. (2013). An innovative methodology for detection and quantification of cracks through incorporation of depth perception. *Machine vision and applications*, *24*(2), 227-241.

[4] Kim, H., Ahn, E., Shin, M., & Sim, S. H. (2019). Crack and noncrack classification from concrete surface images using machine learning. *Structural Health Monitoring*, *18*(3), 725-738.

[5] Hoskere, V., Narazaki, Y., Hoang, T., & Spencer Jr, B. (2018). Vision-based structural inspection using multiscale deep convolutional neural networks. *arXiv preprint arXiv:1805.01055*.

[6] Yeum, C. M., Dyke, S. J., Benes, B., Hacker, T., Ramirez, J., Lund, A., & Pujol, S. (2019). Postevent reconnaissance image documentation using automated classification. *Journal of Performance of Constructed Facilities*, *33*(1), 04018103.

[7] Xu, J. Z., Lu, W., Li, Z., Khaitan, P., & Zaytseva, V. (2019). Building damage detection in satellite imagery using convolutional neural networks. *arXiv preprint arXiv:1910.06444*.

[8] Gueguen, L., & Hamid, R. (2015). Large-scale damage detection using satellite imagery. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1321-1328).

[9] Gupta, R., Hosfelt, R., Sajeev, S., Patel, N., Goodman, B., Doshi, J., ... & Gaston, M. (2019). xbd: A dataset for assessing building damage from satellite imagery. *arXiv preprint arXiv:1911.09296*.

[10] Lenjani, A., Dyke, S. J., Bilionis, I., Yeum, C. M., Kamiya, K., Choi, J., ... & Chowdhury, A. G. (2020). Towards fully automated post-event data collection and analysis: Pre-event and post-event information fusion. *Engineering Structures*, *208*, 109884.

[11] Catlin, A. C., HewaNadungodage, C., Pujol, S., Laughery, L., Sim, C., Puranam, A., & Bejarano, A. (2018). A cyberplatform for sharing scientific research data at DataCenterHub. *Computing in Science & Engineering*, *20*(3), 49-70.

[12] Rathje, E. M., Dawson, C., Padgett, J. E., Pinelli, J. P., Stanzione, D., Arduino, P., ... & Mosqueda, G. (2020). Enhancing Research in Natural Hazards Engineering through the DesignSafe Cyberinfrastructure. *Frontiers in Built Environment*, *6*, 213.

[13] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." *Advances in neural information processing systems* 25 (2012): 1097-1105.

[14] Liu, X., Iturburu, L., Dyke, S. J., Lenjani, A., Ramirez, J., & Zhang, X. (2022). Information fusion to automatically classify post-event building damage state. *Engineering Structures*, *253*, 113765.

[15] Altınçay, H. (2005). On naive Bayesian fusion of dependent classifiers. *Pattern Recognition Letters*, *26*(15), 2463-2473.

[16] Hirzel, A., & Guisan, A. (2002). Which is the optimal sampling strategy for habitat suitability modelling. *Ecological modelling*, *157*(2-3), 331-341.

[17] Chungwook Sim, Nick Skok, Ayhan Irfanoglu, Santiago Pujol, Mete Sozen, Cheng Song (2016), "Database of low-rise reinforced concrete buildings with earthquake damage," https://datacenterhub.org/deedsdv/publications/view/454.

[18] Prateek Shah, Santiago Pujol, Aishwarya Puranam (2016), "Database on Performance of High-Rise Reinforced Concrete Buildings in the 2015 Nepal Earthquake," https://datacenterhub.org/deedsdv/publications/view/409.

[19] Prateek Shah, Santiago Pujol, Aishwarya Puranam, Lucas Laughery (2016), "2015 Nepal Earthquake Building Performance Database," https://datacenterhub.org/deedsdv/publications/view/537.

[20] NCREE, Purdue University (2016), "2016 Taiwan (Meinong) Earthquake," https://datacenterhub.org/deedsdv/publications/view/534.

[21] Purdue University (2018), "Buildings Surveyed after the 2017 Mexico City Earthquakes," https://datacenterhub.org/deedsdv/publications/view/536.

[22] Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

[23] Ting K.M. (2011) Precision and Recall. In: Sammut C., Webb G.I. (eds) Encyclopedia of Machine Learning. Springer, Boston, MA. https://doi.org/10.1007/978-0-387-30164-8_652

[24] Davis, J., & Goadrich, M. (2006, June). The relationship between Precision-Recall and ROC curves. In *Proceedings of the 23rd international conference on Machine learning* (pp. 233-240).

5. CONCLUSIONS

Rapid reconnaissance data collection is a critically important tool that civil engineers use to identify gaps in design procedures and in construction practices. These data are collected at great expense by reconnaissance teams after each natural hazard event. Evidence from post event reconnaissance missions informs building code changes and suggests new research directions, and the amount of available data is growing rapidly.

Global positioning systems (GPS) are often used to provide useful spatial information for localizing image data. It is, however, challenging to collect such information when images are captured in places where GPS signals are weak or interrupted. The indoor spaces of buildings are a good example of such a case, and thus I developed other means to achieve this goal. Without such information, the data collector is typically the primary individual that is able to actually use such image data. At the same time, for users other than the data collector, it is difficult to know and document where each image was taken. Without spatial information, engineers are not able to conduct in-depth studies needed to understand the consequences of natural hazard events on our buildings and improve our building codes. This work aims to address this gap by developing methods to automatically link the location in the building where the image was taken with each of the images, and overall make the data more complete and useful to the engineer analyzing them.

This dissertation focuses on enabling the engineer in the field to automatically determine and document the indoor location of image data. By implementing and integrating techniques including, visual odometry, CNN-based image classifier, optimization, clustering, data fusion, etc., this tool is able to automatically process reconnaissance data that are collected from a large-scale area, and then provide the indoor location information for image data. Furthermore, the research in this dissertation provides a method to output the damage state evaluation for buildings based on a large volume of images. This work contributes to the post event reconnaissance and data documentation in the following ways:

- First, this work offers the capability to localize reconnaissance data on structural drawings in an automated and rapid manner. By automating all the processing steps, this work is able to take in the raw reconnaissance data as the input, then generates the outputs without any manual supervision. Processing speed is also taken into consideration in this work. Techniques are utilized to accelerate the most time-

consuming steps, the optimization of path overlay and the data fusion for damage state evaluation. Through demonstrations using data collected from real buildings, this work is shown to perform over the entire end to end process and provide the results in a reasonable amount of time. Most importantly, this process does not add extra effort as compared to the existing data collection protocol. Data collection does not require the use of expensive cameras or special 3D sensors. The only addition here is to mount an inexpensive motion camera to the data collector and keep it recording while data collection progresses.

- Second, this work increases the accuracy and robustness of data localization and herein data documentation based on indoor localization. Compared to manually work, the automated process is based on time step matching and optimization and can thus generate results with highly consistency. This avoids the human factor. Based on the reconstructed and overlaid path, the location of images can be specified to a narrow area even when consider the errors accumulated in the process. This approach has the potential to be further developed to deliver more accurate results than the locations based on data collector's memory or guess from the image content. Furthermore, this method can free engineers from performing the tedious work to manually provide location information that may consumes long time and lead to unreliable results.
- Third, in reconnaissance missions, an important task is to develop methods to classify the damage state of buildings after an event. However, the process of collecting the data can be both exhausting and dangerous for the reconnaissance teams, and efforts to increase the reuse of those data collected under such difficult conditions should be energetically pursued. The automated damage state evaluation developed in this dissertation will facilitate an increase in the amount of data collected and used to develop new knowledge, while also building greater confidence in vulnerability models developed from the reuse of such data. This component can also serve as a tool to extract information from a large dataset, support data documentation and organizing the data.

Researchers and engineers that are active in reconnaissance missions can get more out of their data than we presently do by leveraging these methods. Though the collection of the image data can influence the outcomes of such automated techniques, and thus there is value in

considering some suggestions to follow for collecting data that will yield robust results from these, and possibly other, techniques. Recommendations include:

- First, a lot more images should be collected in the field. Reconnaissance teams are encouraged to cover as much of the indoor areas of the buildings as possible and collect more images from each target building. As the number of images gathered from a given building grows, more knowledge can be extracted, and important lessons can be learnt. Most importantly, this includes collecting images about non-damaged building components. For a specific problem, the non-damaged building components, comparing to the damaged components, could yield the same or more amount of valuable information.
- Second, the images should cover as many building components as possible. For every visible building region, it is essential to know whether or not those components contain damaged or undamaged building components, structural or non-structural components, relevant or irrelevant to the damage state of buildings, etc. This work can make more robust predictions when buildings are sufficiently covered by the reconnaissance images.
- Third, images should not be taken from so close that the context of the scene is not clear. A close in view of a crack can be useful, but does not provide information about whether the damage is to structural or nonstructural components, nor does it provide any sort of scale information. Finally, the image data need to be collected with sufficient lighting. If the indoor environment is not illuminated well, it is recommended that the data collector bring extra lights and use these to illuminate the scenes.

The contributions in this dissertation will promote the collection and reuse of more reconnaissance data to inform building design procedures, and facilitates a much broader range of studies using image data. In general, this work provides a systematic tool for assisting the normal reconnaissance data collection missions. With this tool, a comprehensive and detailed spatial information of the images collected in the field are generated in an automated and rapid manor. This ability will increase the accessibility of post event reconnaissance images, supporting a safer built environment and accelerating the adoption of new design procedures and codes. In addition, this work provides the ability to analyze a vast volume of reconnaissance images to predict the damage state of buildings. I also anticipate this work has the potential to support the use of drones

or robots for field data collection, which in turn, reduces life-threatening situations for reconnaissance teams.

In the end, there are some future directions to consider to complete and extend the application of this dissertation.

- The current work is realized in an end-to-end relative coordinate system. Therefore, the preferred types of input data are from continuous data collection or data collection with stops but continues at where the previous the stops are at. If the same building or floors are visited in separate and for multiple times, it is difficult for this work to figure out the different portion or the same portion. All results have to be presented to engineers without further clarification. To address, data fusion would be the direction to explore. For example, scene recognition and similarity analysis could be engaged as a basis for such data fusion. Then, a complete data set could be presented.
- Though we emphasis the importance that this tool is able to work without any additional signals or systems due to the fact that post event environments normally lack of them, we should acknowledge in some scenarios signals are indeed available during reconnaissance missions. Thus, to utilize these signals, e.g., WIFI, telecom signal, GPS, etc. and integrate them into the tool could potentially increase the quality of the results and the applicability of the work.
- In this work, damage state evaluation for buildings is yet based on image classifiers where classification decisions only depend on the overall information possessed by the images and information of objects that are small or in the background of the images are commonly neglected. This inevitably leads to loss of information conveyed through the process and in the results. A possible way to overcome this is to resort to CNN-based object detection or instance detection techniques. Instead of yielding classification results on the image level, these techniques distinguish individual objects in the visible content and classify each of them and output the more meaningful and concrete results. These techniques can be combined with object level information fusion to upgrade the damage state evaluation step to a higher level.