Automated image localization to support rapid building reconnaissance in a large-scale area

Xiaoyu Liu[1], Shirley J. Dyke[1,2], Ali Lenjani[3], Ilias Bilionis[1], Xin Zhang[2], Jongseong Choi[4,5]

1 School of Mechanical Engineering, Purdue University, West Lafayette, Indiana, USA
2 Lyles School of Civil Engineering, Purdue University, West Lafayette, Indiana, USA
3School of Medicine, Stanford University, Stanford, California, USA
4 Department of Mechanical Engineering, The State University of New York, SUNY Korea, Incheon, South Korea
5 Department of Mechanical Engineering, The State University of New York Stony Brook University, Stony Brook, New York, USA

Abstract

Collecting massive amounts of image data is a common way to record the post event condition of buildings, to be used by engineers and researchers to learn from that event. Key information needed to interpret the image data collected during these reconnaissance missions is the location within the building where each image was taken. However, image localization is difficult in an indoor environment, as GPS is not generally available because of weak or broken signals. To support rapid, seamless data collection during a reconnaissance mission, we develop and validate a fully automated technique to provide robust indoor localization while requiring no prior information about the condition or spatial layout of an indoor environment. The technique is meant for large-scale data collection across multiple floors within multiple buildings. A systematic method is designed to separate the reconnaissance data into individual buildings and individual floors. Then, for data within each floor, an optimization problem is formulated to automatically overlay the path onto the structural drawings providing robust results, and subsequently, yielding the image locations. The end-toend technique only requires the data collector to wear an additional inexpensive motion camera, thus, it does not add time or effort to the current rapid reconnaissance protocol. As no prior information about the condition or spatial layout of the indoor environment is needed, this technique can be adapted to a large variety of building environments and does not require any type of preparation in the post event settings. This technique is validated using data collected from several real buildings.

1. Introduction

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 videos 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.

Due to the ease with which images can be taken, reconnaissance data collected during different events are being amassed in several large repositories. Although these images are clearly useful to the researchers who 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 goes unused. The rapid organization, analysis, and publication of these data are valuable activities for the hazards community.

In the Unites States, 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 (Datacenterhub, 2014; EERI, 2009; PEER, 2013; QuakeCoRE, 2016). However, these platforms do not offer functionalities to support researchers in sorting, classifying, organizing, and analyzing these data. Other platforms have been developed, for example, Automated Reconnaissance Image Organizer (ARIO), which are designed to provide automated image classification and report generation services (Yeum, Dyke, Benes, Hacker, Ramirez, et al., 2019).

Clearly, tremendous resources are devoted to reconnaissance image data collection and storage. However, the use of these data is severely limited without putting them into the proper context. For instance, spatial location information is often lacking in such image data. Without knowing the location where an image was taken, a researcher who did not collect the data cannot be certain where the image was collected within a given building. Without such location information, interpretation of the data, whether for a single image or an entire building, is challenging or impossible. For example, an engineer may need to know the condition of components of a building that are relatively close to each other, or of components at opposite ends of a building. Similarly, one may need to examine images of a column or wall that extends vertically through multiple floors in a given building. Estimating the location information among a group of images can consume a great deal of time and effort, and may lead to untrustworthy results.

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 (Kos et al., 2010). To address this issue, we have previously developed a technique to localize reconnaissance images on a single structural drawing (Liu et al., 2020). We used visual odometry (hereafter, VO) to reconstruct the walking path and associate it with the visual data. In this work, the step of overlaying the reconstructed path onto a drawing required manual user input, which is not preferred (Liu et al., 2020). To overcome this limitation, here we develop the ability to entirely automate these steps and evolve the single-floor image localization process into a fully automated multibuilding, multifloor 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 we 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 work, 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. 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. The integrated technique developed 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 the 3D path (hereafter, 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. For convenience, the abbreviations used herein are defined in Table 1.

TABLE 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

The remaining sections of this paper are organized as follows. Section 2 reviews the research relating to this work. Section 3 explains the technical approach and the key challenges encountered and overcome, mainly focusing on data separation and the automated overlay process. In Section 4, experimental validation of the individual components of the technique is performed, including indoor–outdoor separation, multifloor separation, and Path overlay. We then validate the entire technique with data collected over a large-scale area. The conclusions are documented in Section 5.

2 Literature review

Although to date no research has focused on tackling this problem, here we summarize past research conducted to look at tasks that are somewhat similar to those that we 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 (Bahl & Padmanabhan, 2000; Gutmann et al., 2013; Meng et al., 2012; Pierlot & Droogenbroeck, 2014; Want et al., 1992; Willis & Helal, 2004). 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. (2018) 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 (Espinace et al., 2010; Gupta et al., 2013; Quattoni & Torralba, 2009) 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. (2009) 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 our work. In other past work, Ni et al. (2013) 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. (2014) 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 (Holland, 1992) 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 (1995) 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 (Akhand et al., 2020; Dorigo et al., 2006; Yang, 2020). Comparisons among these have already been made, and serve to guide researchers in choosing the most suitable

algorithms. Hassan et al. (2005) 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 (2021) 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, we adopt PSO for our overlay problem for its quality, robustness, computation efficiency, and widespread availability.

3 Technical approach

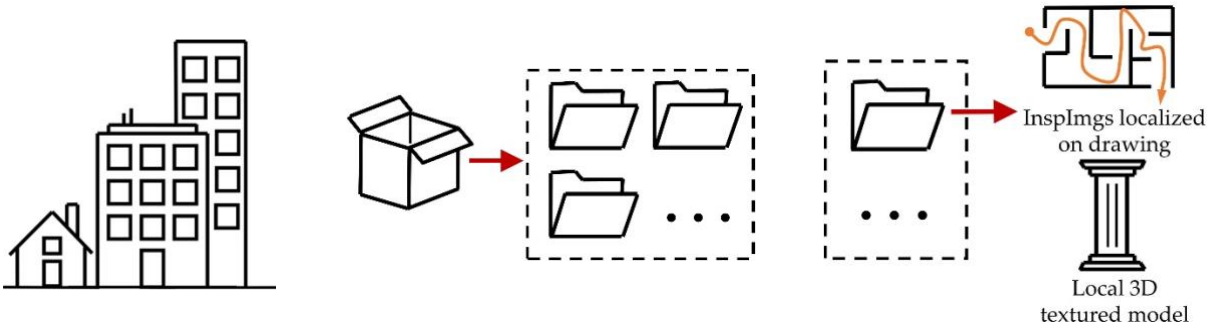


FIGURE 1. Overview of the technical approach

An overview of our automated technique is shown in Figure 1. The technique has three stages, including data collection, data separation, and data processing, each with its own challenges, which we 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. We 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, we 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, we 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.

3.1 Data separation

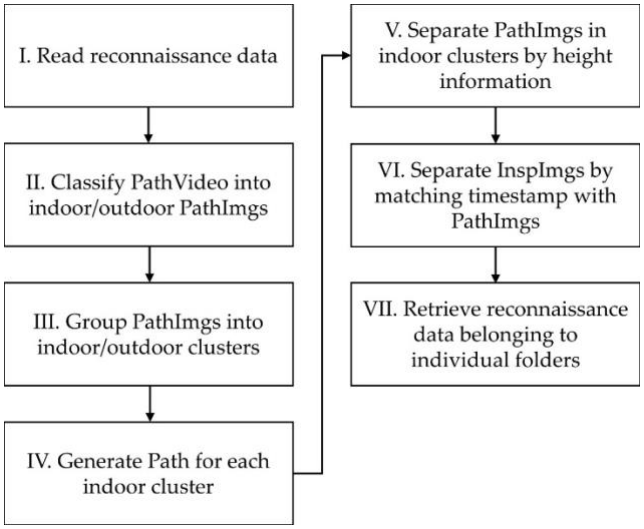


FIGURE 2. Workflow for completing the data separation stage

In this section, the details of our data separation method, Stage 2 in Figure 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 2. Note that all steps in this stage are completely automated. To begin with, the reconnaissance data are read in step 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 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 III, we aim to group the PathImgs according to their indoor or outdoor labels or probabilities from step 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 IV, Path is generated for each indoor group using VO technique (Engel et al., 2017). 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 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, we obtain PathImgs taken at each floor level.

In step 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 VII, we 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.

3.2 Path overlay in data processing

Stage 3 in Figure 1 is data processing where we 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 we 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 work 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 our approach and are thus encoded into the cost function discussed next.

3.2.1 Cost function formulation

The cost function is defined to quantitatively evaluate the quality of the overlay result. Thus, to define the cost function, we 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). We 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, we 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

$$p'(p;\phi) \coloneqq \begin{bmatrix} s\cos(\theta)\, p_x - s\sin(\theta)\, p_y + t_x \\ s\sin(\theta)\, p_x + s\cos(\theta)\, p_y + t_y \end{bmatrix} \qquad (1)$$

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, we 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

$$C_{\mathrm{Path}}(\phi; D) = \frac{1}{N_{\mathrm{Path}}} \sum_{p \in P_{\mathrm{Path}}} B(p'(p;\phi), D) \qquad (2)$$

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

$$C_{\mathrm{Pcl}}(\phi; D) = \frac{1}{N_{\mathrm{Pcl}}} \sum_{p \in P_{\mathrm{Pcl}}} E(p'(p;\phi), D) \qquad (3)$$

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) (Breu et al., 1995) 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, we 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

$$C(\phi; D) = \begin{cases} \alpha \cdot C_{\mathrm{Path}}(\phi; D) + C_{\mathrm{Pcl}}(\phi; D), \text{if } \phi \in \Phi \\ C^1_{\mathrm{penal}}, \text{if } \phi \notin \Phi \\ C^2_{\mathrm{penal}}, \text{if } p'(p; \phi) \notin \Phi(t_x, t_y), \text{for any } p \in P_{\mathrm{Path}} \end{cases} \quad (4)$$

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. 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, we simply set C^1_{penal} to $C(\phi; D)$, which is a penalty value set to 1100 in this work. 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), we set another C^2_{penal} to $C(\phi; D)$, which is a penalty value set to 2200 in this work. 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 work, Pcl filtering as explained in the next section, and so forth, α is set as 0.1 in this work to provide the best overlay results, although the method is not sensitive to this parameter. By minimizing the value of C, we obtain the values of the variables in ϕ that correspond to the optimal overlay result. Note that we 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 Search strategy

To avoid being trapped by local minima, we seek to form a search strategy that is highly likely to yield the global minima. Given the design of our problem, it is guaranteed that at least one optimal solution exists for this optimization problem, which corresponds to the optimal overlay result, and thus we 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, we form a derivative-free method to search for the best values of the variables and to obtain such a result quickly. Our 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, our method is able to achieve the global minimum with high robustness (this will be demonstrated in Section 4.1.3). In the next paragraphs, we explain our approach, and in the last paragraph of this section, we briefly discuss PSO and how PSO is integrated into our method.

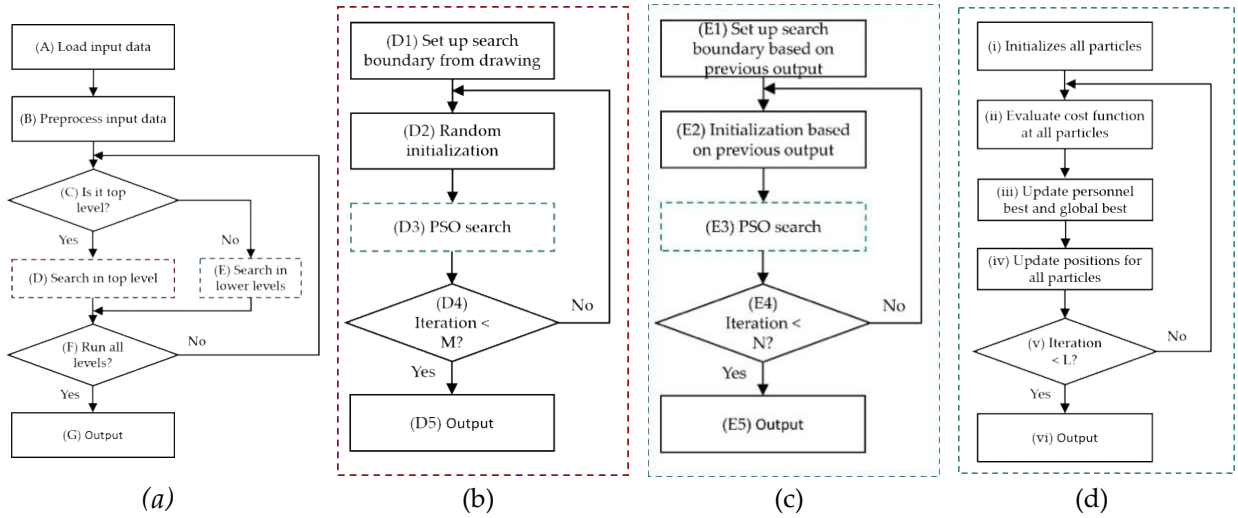


FIGURE 3. Workflow of the search strategy: (a) Overall workflow, (b) Detailed workflow of search in the top level, (c) Detailed workflow of search in lower levels, (d) Detailed workflow of PSO search.

The front-end workflow of the search strategy is shown in Figure 3a. Step (A) is to load the input data of a structural floor, including PathPcl and SDI. A series of preprocessing procedures are applied in step (B): (B-1) transform SDI to a binary image, as binary SDI; (B-2) generate a multilevel image pyramid of binary SDI (Adelson et al., 1984). 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. (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 (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 (Kaminsky et al., 2009). Step (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 (D), which applies the search at the top level. And if the answer is no, it goes to step (E) to carry out the search at the lower levels. Then, step (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 (G), where we obtain the output, the specific values of the variables ϕ yielding the optimal overlay result.

The details of step (D), to search in the top level, are shown in Figure 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 (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 (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, we 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 (D3) is to apply PSO search (Kennedy & Eberhart,1995) to find the global minimum of the cost function, as designed in Section 3.2.1. And in step (D4), a loop is carried out to repeat steps (D2) to (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 3.2.3. When the iteration meets its preset limit, M, the process goes to step (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 3.2.3.

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 3c. Step (E1) is to set up the search boundary based on the outputs from the previous level. In particular, we 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 (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 (E3). And then step (E4) is used to check whether the loop meets the iteration limit, N. Compared to M, N is a small number. In this work, N is set to 10. After N iterations, the process gives the output in step (E5).

The PSO algorithm used in step (D), search in the top level, and step (E), search in the lower levels, is shown as in Figure 3d. In PSO, we have 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 (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 (D2), random initialization, or from step (E2), initialization based on output from the previous level. In step (ii), we evaluate the cost function at the positions of each particle, yielding the corresponding values of the cost function. Then in step (iii), we 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. We 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 (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 (v), we determine whether the iteration meets its limit, L, from step (ii) to step (iv). When the iteration limit is reached, the output is generated in step (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 (iv) of Figure 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

$$V_{i,k+1} = w \cdot V_{i,k} + c_1 \cdot z_{i,k} \cdot \left(\hat{\phi}_{i,k} - \phi_{i,k}\right) + c_2 \cdot z_{i,k}^{+} \cdot \left(\hat{\phi}_{g,k} - \phi_{i,k}\right) (5)$$

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

$$\phi_{i,k+1} = \phi_{i,k} + V_{i,k+1} (6)$$

The question yet remains as to how to determine the hyperparameters used in the search, including M, L, P, w, c_1, c_2. This question will be discussed in Section 3.2.3.

3.2.3 Hyperparameter tuning

The search algorithm requires the hyperparameters to be selected before the optimization is performed. The choice of the hyperparameters may influence the reliability of the search algorithm and its computation time. Thus, we use a simple method to tune the hyperparameters and find appropriate values. We 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, we 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.

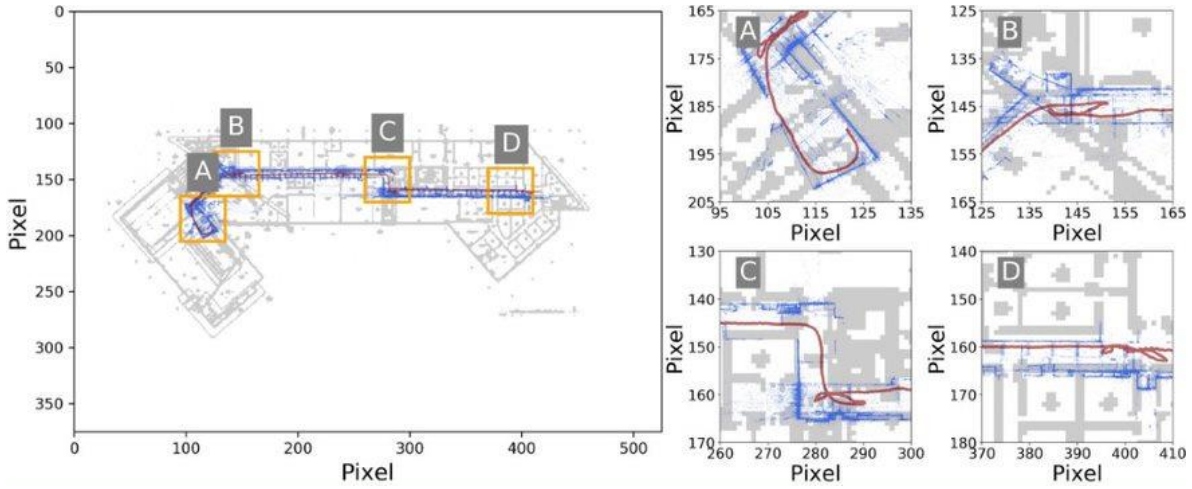


FIGURE 4. The overlay result with the global minimum of the cost function

We use the data collected in the underground floor in Armstrong Hall, Purdue, to tune the hyperparameters. The data generate 1428 Path points and 489,930 Pcl points. To save time, we perform the hyperparameter 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 hyperparameter 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 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.

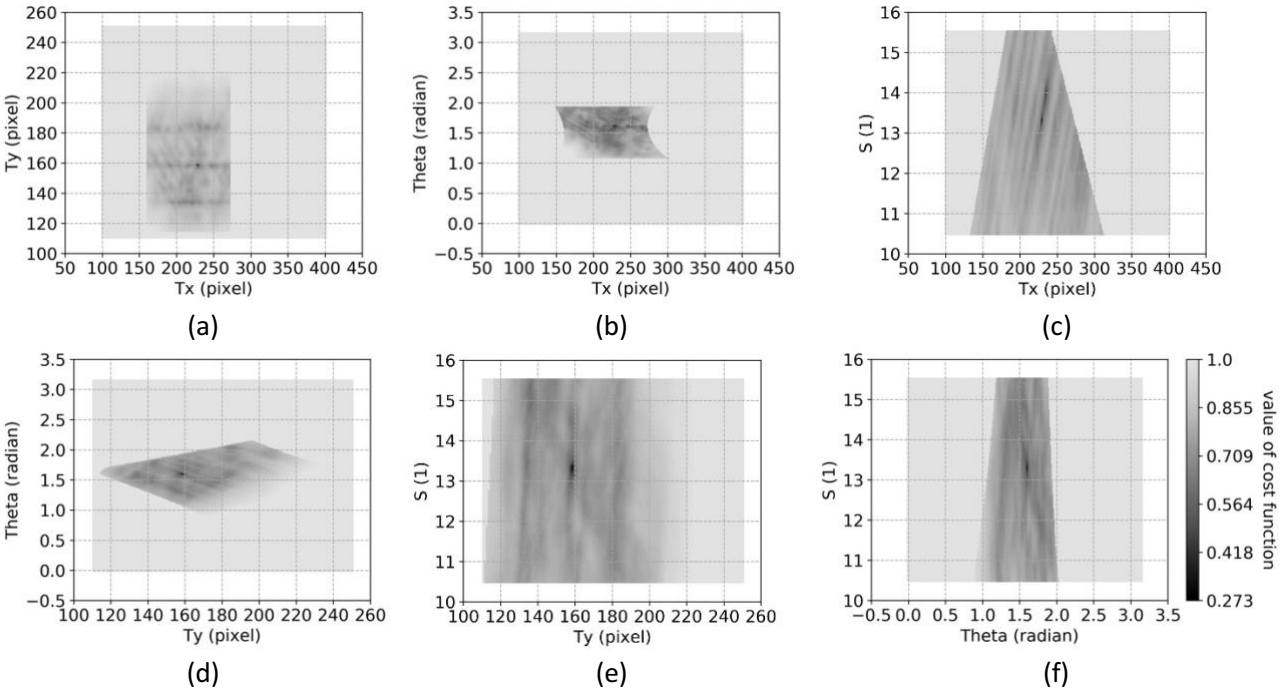


FIGURE 5. Results of validation by brute-force grid search

The next part of the process is to find the threshold. We 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. Sample results from the brute-force grid search are shown in Figure 5. To illustrate the behavior of the cost function near the optimal overlay result, we 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 5a as an example. For $t_x = 230, t_y = 158$, the point represents the global minimum. Notice in Figure 5a, the values of θ and s are set to achieve global minimum. This approach is simply for aiding visualization. As we 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 our method is to avoid falling into a local minimum. As mentioned in Section 3.2.2, 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, we easily 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, we use this global minimum and the corresponding value of the cost function (0.273) as the criteria in the evaluation discussed next.

TABLE 2. 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*
w	*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 hyperparameter tuning can begin. Candidate values of the hyperparameters are listed in Table 2. To save time, we 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, we pick the hyperparameters 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 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, we choose the first set of hyperparameter 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 hyperparameter values, our 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.

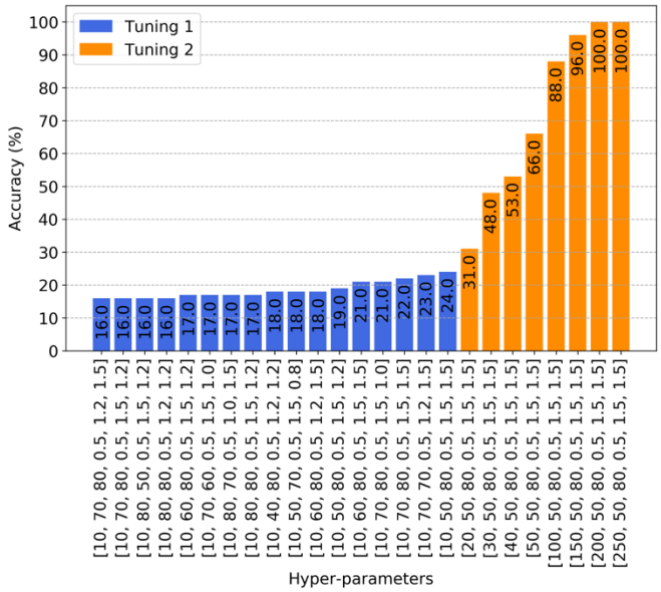


FIGURE 6. Result of hyper-parameter tuning

4 Experimental validation

The validation is divided into two parts. First, we individually verify each of the major steps in this technique. These steps include indoor–outdoor separation, multifloor 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, we 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.

4.1 Verification of essential steps

4.1.1 Verification of indoor and outdoor separation

As explained in Figure 2, step I to step III, we must process PathVideo and separate PathImgs into indoor and outdoor groups. This function is the first key component of this technique. Here we discuss the design of the indoor–outdoor image classifier, and the verification of the indoor and outdoor separation with several test data.

4.1.1.1 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, we 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 we used from each data set are listed in Figure 7. Data sets used in the training and testing include CDSE, SUN, DOIDE, and Indoor scene (Quattoni & Torralba, 2009; Vasiljevic et al., 2019; Xiao et al., 2010; Yeum, Dyke, Benes, Hacker, Gaillard, et al., 2019). Images from the two classes are not equally selected from each data set. Instead, we 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.

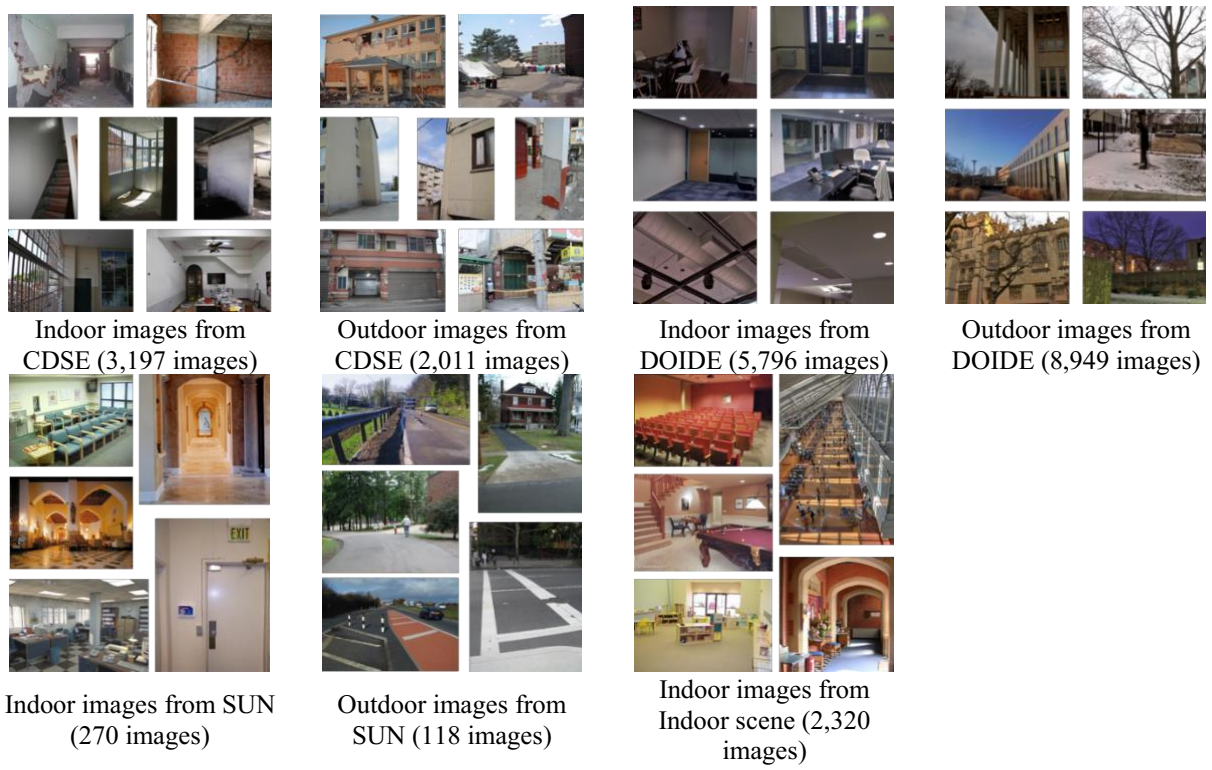


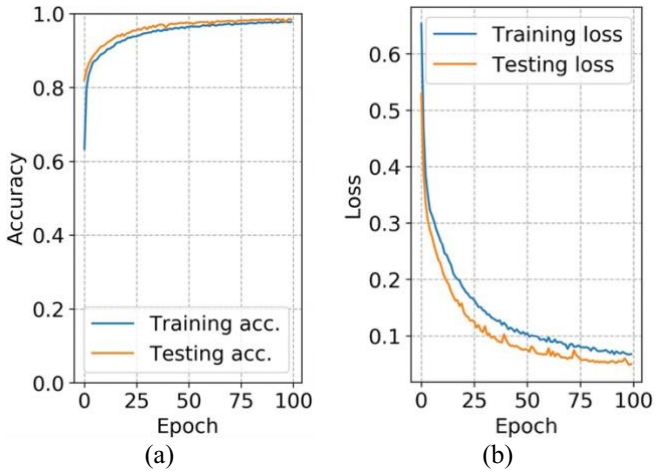
FIGURE 7. Sample images in the dataset (Yeum et al., 2019; Xiao et al., 2010; Vasiljevic et al., 2019; Quattoni & Torralba, 2009)

The structure of the classifier is configured based on a popular CNN model, VGG16 (Simonyan & Zisserman, 2014). 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, we balance the training of new weights with the use of the pretrained network. We use the pretrained VGG16 weight trained with ImageNet data set (Krizhevsky et al., 2012). 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 8. The classifier is trained for 100 epochs. The accuracy of the classifier in Figure 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 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. Clearly, our model achieves both high recall and precision in predicting indoor and outdoor classes.



(a) (b)

FIGURE 8. Training process of the indoor-outdoor classifier: (a) accuracy history, (b) loss history

TABLE 3. 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%*	

4.1.1.2 Results of indoor and outdoor separation

To increase the confidence in assigning the images into indoor or outdoor categories, we also develop a method we 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. We remove PathImgs that have probabilities between .1 and .9, and leave all of those remaining. Then we apply an unsupervised cluster on the remaining PathImgs. Here, we perform hierarchical clustering with the single linkage algorithm (Gower & Ross, 1969). To perform the cluster, the data are treated as 2D points, and the clustering is based on the 2D Euclidean distance between the points. We scale the image indices by a scaling factor, which is roughly the total number of PathImgs. Here, we

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, we remove any redundant clusters, which fall within other clusters. The remaining clusters are the final separation results.

We 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, we use one PathImg from every 200; 1039 PathImgs are used. The raw probability results from the classifier are shown in Figure 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 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 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, we can easily determine the boundary between the two groups in the PathImgs. As shown in Figure 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, we 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.

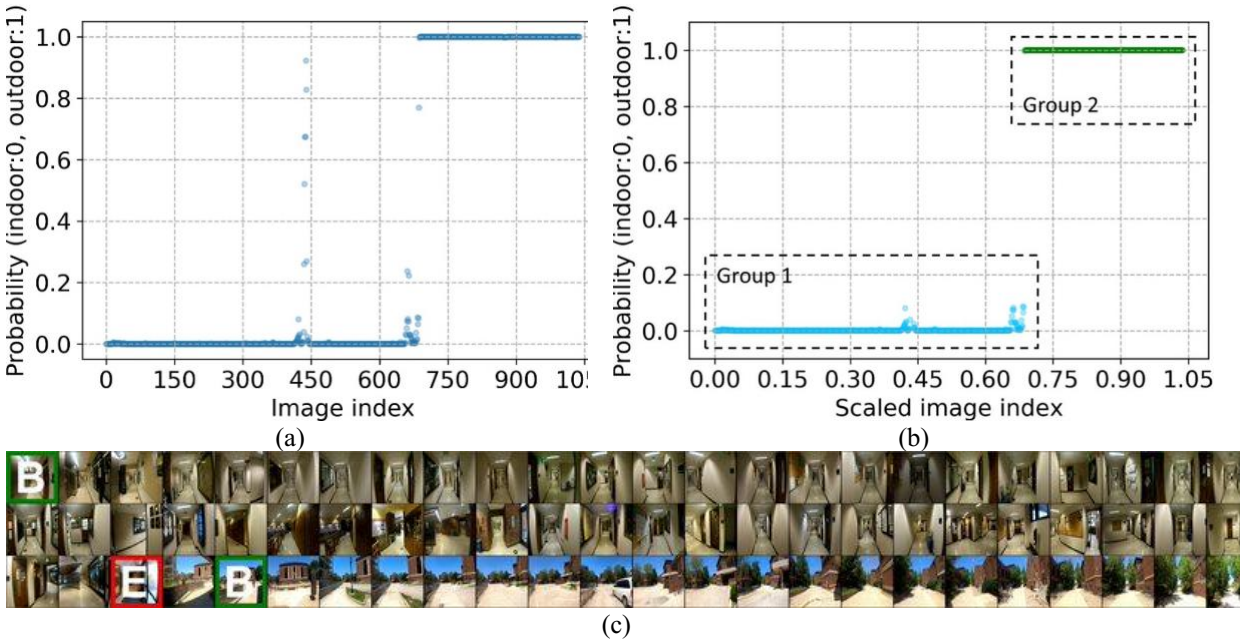


FIGURE 9. Results of indoor and outdoor separation with dataset 1: (a) probability, (b) final separation, (c) separation of PathImgs

Similarly, data set 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 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 10c.

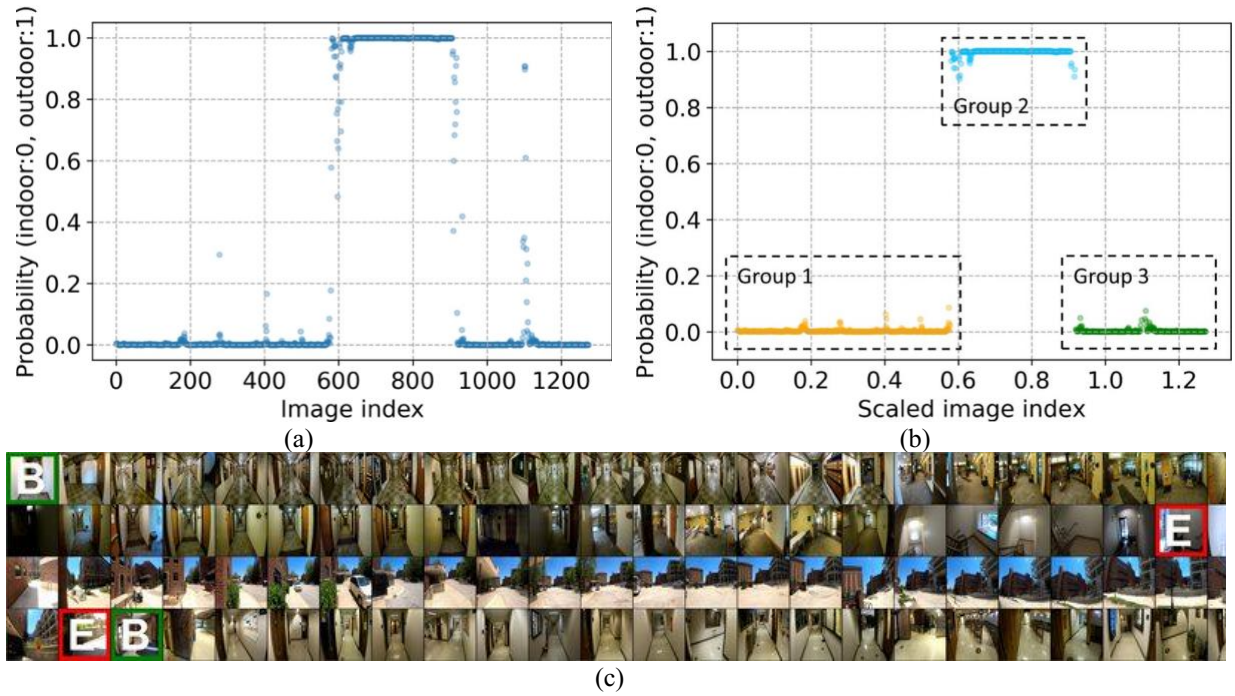
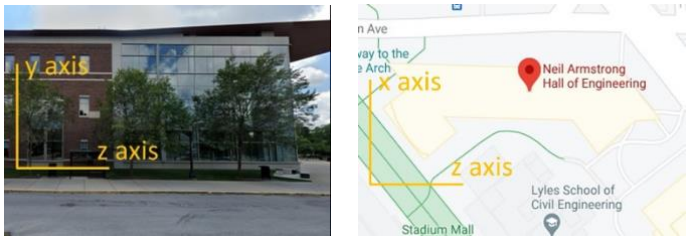


FIGURE 10. Results of indoor and outdoor separation with dataset 2: (a) probability, (b) final separation, (c) separation of PathImgs

4.1.2 Verification of multifloor separation

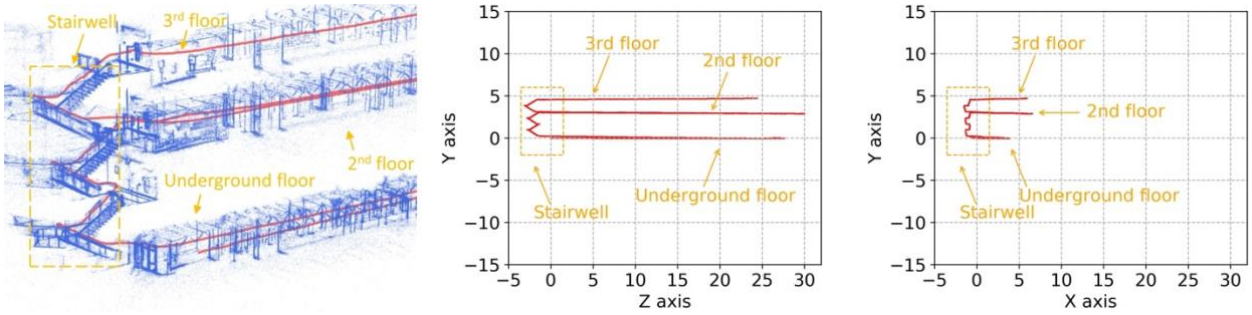
After indoor–outdoor image separation, we must process each indoor PathImg group. As in Figure 2, step IV to step V, if the PathImgs of one building contains data from multiple floors, we use the height information in the Path reconstruction to separate data collected at different floors. The multifloor 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 11. 3D coordinate system for the Path reconstruction: (a) $\boldsymbol{x-y}$ coordinate plane (Google Street View, 2021), (b) $\boldsymbol{x-z}$ coordinate plane (Google Maps, 2021)

Before demonstrating the results, we 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 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 12. 3D reconstruction of PathPcl of the dataset: (a) 3D reconstruction, (b) Path reconstruction in z-y plane, and (c) Path reconstruction in z-x plane

We use VO to rebuild the 3D PathPcl using PathImgs. The 3D reconstruction is displayed in the z–y coordinate plane, as in Figure 12a (Engel et al., 2017). 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 12b, and in the x–y coordinate plane as in Figure 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 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.

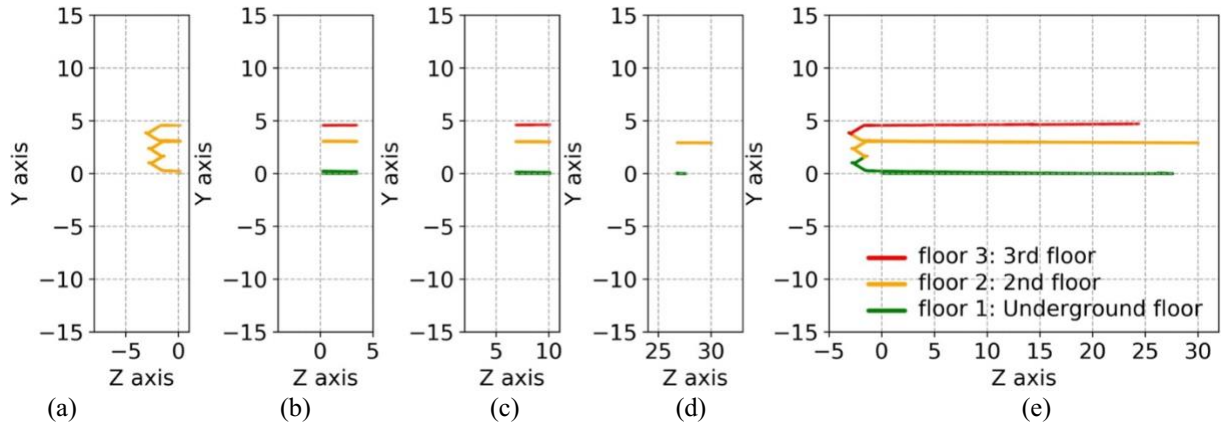


FIGURE 13. Clustering results of segments using unsupervised clustering and final separation results using supervised clustering: (a) cluster results of segment 1, (b) cluster results of segment 2, and (c) cluster results of segment 4, (d) cluster results of segment 10, (e) final separation results

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, we 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 4.1.1, we perform the hierarchical clustering with the single linkage algorithm (Gower & Ross,1969) based on the 2D Euclidean distance between Path points. Representative results of some segments are shown in Figure 13. Although the first segment is clustered into one cluster corresponding to the location of the stairwell, as shown in Figure 13a, most other segments yield the correct number of floors, three, as shown in Figure 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, we apply another supervised clustering method along the direction of y axis. Here, we adopt the K-Means clustering methods (Hartigan,1975). This process generates the final results shown in Figure 13e. As denoted in the figure, different colors represent path points belonging to different floors. It should be mentioned that we 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 multifloor separation of PathImgs of one indoor group is complete. It should be pointed out that, in a multifloor separation, for each floor, we also automatically cut a number of PathImgs (the number is set to 60 times step 200) from the beginning and the end after we apply multifloor separation. This is merely to remove Path in the stairwell and to avoid the possibility of including bad boundary PathImgs between floors.

4.1.3 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 multifloor separation step. PathPcl for each floor is reconstructed using VO (Engel et al., 2017), while structural drawings are saved as digital images. Prior to solving the optimization problem of Path overlay, we 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, we 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, we automate this overlay process. Hyperparameters are tuned and set up before the validation, as discussed in Section 3.2.3. The only term that will vary with the specific structural drawing is the total level of the image pyramid, in Step B-2 in Figure 3. Based on our 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 we use data collected from the underground floor in Armstrong Hall, which is part of the data used in Section 4.1.2. 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, 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 14. The red-colored points are the minimum value of the cost function in each iteration during the entire search process of our method. The orange-colored line corresponds to the global minimum value of the cost function with our method. Clearly, only one iteration or one PSO cannot guarantee reaching the global minimum. With our iterative scheme, the chance of reaching the global minimum is greatly increased. As a comparison, we 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 our method robustly finds the global minimum.

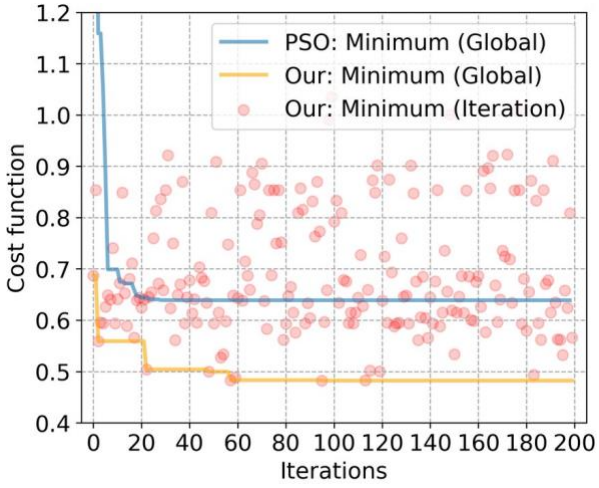


FIGURE 14. Cost function history at level 4

The results for level 4 and level 0 are shown in Figure 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 we have addressed how to reassemble such photos into a drawing (Yeum, Lund, et al., 2019). This stitched image can serve as the role of SDI in this work when a digital SDI is not available.

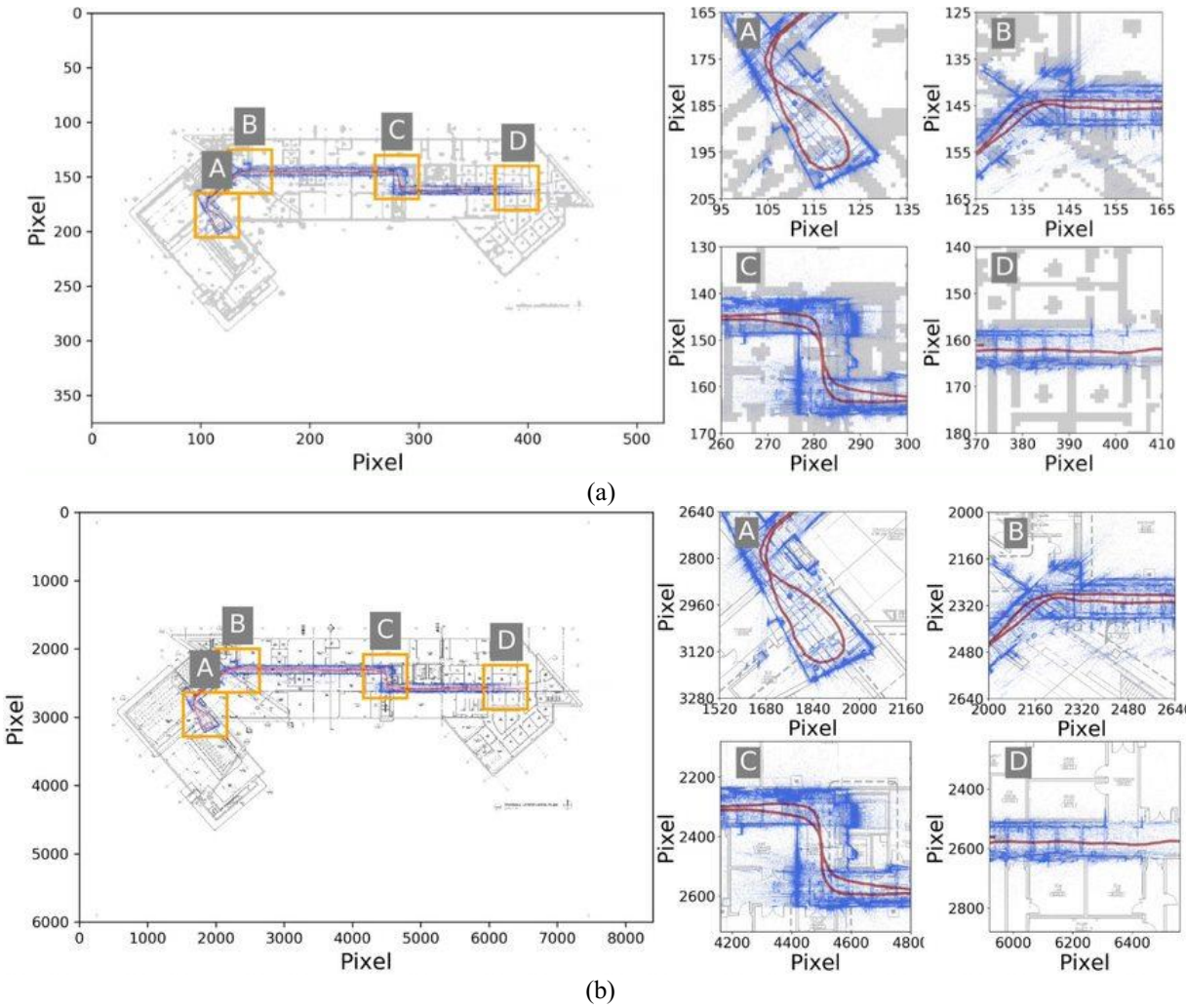


FIGURE 15. Results of automated overlay for underground floor of Armstrong Hall: (a) overlay results in level 4, (b) overlay results in level 0 (origin structural drawing)

4.2 Validation with large scale reconnaissance data

To assess the complete technique, we also perform an end-to-end validation. We 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 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. We 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.

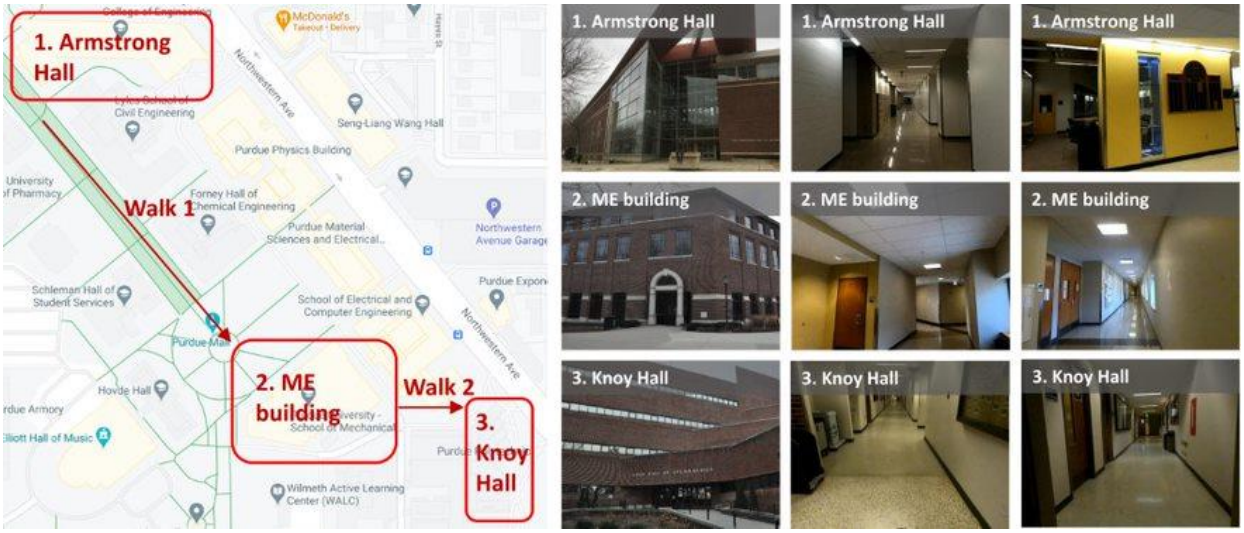


FIGURE 16. Buildings covered in the validation data (Google Maps, 2021)

In this experiment, we 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.

We 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 17. Again, we use one PathImg from every 200, and here 3806 PathImgs are used. In Figure 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 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 17c. Images with green-colored box correspond to the beginning PathImg for each group, while red-colored ones correspond to the ending PathImg.

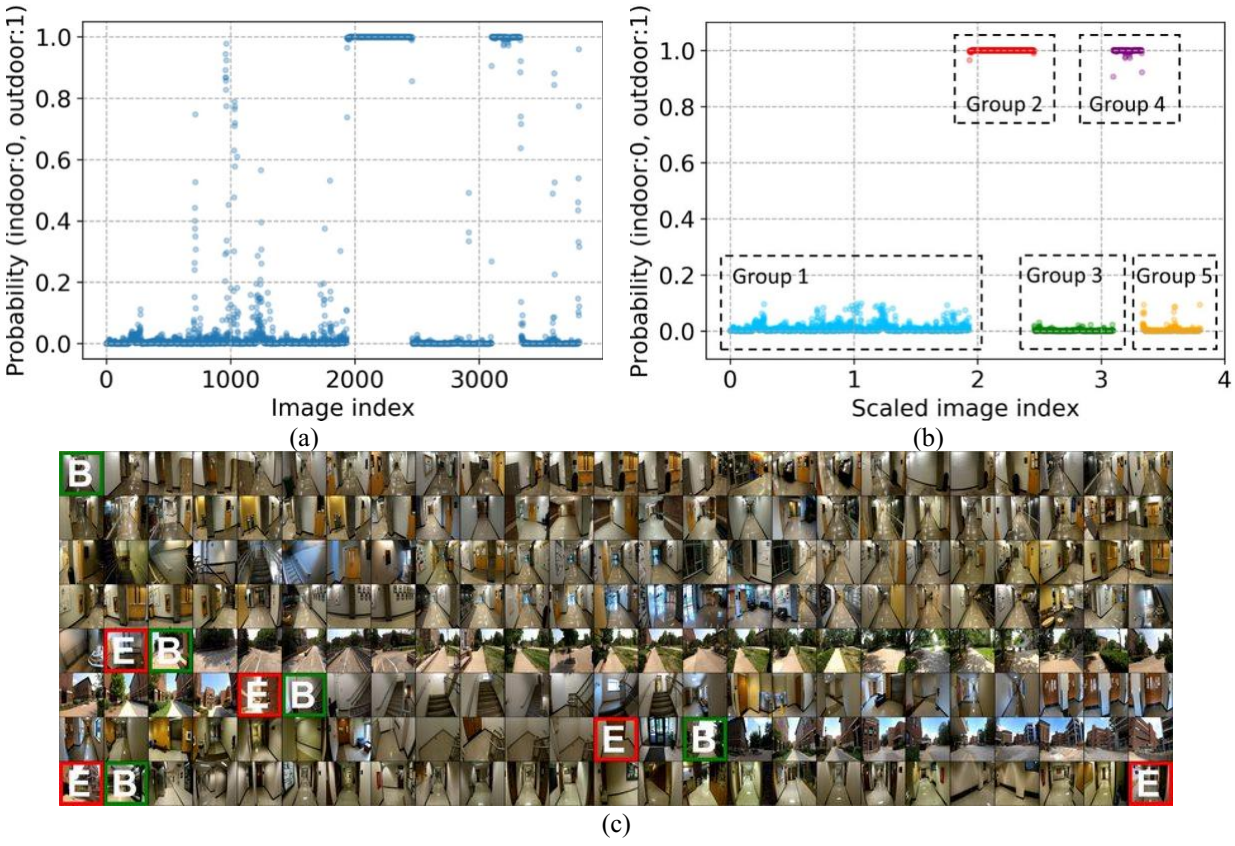


FIGURE 17. Results of indoor and outdoor separation: (a) probability, (b) final separation, (c) separation of PathImgs

After indoor–outdoor separation, we apply multifloor separation for each indoor group. We use VO (Engel et al., 2017) 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 multifloor separation results are shown in Figure 18 in the same coordinate system as Figure 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 we choose different stairwell as in Section 4.1.2 for the Armstrong Hall to justify multifloor 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, we determine the index of the PathImgs that define the boundaries for the PathImg group for a single floor.

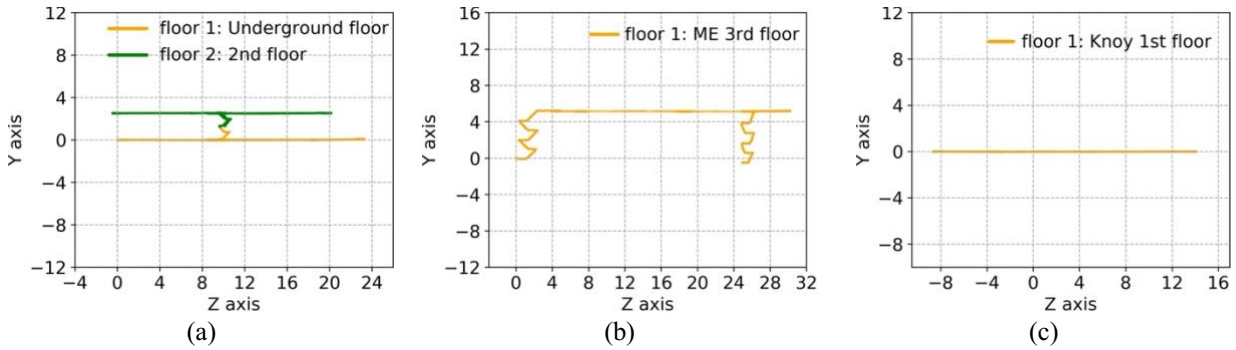
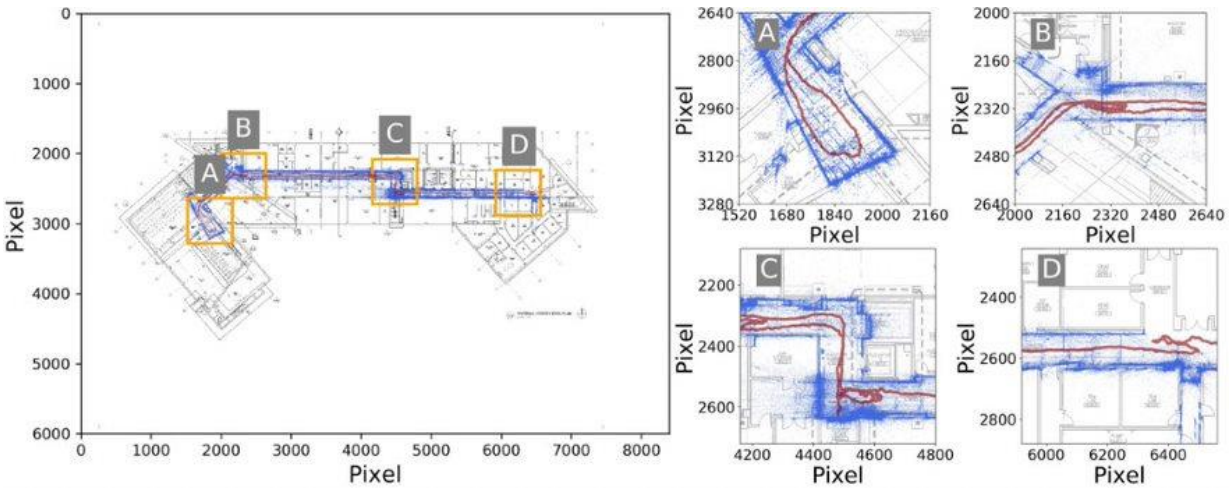
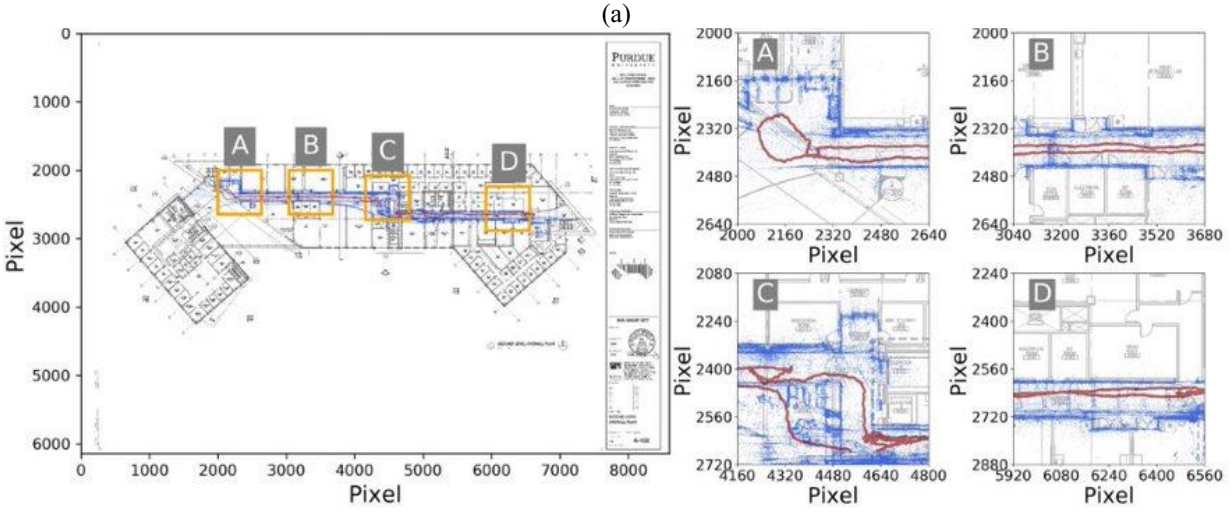


FIGURE 18. Results of multi floor separation for each indoor group: (a) indoor group 1, (b) indoor group 2, and (c) indoor group 3

On each floor, VO (Engel et al., 2017) 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 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)

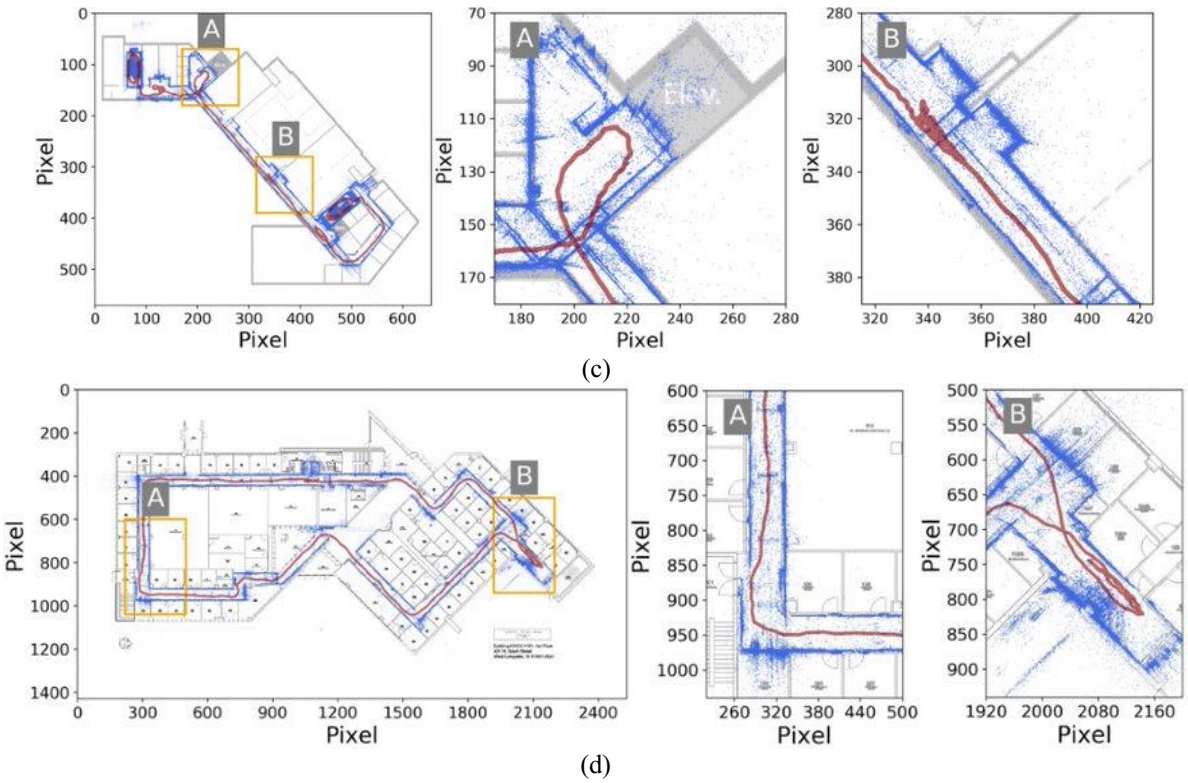


FIGURE 19. Overlay results of each floor: (a) results of underground floor in Armstrong Hall, (b) results of 2nd floor in Armstrong Hall, and (c) results of 3rd floor in the ME building, (d) results of 1st floor in Knoy Hall

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 20. Using data for the second floor in Armstrong Hall, we illustrate how one InspImg and its location can be obtained and plotted on the overlaid structural drawing. The InspImg is shown in Figure 20a. The 3D textured model reconstruction at the selected InspImg is also shown in Figure 20b. This model is constructed using the InspImgs and PathImgs that are identified as being within a specific range of the selected InspImg. Here we 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 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.

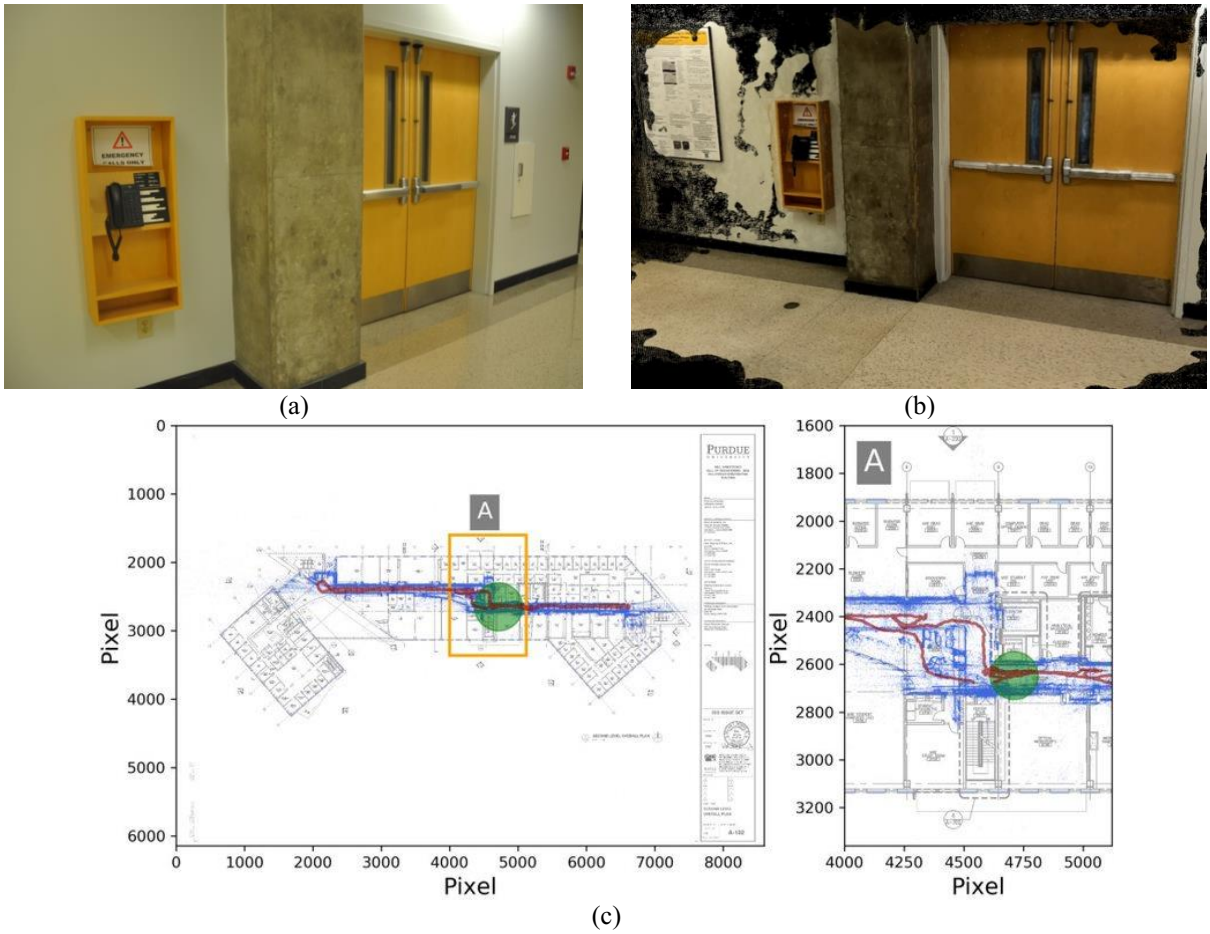


FIGURE 20. Representative results of image localization and local 3D textured model generation: (a) selected InspImg, (b) reconstructed local 3D textured model, and (c) its location on the structural drawing

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.

To obtain successful results with this technique, some recommendations are provided:

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 multifloor separation, there may be ambiguity regarding how to determine whether an indoor group corresponds to a multifloor 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 our 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.

5 Conclusion

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. However, due to the time involved in organizing the images, currently the data collector is typically the primary individual that is able to actually use such data.

We aim to alleviate this constraint by enabling the engineer in the field to automatically determine and document the indoor location of image data. The new technique described herein can automatically provide indoor localization of image data collected during such a mission. The inputs to the technique include the indoor image data and a single continuous video stream. The output is the structural drawings overlaid with the path walked and the location of each image collected. Collecting the data needed to deploy this technique does not alter the normal data collection procedure or add significant cost to conducting the mission. The data collector only needs to carry an additional motion camera to record a PathVideo. With this added video stream, we developed an integrated technique that can separate the input data by individual buildings and floors, reconstruct the path, and use overlay this information onto the structural drawings by solving an optimization problem. We formulate the optimization problem here by designing a suitable cost function that places the path on the structural drawings and links the images to their 3D position in the building.

Both the individual steps, and then the entire technique, are validated using data collected from real buildings. This automated technique provides simple tools to increase the accessibility of post event reconnaissance images, supporting a safer built environment and accelerating the adoption of new design procedures and codes.

ACKNOWLEDGMENT

We acknowledge that this work is supported by U.S. National Science Foundation OAC Program under grant no OAC-1835473.

REFERENCES

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.

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.

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

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.

Datacenterhub. (2014). https://datacenterhub.org/

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

EERI. (2009). Earthquake clearinghouse, Earthquake Engineering Research Institute. https://www.eeri.org/

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

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.

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

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

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

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.

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.

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.

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

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).

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

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.

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.

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

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

LeCun, Y., & Bengio, Y. (1995). Convolutional networks for images, speech, and time series. In M. A. Arbib (Ed.), The handbook of brain theory and neural networks (Vol. 3361, p. 10). MIT Press.

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.

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.

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.

Ni, K., Armstrong-Crews, N., & Sawyer, S. (2013, May). Geo-registering 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.

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

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.

QuakeCoRE. (2016). http://www.quakecore.nz/

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

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

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

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.

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

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.

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.

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

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.

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.

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.

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.