Available online at www.sciencedirect.com

ScienceDirect

Computer Vision and Image Understanding 00 (2017) 1–20

CVIU

Temporal City Modeling using Street Level Imagery

Ken Sakurada[a,*], Daiki Tetsuka[b], Takayuki Okatani[b,c]

[a]*Graduate School of Engineering, Nagoya University, Furo-cho, Chikusa-ku, Nagoya, Aichi, Japan*
[b]*Graduate School of Information Science, Tohoku University, 6-6-01 Aramaki Aza Aoba, Aoba-ku, Sendai-shi, Miyagi, Japan*
[c]*RIKEN Center for Advanced Intelligence Project, 1-4-1 Nihonbashi, Chuo-ku, Tokyo, Japan*

Abstract

Estimation of the temporal changes to a city is useful for city management, disaster recovery operations, and understanding natural phenomena. When several types of data are available for this task, the optimal type should be chosen depending on the changes that need to be detected. However, data of the desired type are not always available, particularly historical data. In this study, we propose two methods for detecting changes in a city, which can be used in complement to process available data types and detect changes in selected targets. The first method estimates the presence of buildings by comparing street-level images and a 2D city map of buildings created at different points in time. This method uses the Structure from Motion (SfM) technique to reconstruct a point cloud of the structures of the city, and matches the point cloud with the 3D building structures recovered from its 2D map. While 2D city maps are available for most cities, most are not very accurate. Therefore, this method is designed to overcome these inaccuracies and thus is widely applicable. On the other hand, the method cannot detect the following types of scene change: wall paintings, buildings that were reconstructed and closely restored to their previous shape, pedestrians, cars, and vegetation. The second method uses a pair of street-level images that are roughly aligned with GPS data collected at different points in time to detect such scene changes. This method uses the features of a convolutional neural network (CNN) in combination with superpixel segmentation to address inaccurate image alignment and it also enables change detection with pixel-level accuracy. Additionally, the second method is scalable for large-scale estimation because it can quickly detect scene changes by merely using an image pair without performing large-scale SfM. The authors consider the proper use of these two methods to enable temporal city modeling in various situations. We experimentally apply these methods to cities damaged by the tsunami that struck Japan in 2011 and the results show their effectiveness.

Keywords: Change detection, SfM, CNN, City-scale, Vehicular imagery, City map

1. Introduction

This paper describes two novel methods for estimating temporal change in a city using street-level images. The first method estimates the existence of buildings by using street-level images and a 2D city map taken from a different time period. The second method detects scene changes from a street image pair taken at different points in time.

Temporal city modeling is one of the most important tasks for managing a city, recovering from disasters, and understanding natural phenomena. Accurately understanding the changes in a city requires detecting of changes between data collected at different times. However, data obtained at a specific point in time, especially historical data, are not always available. Indeed, for the cities damaged by the tsunami that struck Japan in 2011, no street

*TEL: +81-52-789-3942, FAX:+81-52-789-4696, E-mail: sakurada@nagoya-u.jp

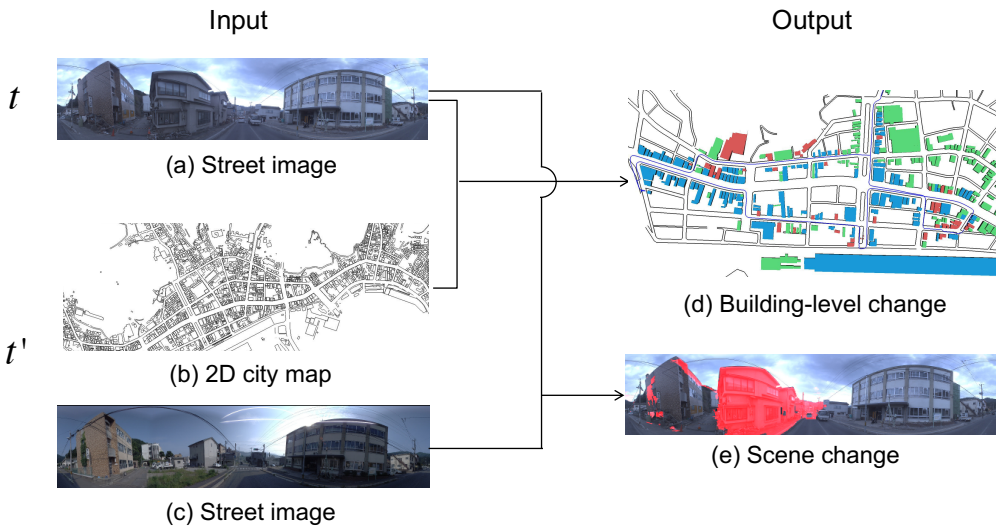


Figure 1. Inputs and outputs in our strategy. When (a) the street image for time t and (b) a 2D city map for time t' are available, we can estimate (d) building level changes. When street images for both time (a) t and (c) t' are available, we can estimate (e) the scene change.

images of any type (e.g., Google Street View) are available before the tsunami; the only available data are 2D city maps. Consequently, detecting changes in textures (e.g., wall paintings) and objects not present on a city map (e.g., pedestrians, cars, and vegetation) inevitably requires the comparison of images taken at different points in time, if they are available. As an example, if the shape of a reconstructed building has remained largely unchanged, it is difficult to detect the changes resulting from its reconstruction from the street images and 2D city map data taken at different time points, because the shape of the 2D city map is neither sufficiently accurate to detect small changes nor does it contain textural information.

Therefore, we propose two methods for detecting city changes depending on the available data types and the target changes. Figure 1 shows our strategy for different types of input data. We can estimate building-level changes when the street image for time t and 2D city map for time t' are available. Figure 2 presents visual images of a city prior to the tsunami and the subsequent changes subsequently caused by the tsunami, as obtained by this method. When street images for both time t and t' are available, we can estimate the scene change. The proper use of the two methods enables temporal city modeling for a variety of situations.

More specifically, the method for estimating changes at the building-level reconstructs the point cloud of city structures using the Structure from Motion (SfM) technique, and matches the point cloud with the 3D building structures recovered from the map. There are several difficulties faced when using this approach, including the inaccuracy in the recovered building structures, large differences in observation and thus in the point cloud size of individual buildings, and the mutual dependency of the existence of buildings due to potential occlusions. These problems are addressed by developing a model that represents the generation of a point cloud that sequentially utilizes SfM, a building wall observation model, and a greedy iterative approach to address the mutual dependency.

Most previous approaches for scene change detection require a 3D scene model and/or pixel-level registration between different temporal images. These approaches (as well as the first method) are computationally expensive for estimating city-scale changes. The second method does not have these problems because it uses the features of a convolutional neural network (CNN) in combination with superpixel segmentation. Comparison of CNN features produces a low-resolution map of scene changes that is robust to illumination changes and viewpoint differences. Superpixel segmentation of the scene images is integrated with this low-resolution map to estimate precise segmentation boundaries for each change. Hence, it can quickly detect scene changes from an image pair aligned using GPS data, making it possible to process an entire city with a single workstation.

The specific targets of our study are cities that were seriously damaged by the tsunami resulting from the Great East Japan Earthquake, which occurred on March 11, 2011. (Most of the buildings in these cities are at least five meters high, an assumption included in our building-level change detection method.) This event can be considered

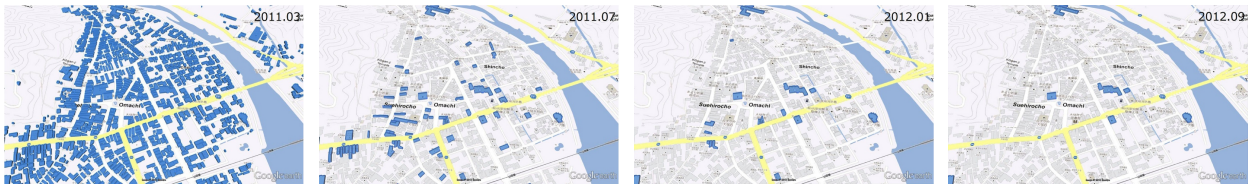


Figure 2. Visualization of temporal changes in a city. The left image shows buildings before the disaster; the other images show the results obtained by our method using several image sequences captured at different times. These images visualize how buildings are initially removed by the disaster and then remaining buildings are gradually demolished in recovery operation.

an unprecedented disaster in human history, as the tsunami forced many modern cities to change their structures in such a short time frame. Hence, visualizing the damage they incurred and the recovery process has many applications. Looking beyond this study, it should be noted that there is considerable opportunity to employ computer vision techniques in these types of disaster-related applications. Moreover, the development of computer vision techniques would also benefit from targeting the tsunami-damaged areas of Japan, because of the variety of damage and recovery that has occurred. These events provide several ideal test cases for enabling the development of new techniques. Both methods are applied to the tsunami-damaged areas and our experimental results demonstrate their effectiveness.

The paper is organized as follows. In Section 2, we summarize related work. Section 3 presents the building-level change detection method, which uses street images and a 2D map. Section 4 describes the scene change detection method, utilizing an image pair. Section 5 concludes this study.

2. Related work

Many studies have examined the problem of detecting building-level or scene changes by using either or both images and a maps [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]. These studies can be classified into several categories depending on the type of scene changes detected, methods, available information etc.

A standard approach is to detect changes in the 2D (image) domain [7, 11]. A typical method involves creating an appearance model of a scene from a set of images captured at different times, against which it compares a newly captured query image to detect changes. However, irrelevant changes can appear, such as differences in illumination; addressing this problem is a major concern in studies of this nature. Additionally, this approach usually requires the images to be captured from the same viewpoint, and is thus unable to process query images captured from different viewpoints.

Some studies have formulated the problem in the 3D domain [7, 8, 9, 12, 13] by building a model of the target scene in a "steady state" and comparing a query image against it to detect changes. A 3D model of the scene is often created by using a 3D sensor, rather than cameras. In [12], to estimate the existence of a building, the edges extracted from its aerial images are matched with the projection of its 3D model.

Studies that use a large number of multi-view images of a scene are used to create its spatio-temporal model leverage the SfM method. Schindler et al. proposed a method that uses many images of a city that were taken over several decades [14]. Their method can perform several types of temporal inferences, such as estimating when each building was constructed. The recent work of Matzen and Snavely [15] is essentially similar to that of Schindler et al. Their method uses photo collections from the Internet to detect 2D changes in a scene, including changes in the advertisements and paintings on building walls. If sufficient number of multi-view images of a scene are available, both methods use SfM to reconstruct the 3D scene models of a scene [1, 2, 3, 4, 5, 6].

The work of Taneja et al. is the most closely related to our method for detecting changes at the building-level [13, 16]. Their method also estimates building-level changes from the images and a map; however, it uses 3D maps that are obtained from multiview images or other sensors. This is a practical approach because their study is motivated by the need to develop a cost-effective method for updating 3D maps [13]. Thus, their method basically considers the detection of small changes, rather than drastic changes (e.g., those caused by a natural disaster), which have a crucial influence on the camera pose estimation between images taken at different points in time. Furthermore, as mentioned above, there is a difference in terms of data availability, especially for historical data. Due to these differences, our method reconstructs 3D building shapes from the 2D city maps. The 3D structures recovered from the 2D maps are

necessarily inaccurate. Although the method developed by Taneja et al. can overcome some inaccuracy in building structures, the inaccuracies in our case (e.g., in tsunami-damaged areas) are beyond the capability of their method. Furthermore, in situations in which the image sequence does not cover every street, our proposed method models potential occlusions among buildings and attempts to estimate the existence of the maximum number of buildings with maximum accuracy.

Our method for detecting changes at the building level approximates that of Schindler et al. [14], who attempted to detect changes on the city scale using the point cloud recovered from images by SfM. Given several photographs of a city from which a precise timestamp is absent, their method creates a 4D city model and estimates the time at which the photographs were taken. Additionally, their method requires 3D building models (although the 3D models could be created from the point cloud provided certain conditions are met). As in our method, their method considers potential occlusions among buildings; thus, the mutual dependency of the existence of buildings does not present a problem. However, their method employs a brute force approach (i.e., MCMC) to overcome this dependency because the images used in their method are sparse snapshots. Our method attempts to resolve this problem more efficiently by leveraging the nature of the images we use, i.e., a continuous sequence of images that are captured by a camera mounted on a vehicle moving along the streets of a city.

The scene change detection method discussed in this paper is categorized as a 2D method. This method detects scene changes by comparing a pair of (omnidirectional) images. The viewpoints of the images can be several meters apart at most, since they are captured from a vehicle running on a street. Assuming that a pair of the images taken at different points in time is aligned to this accuracy (using GPS data), the method is designed to distinguish changes, excluding irrelevant changes that occur owing to the difference of camera viewpoints, illumination and photography conditions. The method does not require a dense 3D scene model or a large number of images to perform SfM and reconstruct a 3D model.

3. Building-level change detection in a city using street images and a 2D city map

This section presents a method that was developed to use street images in conjunction with a city map to identify city-scale, building-level changes. More specifically, this method uses input in the form of a sequence of street images captured by a vehicle-mounted camera in combination with a city map to estimate the existence (or disappearance) of buildings.

This method uses a 2D map in which each building is represented as a polygon. This approach was chosen because 2D maps are available for most cities and residential areas, whereas 3D maps are only available for some major cities, which is the case for the areas in Japan that were damaged by the tsunami. It is important to note that, in 2D maps of this nature, individual buildings are represented as approximated polygons of the ground projection of the building; thus, 2D maps do not provide accurate 3D building structures.

Apart from the information obtained from the 2D maps, this method also uses the image sequences captured with a vehicle-mounted omnidirectional camera. The camera, which captured the images every few meters on the streets, synchronously recorded GPS data with each image. An image archive created in this manner for the tsunami-damaged area is used in our experiments. Figure 1 (a) shows an example of our image archive. Furthermore, aerial imagery could be effective for detecting building-level changes, instead of or in addition to street-level imagery. In this study, the use of aerial imagery was deliberately avoided, because street-level imagery captured by a vehicle-mounted camera is more frequently updated and cost effective to acquire [17, 18]. In addition, the imagery captured using the camera-equipped vehicles provides high-resolution street level information that is not available from aerial imagery.

However, the disadvantages of using street-level imagery must also be accounted for. Notably, problems arise from the occlusion of a building by surrounding buildings, in which case it is impossible to estimate its existence. This problem can be addressed by running the vehicle along all city streets; however, this approach would be costly and often impossible for various reasons (e.g., street closures to allow construction work). Thus, we consider maximizing the accuracy of building existence estimation, as well as the number of buildings being estimated, provided by a sequence of images captured with a limited travel distance.

Thus, the method estimates the existence of buildings on the map in pursuit of this goal. The method first performs SfM to reconstruct a point cloud from the images, and then compares the point cloud with the map. The principle underlying this method is to match the point cloud with the 3D shape of each building recovered from the 2D map,

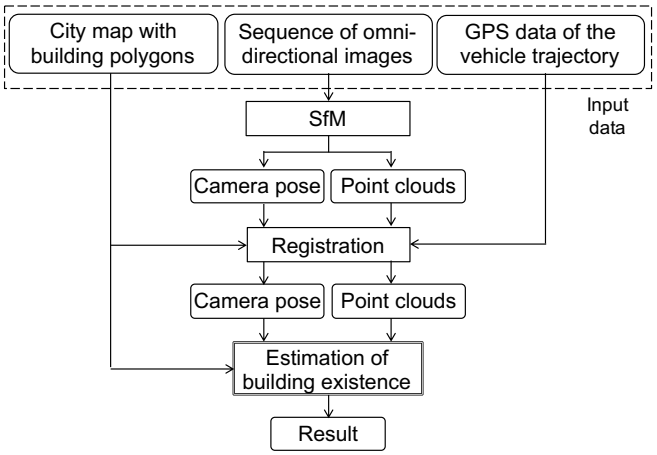


Figure 3. Data flow diagram of the proposed method.

which is represented as a polygon, and then estimate its existence by evaluating how points emerge around the building. Although this is a simple idea, this approach presents several challenges, such as the recovery of inaccurate building structures from the map, large differences in the number of points reconstructed from individual buildings, and the mutual dependency of the existence of buildings because of potential occlusions. We address these difficulties by modeling the process by which the point cloud is generated using SfM and also the observation of each building wall in the images, to evaluate the probability of the existence of each building in a greedy, iterative fashion.

3.1. Problem formulation

Figure 3 shows the data flow diagram of the proposed method, which takes the following three inputs: an omnidirectional image sequence, GPS data associated with the images, and a 2D city map. The method produces either a binary value or a continuous probability of whether each building in the map exists as its output.

The three types of input are described as follows: First, the sequence of omnidirectional images is captured by a vehicle-mounted camera (Ladybug3, Point Grey Research Inc.). An image is captured every two meters while the vehicle drives along the streets of the designated city. Second the GPS data associated with the image contain the camera position at which each image is captured. We render the omnidirectional images as cylindrical panoramic images for SfM; Figure 1 (a) shows an example.

Third, the 2D maps we use in the experiments contain polygons of buildings projected to the ground as shown in Figure 1 (b). The maps do not contain 3D information or building heights data. 3D shapes are recovered by assuming that each polygon represents the outer walls of a building and that the walls are at least five meters high. The 3D shapes obtained in this manner are inaccurate for several reasons, which must be considered in subsequent processing.

Figure 3 shows the pipeline of the method in which the three inputs are used. First, SfM [19, 20, 21] is performed using the omnidirectional image sequence that generates the camera trajectory and the point cloud of various city structures. Secondly, the camera trajectory and point cloud are registered to the city map using the GPS data associated with the camera trajectory. Section 3.2 describes these preprocessing details. Finally, the existence of each building on the map is estimated using the registered camera trajectory, point cloud, and map.

The problem we consider next is the last step in the process, which is shown as the box with a thick outline in Figure 3. This step matches and compares the registered point cloud with the 3D shape of the building recovered from the map to estimate whether each building exists. If a building exists, a number of points should be reconstructed around its walls; otherwise, there should be no point. As mentioned above, the building shape is represented as a polygon whose lines represent the walls of the building. In our method, each building is decomposed into walls, and each of them is matched and compared with the point cloud. Then, the wall results are combined to estimate whether the building exists.

Figure 4. Illustration of differences in the observation of individual buildings (best viewed in color). A building close to the camera trajectory (the green curve) has a large viewing angle (the red triangle). A building can be occluded by others (the blue triangles). How a building is observed in the images determines the size of the emerging point cloud. Observe also that potential occlusion makes the existence of buildings mutually dependent.

3.2. Preprocessing: SfM and registration

We employ a standard approach for SfM. Feature points are first extracted using SURF [22] in each image of the sequence, and are then matched based on descriptor similarity to obtain putative correspondences between each neighboring pair of images. The outliers are discarded by using RANSAC with a five-point algorithm [20], yielding a number of point trajectories and initial camera pose estimates. Their 3D positions are then estimated by triangulation, followed by the application of a robust bundle adjustment.

It should be noted that we match feature points only between consecutive images in the sequence; thus, once a scene point ceases to be tracked (e.g., when it is outside of the image area), the same scene point is treated as a new point if it has reappeared in images. The use of an appropriate matching scheme ensures that these points are correctly matched and identified as a single scene point, which enables loop closure and this could improves reconstruction accuracy. This approach is not intended to avoid the resulting increase in computational cost. We perform SfM in an open-loop manner, thereby ensuring a highly sparse structure resulting in a small computational cost. This is important as we are considering city-scale reconstruction. Moreover, the accuracy of camera trajectories is not a particular problem in our case, as we have GPS data for each camera position (DGPS, R100 from Hemisphere, Inc.). In brief, when the vehicle revisits the same location in a city multiple times, the same scene points can be reconstructed as different points. This will be appropriately considered in the method described later.

The SfM reconstruction is then registered with the map by finding a similarity transformation such that the transformed camera positions and their GPS data are as close to each other as possible using a least-squares measure [23, 24, 25]. As the SfM reconstruction inevitably experiences drifts, we divide the whole sequence into a number of subsequences whose length is about 100 images (or, equivalently, 200 meters in travel distance), for each of which SfM is performed. This also contributes to reducing the overall computational cost. We avoided overlooking buildings at both ends of each subsequence by dividing the original sequence such that the ends of neighboring subsequences overlap with each other. An example of SfM reconstruction after map registration is shown in Figure 5.

In our 2D city map, each building is represented as a polygon approximating its ground projection. It is impossible to recover the original 3D shapes of buildings. Thus, we assume that the polygons depict the outer walls of the first floor of the building and that these walls are at least five meters high. When comparing the SfM point cloud with the buildings, we use only points up to five meters high above ground level and ignore other points.

3.3. Details of the proposed method

3.3.1. Notation

Supposing the map contains K buildings, we denote the existence of the k-th building as a binary variable $b_k = \{0, 1\}$ $(k = 1, \cdots, K)$. The existence of all buildings is represented as a vector $\mathbf{b} = [b_1, \ldots, b_K]$ and the existence of all but the k-th building as $\mathbf{b}_{-k}$. The k-th building has J_k walls, and the existence of the j-th wall of the k-th building is represented by a binary variable $w_{kj} = \{0, 1\}$ $(j = 1, \cdots, J_k)$. We denote the camera pose (6 degrees of freedom) of the i-th camera by $\mathbf{c}_l$ $(l = 1, \cdots, L)$.

The view angle of the camera we used spanned the view of a building wall. In this evaluation, we take occlusion by other buildings (the blue triangles in Figure 4) into consideration. $\theta_{kj}^{(l)}$ represents the angle of the j-th wall of the k-th building viewed from the l-th camera pose. When the existence of all other buildings (i.e.,$\mathbf{b}_{-k}$) is known, it is

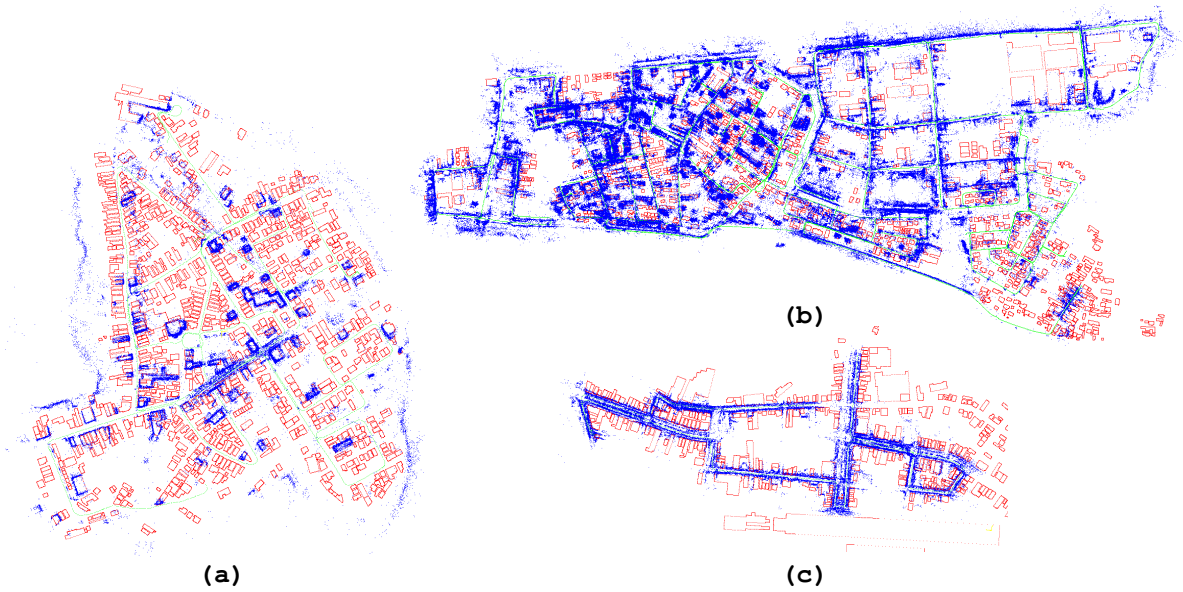


Figure 5. SfM reconstructions for the three cities tested in the experiments; best viewed in color. The reconstructed points and camera trajectories are shown as blue dots and green curves, respectively. The building polygons are shown as red lines. (a) Otsuchi town (reconstructed from 2870 images). (b) Miyagino ward (18002 images). (c) Kamaishi city (1266 images).

possible to evaluate the angle; thus, it may be written as $\theta_{kj}^{(l)}(\mathbf{b}_{-k})$. It is assumed here that each building has a closed shape, and we set $\theta_{kj}^{(l)} = 0$ if the j-th wall is viewed from the back side by the l-th camera.

3.3.2. Modeling number of points emerging from a wall

Individual buildings appear different in the images because they are captured from a vehicle traveling on the streets. For instance, the closer to a building the camera trajectory is, the larger the building will appear in the image, and vice versa. Moreover, a building can be occluded by other buildings depending on its position relative to the camera trajectory. Occlusion is illustrated in Figure 4. The larger a building appears in the images without occlusion, the higher the confidence of its existence should be, and vice versa. Therefore, the estimation of a building's existence must reflect the quantity and quality of the observations, such as occlusion, and the distance from the trajectory. We take this into account by modeling the emergence mechanism of the point cloud.

Each point in the point cloud originates from a feature point in an image. Hence, we begin by modeling the number of feature points that are extracted for a wall from a single image by assuming the model to be proportional to the wall area occupying the image. Although the number of feature points should depend on wall texture, we neglect this dependency by extracting feature points from images as uniformly as possible.

In the SfM process, these feature points are used to match sequential images. Successfully matched points form a trajectory of a scene point in the images and are subsequently used to compute the 3D coordinates of the feature points. Next, we consider modeling the frequency of the feature points that are successfully matched through the image sequence. However, the feature matching mechanism is too complicated to model precisely because it can fail for a variety of reasons. Therefore, we instead evaluate a lower bound of the number of points on a wall reconstructed from consecutive images.

Additionally, we consider a simplified ideal model of point cloud generation for a wall. In this model, the number of points is proportional to the largest wall area seen in the images sequence. It is assumed that new trajectories start before the peak in area and end after it (Figure 6). Because this model is ideal, it should give a lower bound. Indeed, the number of points is always on the higher side, as a point could be lost before the peak occurs and emerge again afterwards, always resulting in an increase in the number of point trajectories.

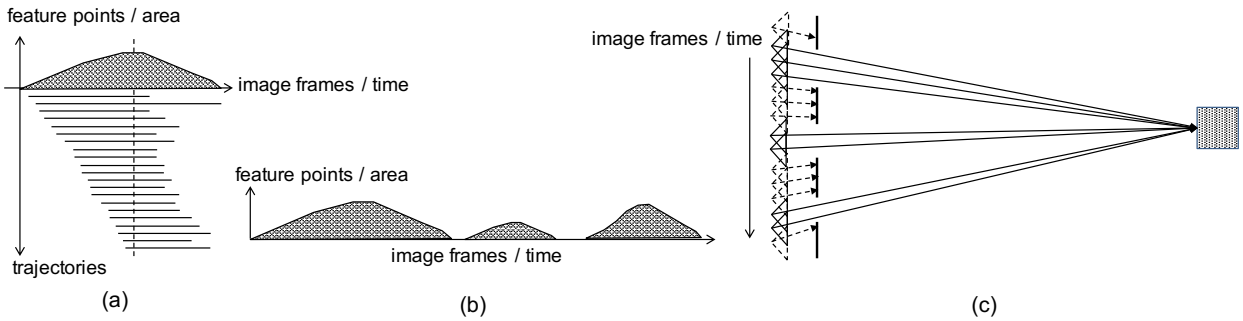


Figure 6. (a) A simplified model showing the emergence and disappearance of the image trajectories of points belonging to a wall in the image sequence. In this model, their number is proportional to the maximum image area of the wall. (b) A wall could be observed multiple times, which increases the number of points, due to occlusions or revisiting a same place, as shown in (c).

The wall area in an image is calculated as follows. We assume that the height of a wall is at least five meters because our city maps do not include height information, and the average height of a one-story houses in these cities is about five meters. We consider only the point cloud below this height. As the input to our system consists of cylindrical panoramic images, the area of the j-th wall of the k-th building observed from the l-th camera is given by

$$v_{kj}^{(l)} \propto \theta_{kj}^{(l)} \left(\tan^{-1} \frac{H}{D_{kj}^{(l)}} \right), \tag{1}$$

where $D_{kj}^{(l)}$ is the distance to the wall from this camera and $H = 5$ m. Note that the angle $\theta_{kj}^{(l)}$ depends on potential occlusion by other buildings, $\mathbf{b}_{-k}$, as does $v_{kj}^{(l)}$.

As discussed above, it is assumed that the maximum of $v_{kj}^{(l)}$ for $l = 1, 2, \ldots$ gives the lower bound of the number of points. The maximum is taken for the period beginning when the area begins with a non-zero value and ending with zero. Thus, as shown in Figure 6(b), if there are several such periods, it is necessary to sum over the largest areas for the periods. This summation reflects the nature of our SfM implementation. Once a scene point is lost during tracking, the same point is reconstructed as a different scene point when it is rediscovered. This summation is represented for the largest areas by dividing the entire sequence $S = [1, 2, \ldots, L]$ into subsequences with non-zero $v_{kj}^{(l)}$ for each (k, j) pair. Letting S_q $(q = 1, \ldots)$ be these subsequences, then the summation v_{kj} is given by

$$v_{kj} \equiv \sum_q \max_{l \in S_q} v_{kj}^{(l)}. \tag{2}$$

Note that v_{kj} depends on $\mathbf{b}_{-k}$.

3.3.3. Observation model of a wall

The estimation of the existence of a wall first requires a decision as to which points belong to the wall. A point is judged to belong to a wall if its orthogonal projection of the point lies inside the wall (the rectangle of the polygon line of the wall $\times$ 5 meters high) and if the distance from the point to the wall is less than a threshold (we used 2.5 meters in these experiments to accommodate for inaccurate 3D structures). Let q_{kj} be the number of such points for the j-th wall of the k-th building. We normalize q_{kj} with the estimate v_{kj} of the point cloud size as q_{kj}/v_{kj} to consider multiple counts of same scene points (Section 3.2) and use it to estimate whether the wall exists.

However, points can emerge from objects other than building walls, for example, from trees, electric wires, telegraph poles, and the remnants of destroyed buildings. Counting the number of points in a naïve manner could lead to erroneous estimation because of the proximity of such objects to building walls.

We overcame this difficulty by using an observation model of building walls that is based not only on the number of points, but also on the distribution of points. If a wall exists and generates points, the points should be distributed over most of the wall area in a 2D manner. As nonbuilding wall objects basically generate points only in a one-dimensional manner, this fact is useful for our purpose.

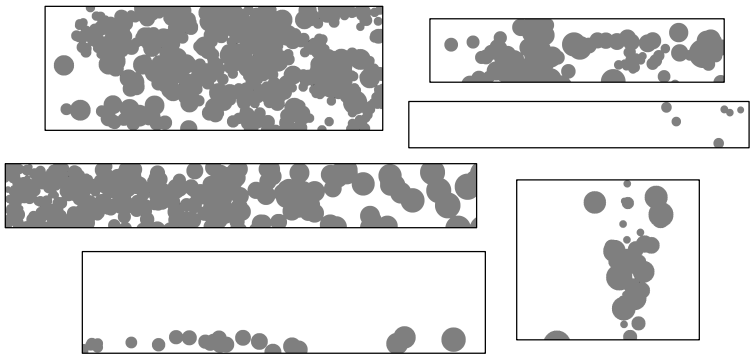


Figure 7. Definition of r_{kj}. Each rectangle represents a wall. For the point cloud emerging around the wall, each point is projected onto it, at which a filled circle is drawn on the wall. The radius of the circle is chosen to be proportional to the distance from the camera observing the point to the wall. r_{kj} is the ratio of the filled area to the entire wall area, which we use as an observation for the wall.

The following three situations exist in relation to the existence of walls and other objects: 1) a wall exists and there are no other objects nearby, 2) a wall does not exist and there either are or are not other objects nearby, and 3) both a wall and other objects nearby exist. A possible approach to the observation model is to assume that the point cloud obeys a uniform distribution, and to then perform some statistical test to assess the uniformity. However, this will only works for the first and second situations mentioned above, because the third situation complicates the point distribution and increases the complexity of the statistical test.

Thus, we choose a simpler approach. The q_{kj} points extracted as above are all projected onto the wall in an orthogonal manner. A filled circle is then drawn for each projected point, as shown in Figure 7. The ratio of the filled area a'_{kj} to the total wall area a_{kj} is calculated as

$$r_{kj} = a'_{kj}/a_{kj}. \tag{3}$$

We regard this as a wall observation. The radius of the circle depends on the distances between the cameras observing the point and the wall. More specifically, letting d be the minimum distance, we set the radius α for each point as

$$\alpha = c_r d, \tag{4}$$

where c_r is a constant, which we set to 0.033 radians throughout these experiments. The value α corresponds to the approximate average distance between pairs of the nearest feature points in images, where we assume their density to be identical and uniform. We neglect the effect of foreshortening here.

The existence of wall (k, j) is estimated from the observation r_{kj}, for which we model two densities $p(r_{kj}|w_{kj} = 1; v_{kj})$ and $p(r_{kj}|w_{kj} = 0; v_{kj})$. Note that v_{kj}, which can be calculated only when $\mathbf{b}_{-k}$ is given and thus should be treated as a random variable, is treated here as a parameter. Figure 8 shows an example of histograms created from city data using the existence of buildings as the ground truth. These histograms are created for those walls with high visibility, i.e., which have a large v_{kj} value. The density of r_{kj} for existing walls is considered to have a large volume and the density for nonexistent walls is considered to have a volume that approximates zero.

These two densities vary their distributions depending on the value of v_{kj}, and are functions of two variables. Their modeling requires a large amount of training data, which contains ground truths of the existence of buildings for a number of cities. However, it is costly to create ground truths of this nature because of their large scale. Therefore, this method employs the simplest model, in which each density changes only once depending on v_{kj} as

$$p(r_{kj}|w_{kj} = 1; v_{kj}) = \begin{cases} p_{exist}(r_{kj}) & v_{kj} \geq c_v, \\ p_{invis}(r_{kj}) & \text{otherwise.} \end{cases} \tag{5a}$$

$$p(r_{kj}|w_{kj} = 0; v_{kj}) = \begin{cases} p_{vanish}(r_{kj}) & v_{kj} \geq c_v, \\ p_{invis}(r_{kj}) & \text{otherwise.} \end{cases} \tag{5b}$$

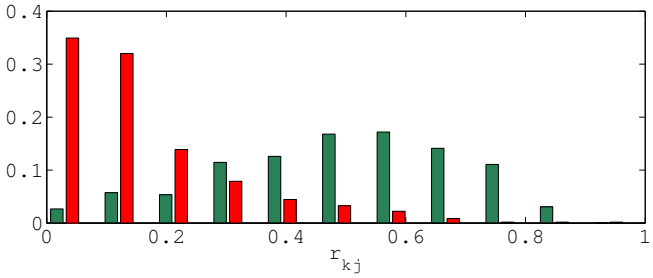


Figure 8. Example histograms of r_{kj} for existing walls (in green) and nonexistent walls (in red) for Otsuchi Town. These are created for the selected walls that have a large v_{kj} value.

where threshold c_v is set to 0.1 throughout the experiments represents the sufficiency of a wall observation, i.e., the j-th wall of the k-th building is regarded as being sufficiently observed if v_{kj} is higher than c_v, and vice versa. Assuming that a wall is observed once in the image sequence, $c_v = 0.1$ is roughly equivalent to that the wall occupies approximately 5% of the width of the cylindrical panoramic image at maximum. The green and the red histograms in Figure 8 are modeled by the two densities p_{exist} and p_{vanish}, respectively, for which we choose the following truncated Gaussian densities: $N(0.5, 0.03)/0.9961$ for p_{exist} and $N(0.0, 0.01)/0.5$ for p_{vanish}, where the denominators normalize the density.) The remaining term $p_{invis}(r_{kj})$ represents the density of low-visibility walls, in which case there is only a single density regardless of whether the wall exists. It is not necessary to explicitly model this density when estimating the existence of buildings, which is described in the next section.

3.3.4. Estimation of building existences

The existence of each building is estimated by evaluating posterior $p(b_k = 1|r_{k1}, \ldots, r_{kJ_k})$ of the k-th building by using all the observations r_{kj} associated with the building. Despite the existence of a complicated mutual dependence between the variables $r_{k1}, \ldots, r_{kJ_k}$, we introduce an assumption of naïve independence and approximate it as

$$\begin{aligned} p(b_k = 1|r_{k1}, \ldots, r_{kJ_k}) &\propto p(b_k = 1) \prod_{j=1}^{J_k} p(r_{kj}|b_k = 1) \\ &= p(b_k = 1) \prod_{j=1}^{J_k} p(r_{kj}|w_{kj} = 1). \end{aligned} \tag{6}$$

The second equality is valid because $b_k = 1$ means $w_{kj} = 1$ for any j and only w_{kj} among $w_{k1}, \ldots, w_{kJ_k}$ has a relation to r_{kj}. Assuming b_k does not have a prior value and setting $p(b_k = 1) = p(b_k = 0) = 1/2$, the probabilities becomes

$$p(b_k = 1|r_{k1}, \ldots, r_{kJ_k}) = \frac{\prod_{j=1}^{J_k} p(r_{kj}|w_{kj} = 1)}{\prod_{j=1}^{J_k} p(r_{kj}|w_{kj} = 1) + \prod_{j=1}^{J_k} p(r_{kj}|w_{kj} = 0)}. \tag{7}$$

Note that the posterior depends on $v_{k1}, \ldots, v_{kJ_k}$ as in $p(r_{kj}|\cdot)'s$, although they are omitted for simplicity.

3.3.5. Greedy iterative estimation

The probability of the existence of the k-th building is given by Eq. (7). However, a straightforward evaluation of this probability is impossible because of the mutual dependence of the existence of buildings on each other. The densities of r_{kj} on the right-hand side (e.g., $p(r_{kj}|w_{kj} = 1; v_{kj})$) depend on v_{kj}, which can be evaluated only when the existence $\mathbf{b}_{-k}$ of other buildings is given.

This complicated dependency is addressed by using our greedy iterative approach. First, all the buildings are assumed to exist ($b_k = 1$) and a binary decision is made iteratively ($b_k = 0/1$) for each individual building in order of decreasing confidence, which is given by $p(b_k|\cdot)$ in Eq. (7) (i.e., its proximity to either 0 or 1). Depending on the value determined for b_k for some buildings, v_{kj} is recalculated, which will change the values of $p(b_k|\cdot)$'s.

This iterative approach should work because the existence of a building alongside a street along which the vehicle travels can be correctly estimated independently of other buildings. Once the existence of this building is determined, the chances of correctly estimating the existence of buildings potentially occluded by this building are improved.

Algorithm 1 Estimating the existence of buildings.

Input: Points and camera trajectories generated by SfM and building polygons. **Output:** Existence values b_k for the buildings that are judged ($d_k = 1$) and existence probabilities determined using Eq. (7) for others.

1: Initialize $d_k = 0$ for $k = 1, \ldots, K$.
2: Compute r_{kj} for any possible pair (k, j) for $k = 1, \ldots, K$ and $j = 1, \ldots, J_k$.
3: **repeat**
4: Assuming that buildings with $d_k = 0$ and those with $d_k = 1$ and $b_k = 1$ exist,use Eq. (2). to evaluate v_{kj} for any pair (k, j).
5: Compute the probability of Eq. (7) for each buildings with $d_k = 0$ using the precomputed r_{kj} values and the latest v_{kj}'s.
6: Use Eq. (8) to set b_k and d_k.
7: **until** No new building is judged as existing.

Table 1. Recognition rates of the compared methods. The numbers in the leftmost column represent (the number of existing buildings)/(the total number of buildings). In each cell, the lower and upper numbers are the number ratio of the building given the judgment and its accuracy, respectively, both as a percentage (%). "Projected area ratio (v and r)" takes both the visibility of a wall and the area ratio of points projected to the wall into account. In contrast, "Number of points (q alone)" and "Normalized number of points (v and q)" consider just the number of feature points on a wall and the density per unit area, respectively.

	Projected area ratio (v and r)					Normalized number of points (v and q)					Number of points (q alone)				
c_m	0.0	0.1	0.2	0.3	0.4	0.0	0.1	0.2	0.3	0.4	0.0	0.1	0.2	0.3	0.4
Otsuchi	95.6	96.1	97.3	96.8	**97.5**	91.9	92.0	92.3	92.7	93.9	94.1	94.4	94.7	94.9	95.2
83/887	78.9	78.4	77.3	76.3	73.5	78.4	77.8	77.3	76.9	75.4	100.0	99.8	99.1	98.6	97.6
Miyagino	89.4	90.7	91.6	92.5	**94.1**	84.7	85.2	85.8	86.2	86.7	86.9	87.5	88.4	89.2	90.0
304/1088	80.0	78.0	76.3	76.2	69.1	78.5	77.8	76.7	75.6	74.1	100.0	98.9	97.3	95.9	94.3
Kamaishi	90.0	90.3	90.3	92.2	93.3	93.3	93.7	94.1	94.1	**94.1**	53.5	53.9	56.1	56.5	59.5
218/269	78.8	78.1	76.6	71.4	68.4	78.8	78.4	78.1	77.3	77.3	100.0	98.1	95.2	93.3	87.4

We introduce a binary variable $d_k = \{0, 1\}$ to indicate whether a decision has been made for the k-th building, i.e., $d_k = 1$ and $d_k = 0$ represents decided and undecided, respectively. The values of v_{kj} are recomputed at each iteration, and the occlusion is evaluated by using the existences b_k of the decided buildings with $d_k = 1$ and by assuming buildings with $d_k = 0$ to exist. In the first iteration, $d_k = 0$ for all the buildings, which results in the evaluation of v_{kj} assuming all buildings to exist, except for the k-th one. The buildings alongside a street on which the vehicle runs will appear large without occlusion in the images, and thus their facing wall(s) will have a large v_{kj} value.

Straightforward implementation of the above process requires considerable computational time because the value of v_{kj} is updated when the decision for every building is made. We therefore reduced the computational cost by incorporating two approximations. The first introduces a confidence threshold c_m and immediately decides the existence of all the buildings with confidence levels beyond this threshold, thereby significantly reducing the number of v_{kj} updates (or equivalently, the number of iterations). The decision is made for undecided buildings with $d_k = 0$ by calculating the probability of Eq. (7), which we denote by $p(b_k|\cdot)$ here, and set b_k and d_k as follows:

$$\begin{array}{llll} b_k \leftarrow 0 & \text{and} & d_k \leftarrow 1 & \text{if } p(b_k|\cdot) < 1/2 - c_m, \\ b_k \leftarrow 1 & \text{and} & d_k \leftarrow 1 & \text{if } p(b_k|\cdot) > 1/2 + c_m. \end{array} \tag{8}$$

Note that if $1/2 - c_m \leq p(b_k|\cdot) \leq 1/2 + c_m$ there is no need to do anything. Here, c_m is the control parameter for the aggressiveness of the decision and ranges from 0.0 to 0.4 in these experiments. The overall procedure is summarized in Algorithm 1.

3.4. Experimental results

We conducted experiments to validate the effectiveness of our method. We applied our method to an image archive captured along the northeastern coastline of Japan, from which we chose three cities that differ with respect to the amount of damage incurred: *Otsuchi Town, Miaygino Ward,* and *Kamaishi City*. Their SfM reconstructions on the maps are shown in Figure 5. SfM was performed by using 2870, 18002, and 1266 images, respectively.

The same problem setting as our study has not previously been studied. Hence, we implement two variants of the proposed method by omitting the main ingredients, and compare their performance to evaluate their effectiveness.

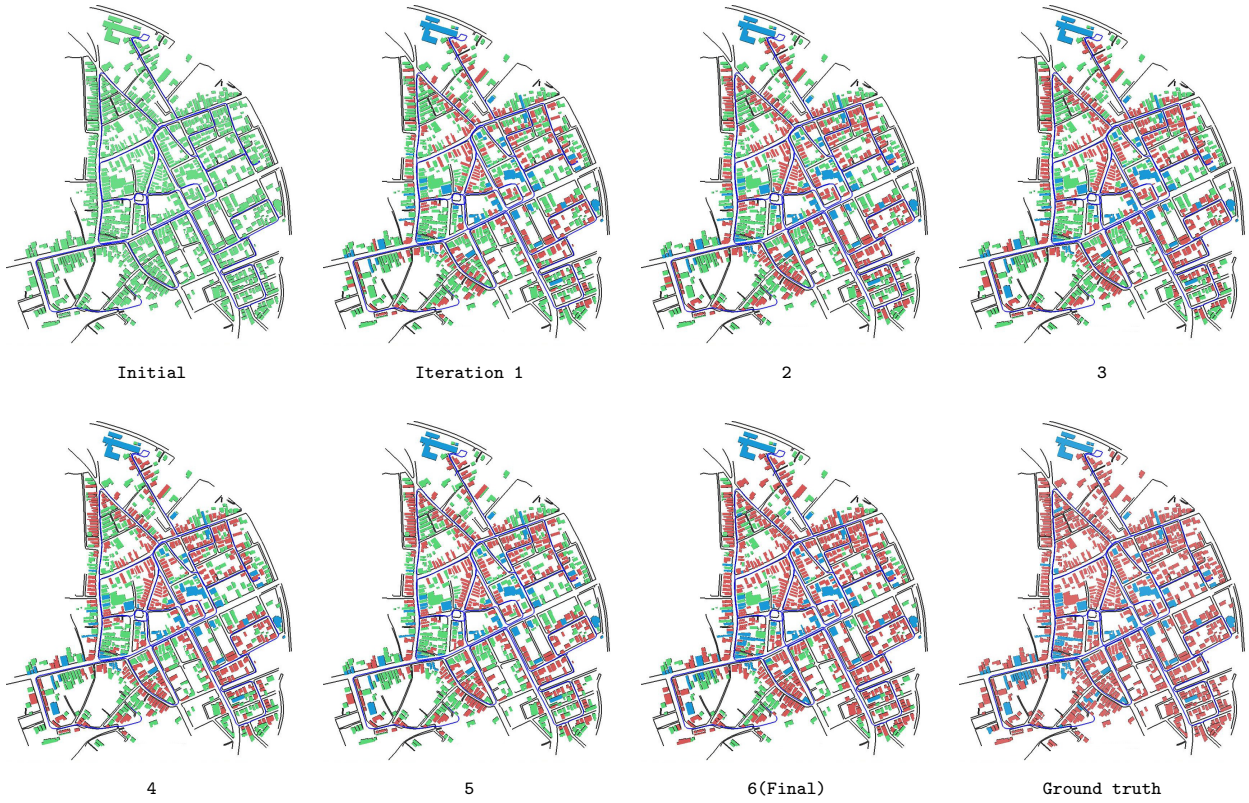


Figure 9. Intermediate results obtained for Otsuchi Town after different iteration counts (best viewed on a color monitor). Green, blue, and red polygons indicate buildings that have yet to be judged, those judged to exist, and those judged not to exist, respectively. The trajectory followed by the camera is displayed in dark blue.

Apart from the underlying idea of comparing an SfM point cloud with building walls, the method has the following three novel components: 1) v_{kj}, introduced in Section 3.3.2, is used to quantify the appearance of each wall in images, 2) r_{kj}, introduced in Section 3.3.3, is used to quantify the uniformity of points on each wall, 3) a greedy iterative algorithm overcomes the mutual dependency among the existence of buildings. The effectiveness of 1) and 2) are evaluated by considering two variants of the proposed method in which v_{kj} and r_{kj}, respectively, are not used. Specifically, in the first variant "Number of points (q alone)" which considers just the number of feature points on a wall, we simply set $c_v = 0$ in Eq. (5), which is evaluated in Eq. (7). In the second variant "Normalized number of points (v and q)", we replace r_{kj} with a simpler quantity q_{kj}/v_{kj}, which is first introduced in Section 3.3.3; the densities for the quantity are modeled as in Eq. (5) for r_{kj}. Then, the effectiveness of the greedy iterative algorithm is evaluated by varying margin c_m, which controls the number of buildings to be assessed in terms of each of the three methods.

The experimental results are listed in Table 1. The method, with all the components we proposed, its variant with q_{kj}/v_{kj} instead of r_{kj}, and its variant omitting v_{kj}, are denoted by "v and r," "v and q," and "q alone," respectively. The experimental results enabled us to make the following three observations: "v and r" achieves the best accuracy for two cities; "v and q" is the best for the third city; however, "v and r" is comparatively good, whereas "q alone" yields a very inaccurate result for the third city. The reason for the last observation may be attributable to the differences between the cities; namely a large number of buildings were demolished in the first two cities, whereas a relatively small number of buildings were demolished in the third city.

Figure 9 visualizes the decision as to whether buildings exist in each iteration. Green, blue, and red show the buildings that have not been judged, both existing and nonexistent, respectively. The images indicate that a certain number of buildings are judged as either existing or being nonexistent in each iteration. Additionally, the buildings that are distant from the camera trajectory (shown in blue curves) and/or are occluded by other buildings remain undecided until the end. These results show that the algorithm performs as we intended. Figures 10 and 11 show more

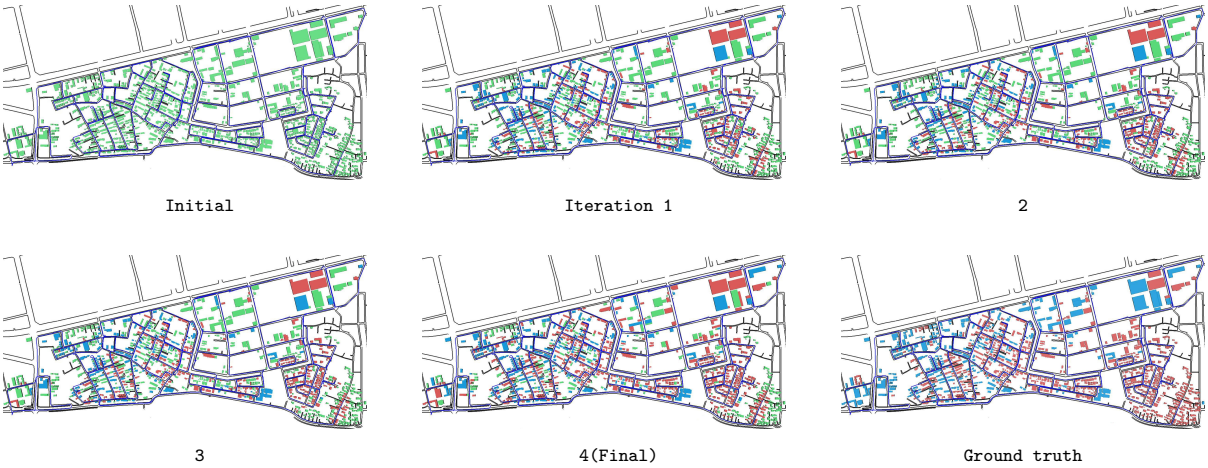


Figure 10. Intermediate results at different iterations for Miyagino Ward.

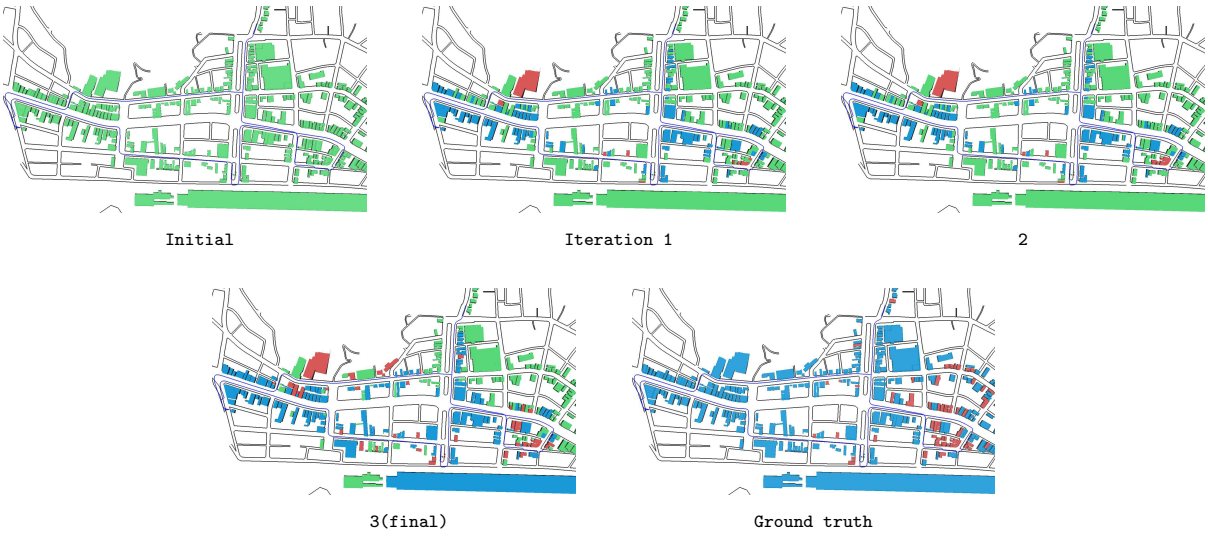


Figure 11. Intermediate results at different iterations for Kamaishi City.

results for the other cities.

The accuracy of the method did not reach 100% and the results contain several errors. There are two types of situations that are responsible for failure, although the boundary between them is somewhat obscure. The first situation results from a limitation of the method, whereas the other situation arises from the fundamental difficulty of labeling each building as existing or nonexistent.

The former situation occurs mostly when the assumptions of the method are violated. Figure 12(a) shows examples of when this occurs. In this scenario, the captured image is backlit, which leads to difficulties with the extracting feature points, thereby resulting in failure. In the same image, there is a building that has lost its walls, possibly due to the tsunami or a fire. This leads to the same difficulty, which is also the case when walls do not have sufficient textures.

There are some buildings for which even humans would find it difficult to judge their existence/nonexistence, and this could result in failure situations of the second type. Figure 12 (b) shows an example, where only foundations of buildings remain after the tsunami. Figure 12 (c) shows an example of a partially demolished building. In Fig.12 (d), the reconstructed point cloud is overlaid on its original building shape of Fig. 12 (c). Due to the nature of the disaster,

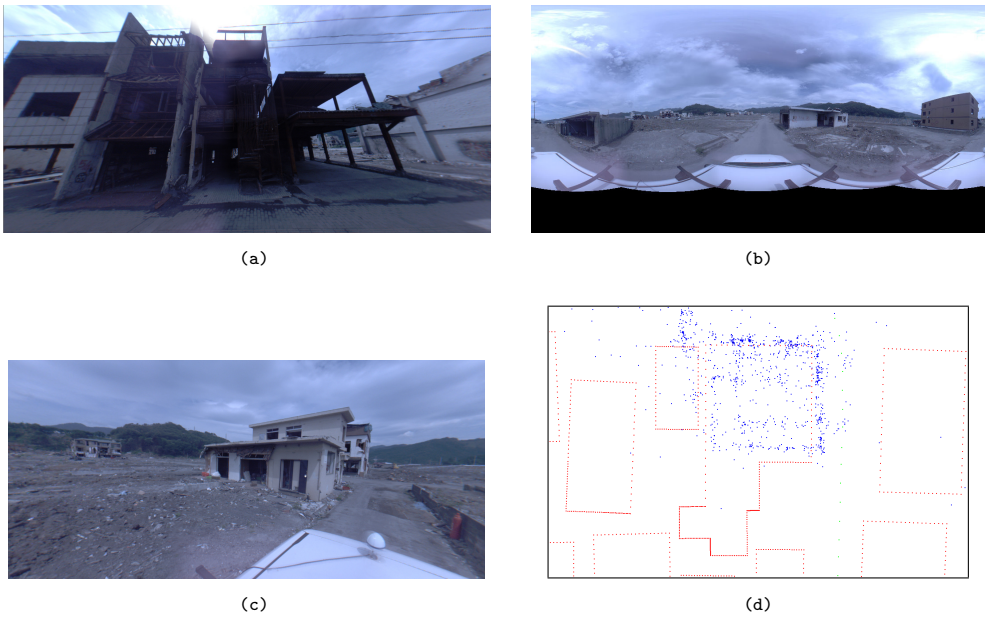


Figure 12. Examples of failure cases. (a) Presence of backlighting and building structures without walls. (b) The only remaining structure is the building foundation. (c) A building that is partially demolished. (d) Reconstructed point cloud overlaid on the original building shape in (c).

there are many such buildings in the cities considered in our experiments. Whether they should be labeled as existing or nonexistent will depend on the application. For our experiments, we labeled all of these buildings as existing in the ground truths.

As shown in Fig.2, we applied the proposed method to image sequences that are captured at different times after the disaster. The results demonstrate that our method is capable of visualizing structural changes in a city over time.

4. Scene change detection from a street image pair using CNN features and superpixel segmentation

This section describes the method for scene change detection, which uses as input a pair of omnidirectional images captured from a vehicle running on a street at different points in time. There are several challenges in estimating scene changes using vehicular imagery owing to differences in viewpoints, illumination conditions, photographic conditions, an the environmental conditions (e.g., clouds in the sky or dust on the road) between images taken at different time points. An example of such image pairs is shown in Figures 1 (a) and (c), in which the illumination and photographic conditions are different. Furthermore, visual difference in camera viewpoints inevitably exists, even though the images were captured from a vehicle running on the same street and were matched using GPS data, because of differences in vehicle paths and shutter timing. The target of scene changes in this study includes 3D (e.g., vanishing/emergence of buildings, and cars) as well as 2D changes (e.g., changes in textures on building walls). To precisely detect these types of changes using such an image pair, it is necessary to overcome these unwanted visual differences.

To overcome these issues, some previous approaches consider the problem in the 3D domain. These research studies assume that the 3D model of a scene is available or can be reconstructed from images, and that it is possible to register the input image to the model with pixel-level accuracy [14, 16, 15]. However, a 3D model is not always available for every city, especially for the past. Moreover, sometimes, there are insufficient visual features in a scene to precisely register the images. This is the case for the actual tsunami-damaged areas in Japan, where the scene drastically changed. Working in the 3D domain tends to be computationally costly, which can be another difficulty in change detection for a large city.

Therefore, we tackle 2D change detection; our approach is based on the direct comparison of an image pair. However, in the 2D domain, the unwanted visual differences mentioned above (i.e., viewpoint differences etc.) complicate image comparison. We therefore employ the features extracted by CNNs. To be more specific, we use a fully trained

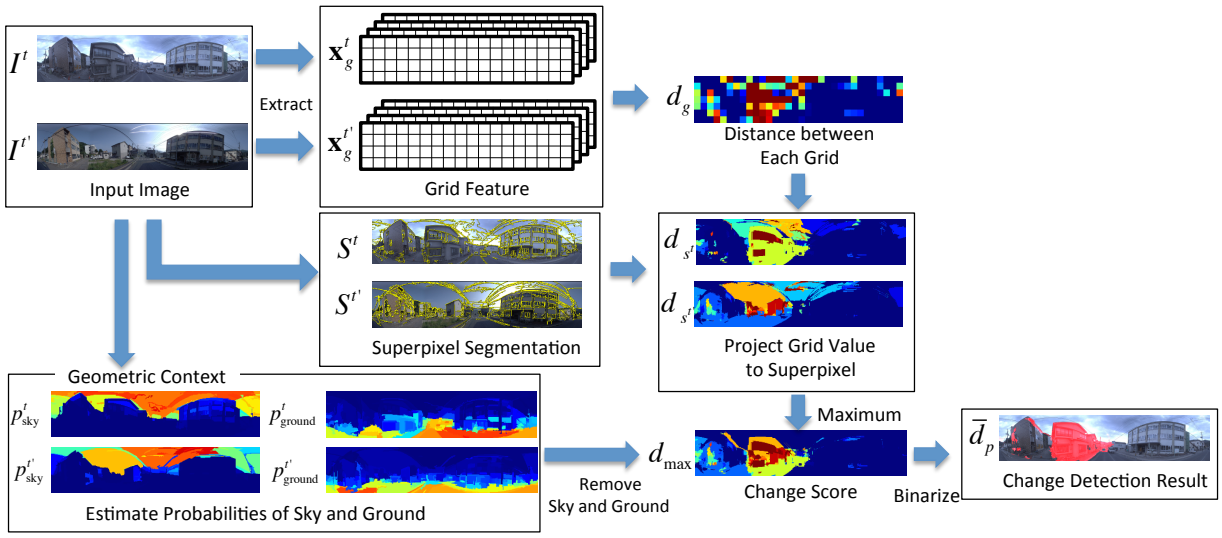


Figure 13. Flowchart of the proposed change detection method.

CNN for large-scale object recognition tasks [26] in a transfer learning setting. It has been reported in the literature that the activation of the upper layers of a CNN trained for a specific task can be reused for other visual classification tasks. Several recent works imply that the upper layers of CNNs represent and encode highly-abstract information about the input image [27, 28, 29]. We conjecture that highly abstract (or object-level) changes can be detected by using the upper layers, whereas low-level visual changes (e.g., edges and textures) are be detected using the lower layers. We show that this conjecture is true through several experimental results.

As shown below, CNN features indeed can accurately detect the scene changes even if there are the unwanted visual changes mentioned above. However, CNN features, by their nature, cannot provide precise segmentation boundaries for the changes. Thus, our method integrates CNN features with the superpixel segmentation of the input images. To be specific, the method first obtains coarse grid features of input images from the upper layer of a CNN and estimates the likelihood of scene changes at each grid cell by comparing the CNN features of the grid cell at different times. Next, the method projects the change likelihood of each grid cell into superpixel segments to obtain precise change boundaries. Figure 13 outlines the proposed method.

The main application scenario of the method is as follows. A vehicle on whose roof is mounted an omnidirectional camera and a GPS receiver drives on every street of a city twice at some time apart. This yields two sets of a large number of omnidirectional street images with the camera positions. Then, the two images from each of the two sets taken at the closest viewpoints are paired using the GPS data. Each image pair is input to detect changes for the pair, resulting in the estimation of changes over the entire city. To evaluate this method, we created a dataset called the *Panoramic Change Detection Dataset*, which is publicly available for the performance evaluation of change detection methods in this scenario.

4.1. Change detection using grid features

The flowchart of this method is shown in Figure 13. The method consists of the three components: i) extraction of grid features, ii) superpixel segmentation, and iii) estimation of sky and ground areas by geometric context [30]. These are described below.

(i) Extraction of grid features. We denote two input images as I^t and $I^{t'}$, where t and t' are the times at which the images were captured. Features of I^t and $I^{t'}$ are extracted from each grid cell $g(= 1, ..., N_g)$, yielding $\mathbf{x}_g{}^t$ and $\mathbf{x}_g{}^{t'}$. Our change detection targets are object-level changes (e.g, the emergence/vanishing of cars and buildings), and not low-level appearance changes owing to differences in viewpoints, illumination, or photography conditions. To distinguish these two, the method uses the activation of an upper layer of a deep CNN for the grid features $\mathbf{x}_g{}^t$ and $\mathbf{x}_g{}^{t'}$. Specifically, we use a pooling layer of the CNN. Each feature (e.g., $\mathbf{x}_g{}^t$) is the activation of all the units in the

same location across the maps of the pooling layer. Thus, ${\mathbf{x}_g}^t$ has the same number of elements as the maps of the pooling layer. Section 4.2 describes this in greater detail.

Next, we normalize these features, so that $\|{\mathbf{x}_g}^t\|_2 = 1$ in order to calculate their dissimilarity at each grid cell g as

$$d_g = \|{\mathbf{x}_g}^t - {\mathbf{x}_g}^{t'}\|_2. \tag{9}$$

Then the dissimilarity at each pixel $d_p (p = 1, ..., N_p)$ is determined by projecting dissimilarity d_g onto input images I^t, where N_p is the number of pixels. We simply set $d_p = d_g$ for any pixel p that is contained in the grid cell g.

(ii) Superpixel segmentation. Viewpoint differences are the major source of difficulty in change detection in the 2D domain. We use CNN features, expecting them to mitigate this difficulty because they are invariant to geometric transformations such as translation, 3D rotation, and even more complicated transformation. However, the resolution of the dissimilarity map d_p's is basically very low because of its pooling layers, which help to provide the invariances. To precisely estimate the change boundaries, we use superpixel segmentation and refine the dissimilarity map.

We start by computing the superpixel segmentation of I^t and $I^{t'}$. Let s^t be a superpixel and S^t be the set of superpixels. We define the dissimilarity d_{s^t} at a superpixel $s^t \in S^t$ to be the average of all pixels in s^t as

$$d_{s^t} = \frac{1}{|s^t|} \sum_{p \in s^t} d_p. \tag{10}$$

We denote the maximum value of d_{s^t} and $d_{s^{t'}}$ by $d_{\max}$, i.e., $d_{\max} = \max(d_{s^t}, d_{s^{t'}})$.

(iii) Estimation of sky and ground areas by geometric context. The street images we use contain the sky and the ground, which could have unwanted visual differences. To remove these segments from the change estimation, in the last step, the proposed method performs geometric context [30] segmentation method, which is known to be robust to changes in illumination and photography conditions. Geometric context estimates the probabilities that each pixel belongs to the sky and the ground $({p_{\text{sky}}}^t, {p_{\text{ground}}}^t)$ in input image I^t. Using these probabilities, the sky and ground areas are removed from the change estimation, converting the dissimilarity at each pixel into $\overline{d_p}$ as

$$\overline{d_p} = \begin{cases} 0 & (({p_{\text{sky}}}^t > a) \wedge ({p_{\text{sky}}}^{t'} > a)) \\ & \vee (({p_{\text{ground}}}^t > b) \wedge ({p_{\text{ground}}}^{t'} > b))), \\ d_{\max} & (\text{otherwise}) \end{cases} \tag{11}$$

where $a = t_{\text{sky}}$ and $b = t_{\text{ground}}$ are constant values within the range $0 \leq t_{\text{sky}}, t_{\text{ground}} \leq 1$.

4.2. CNN layer activation as grid features

CNNs perform object recognition tasks with high accuracy because they are sensitive to highly abstract semantic differences in images. CNNs are robust (i.e., invariant) to differences in viewpoints and illumination conditions. This property is key to the success of CNNs for object category recognition, which is the rationale for us to employing CNN features in our research. We expect that if a scene has not changed, then its image features should not change, even when the viewpoints or illumination conditions are slightly different between image acquisitions, and vice versa.

We use the VGG net, which is a state-of-the-art CNN for image recognition, proposed by Simonyan and Zisserman [31]. The VGG net has 16 layers, from which we choose one of its five pooling layers as a feature extractor. The spatial resolution does not change before and after each convolution layer since the stride of the convolutional layers is 1. As mentioned above, each grid feature is normalized so that its vector length is 1. Furthermore, the feature vector will have nonnegative values owing to the rectified linear units (ReLUs) [26] of the CNN. Therefore, each element d_i of dissimilarity $\mathbf{d}_g$ is within the range $[0, \sqrt{2}]$.

We tested two models of this CNN trained for two different tasks: large-scale object recognition for ImageNet Large-Scale Visual Recognition Challenges (ILSVRC) and a scene classification using the SUN dataset [32]. Specifically, starting from the above fully-trained model for ILSVRC, we retrained it using the SUN dataset. The idea is that we may expect an improvement in performance by tuning the CNN to more relevant tasks for our purpose.

Any feature may be used for the grid feature instead of CNN features. In our experiments, the CNN feature is compared to Dense-SIFT [33] and local patch features (raw pixel brightnesses). It has been reported that CNNs

perform well for recognition tasks; Bag-of-Visual Words (BoVW) and Fisher vectors, which encode the population of local features such as SIFT, are former state-of-the-art approaches [34, 35, 36, 37, 33, 38].

4.3. Experimental results

4.3.1. Panoramic change detection dataset

To evaluate this method, we created the *Panoramic Change Detection Dataset* [1], which consists of two subsets, TSUNAMI and GSV, which contain 100 panoramic image pairs of scenes in tsunami-damaged areas of Japan, and Google Street View, respectively. The image pairs consists of a query image and an image created at a different time, which is the closest to the query image in 3D space. The size of these images is 224×1024 pixels.

We manually labeled the ground truth of scene changes at each pixel. If a change has occurred at a scene point corresponding to a pixel in the paired images, the binary value is true, and vice versa. We defined the scene changes to be detected as 2D changes of object surfaces (e.g., updates of a billboard) and 3D, structural changes (e.g., emergence/vanishing of cars and buildings). Changes resulting from the following differences are excluded: illumination and photographic conditions (e.g., specular reflections on building windows), and the sky and ground (e.g., clouds and signs on the road surface). It is difficult for even human vision to judge the existence of changes owing to differences in viewpoint and illumination. In fact, it took 20 minutes on average for an annotator to create the ground truth map for an image pair, demonstrating the necessity of a method for detecting scene changes automatically and accurately.

4.3.2. Detailed experimental configuration

The method is based on several parameters: (1) threshold t_{dist} to judge whether the scene in each grid changes or not, (2) thresholds to detect the sky and ground using geometric context (t_{sky}, t_{ground}), and (3) the parameters for superpixel segmentation.

Table 2 shows the thresholds t_{dist} determined by fivefold cross-validation using the change detection dataset. In the case of the pooling-layer features and Dense-SIFT, $d_i \in \mathbf{d}_g$ ranges from $0 \leq d_i \leq \sqrt{2}$ because all elements of the features are nonnegative values. In the case of a grayscale local-patch, d_i ranges from $0 \leq d_i \leq 2$. The thresholds for pools 3, 4, and 5 and the grayscale local-patch are approximately the median values of their ranges. The thresholds for the geometric context are fixed for all the experiments, with $t_{\mathrm{sky}} = 0.2$ and $t_{\mathrm{ground}} = 0.8$.

To perform superpixel segmentation, we used the efficient graph-based image segmentation method proposed by Felsenszwalb [39]. The parameters of the superpixel segmentation (scale, diameter of the Gaussian kernel, and minimum component size) are fixed throughout the experiments. We used two CNN models that with the same structure as in Simonyan-Zisserman [31], trained for object category recognition (ILSVRC) and scene classification (SUN).

To evaluate the effectiveness of the pooling layer as a grid feature, we use Dense-SIFT [33] and a grayscale local patch. Since features can be independently generated for each grid cell, we set the grid size to be the same as that of the pool-5 layer. In the case of Dense-SIFT, we extract descriptors at four different scales whose basic size is the grid size, and concatenate them to form a feature vector. That is, the dimensions of Dense-SIFT at each grid cell is 512 (128×4). As a local patch feature, we use the set of the raw grayscale pixels within a grid cell. Each grid cell is resized to 16×16 pixels, generating a feature vector of of 256 elements.

4.3.3. Comparison of results

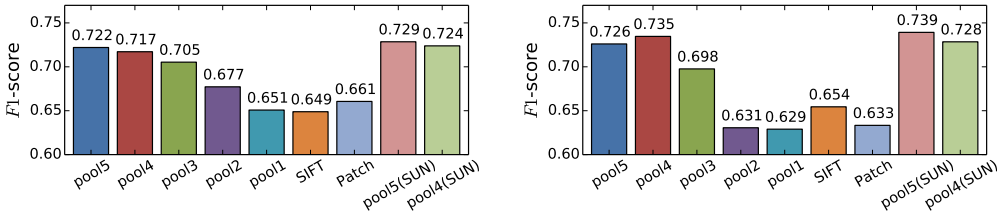
The F_1 scores for all features are shown in Figure 14, which shows that the features of pools 4 and 5 result in the best F_1 scores. Retraining of the CNN for a scene recognition task does not improve the results much. Additionally, upper pooling layers perform better than lower pooling layers. F_1 scores for pool 1 are almost the same as those of the baseline methods (Dense-SIFT and grayscale local-patch). These results validate the use of the CNN feature for our change detection problem.

The detection accuracy of GSV is worse than that of TSUNAMI. We believe that there are two major reasons for the low accuracy. One is that viewpoint differences of some image pairs are too large to be handled by the grid size

[1] The data used in this study (the pairs of the omnidirectional panoramic images taken at different time points and the manually labeled ground truth of change detection) are available from our web site: http://www.vision.is.tohoku.ac.jp/us/download/

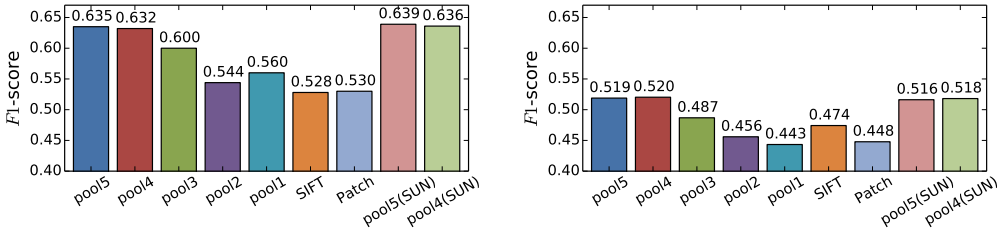
Table 2. Thresholds t_{dist} determined by fivefold cross-validation using panoramic change detection datasets TSUNAMI and GSV.

	pool5	pool4	pool3	pool2	pool1	Dense SIFT	Patch	pool5 (SUN)	pool4 (SUN)
TSUNAMI	0.75	0.75	0.71	0.64	0.35	0.24	0.83	0.76	0.86
GSV	0.75	0.78	0.72	0.65	0.58	0.24	0.92	0.76	0.88



(a) With superpixel segmentation (b) Without superpixel segmentation

TSUNAMI



(a) With superpixel segmentation (b) Without superpixel segmentation

GSV

Figure 14. F_1 scores of change detection by various methods (average of 100 images in TSUNAMI and GSV datasets). (a) and (b) show the results with and without superpixel segmentation in the projection step (Section 4.1), respectively. "pool-x" indicates the features extracted from the corresponding pooling layer of the CNN trained for the ILSVRC object recognition. "pool-x(SUN)" indicates similar features from the CNN trained for a scene classification task using the SUN database.

we used. The other reason is that, conversely, most of the scene changes are too small to be dealt with by the grid size used. These two are contradictory and it is not easy to resolve this contradiction immediately.

The differences between the results in Figure 14 (a) and (b) show the effectiveness of superpixel segmentation in each dataset. Superpixel segmentation did not work as well as we expected for some objects, for example, debris and damaged buildings. Therefore, superpixel segmentation did not improve change detection accuracy in the TSUNAMI dataset. On the other hand, in the GSV dataset, the F_1 scores of the results with superpixel segmentation are much higher than those without. Additionally, superpixel segmentation can compensate for the coarseness of the grid size.

Figures 15 and 16 show examples of the change detection results. From these results, it can be observed that the proposed method can correctly detect the scene changes (e.g., demolished and new buildings, cars and debris). In some cases, geometric context could not accurately estimate the sky owing to electrical wires and poles, or could not distinguish between the ground and low-height objects (e.g., debris and cars). The method was nevertheless able to detect object-level scene changes fairly accurately. Figure 17 shows additional scene results.

As shown in Figure 18, we also compared the results of different CNN pooling layers as grid features. Note that the grid size for each result is determined by the number of units in a chosen pooling layer. The results show that the feature of the upper pooling layer discriminates highly abstract, object-level differences in the scene, whereas that of the lower pooling layer detects the differences between low-level visual features (e.g., edges). This tendency confirms our conjecture described earlier. Furthermore, it implies that we can improve the estimation accuracy by adaptively

choosing layers depending on the abstraction level of interest.

5. Conclusion

Our paper describes two novel methods for estimating temporal changes in a city using street-level images, which can be used in a complementary fashion depending on the type of available data and target changes. The first method estimates the existence of buildings based on street-level images and a 2D city map. A 3D point cloud of the city structure is reconstructed using SfM, which matches the city structure with the 3D structures of buildings recovered from the 2D maps. In our research, we overcame various difficulties presented by this approach by developing a model of the point cloud generation mechanism. In addition, our method also enables us to observe each building wall and uses a greedy iterative approach to evaluate the probability that each building exists. The second method uses a pair of street-level images to detect scene changes. This method overcomes differences, such as in the viewpoint and illumination, between images acquired at different times using the activation of the upper layer of a CNN, which is expected to be invariant to changes in appearance caused by such differences. The spatial resolution that is lost by the pooling operations of the CNN is compensated for by integrating the CNN features with superpixel segmentation of the scene images. We evaluated the performance of our methods by applying them to tsunami-damaged areas, and the experimental results confirm the effectiveness of our approaches. A dataset, the Panoramic Change Detection Dataset, was created to the evaluate scene change detection and is publicly available from our web site. In the future, we plan to further improve the estimation accuracy of this method, for example, by using activations of multiple layers in scene change detection, and by integrating the two methods in cases for which both street-level images and a 2D city map are available.

Acknowledgement

This research is supported by the ImPACT Program of the Council for Science, Technology, and Innovation (Cabinet Office, Government of Japan).

References

[1] S. Agarwal, N. Snavely, I. Simon, S. M. Seitz, R. Szeliski, Building Rome in a day, in: ICCV, 2009, pp. 72–79.
[2] D. Crandall, A. Owens, N. Snavely, D. Huttenlocher, Discrete-Continuous Optimization for Large-Scale Structure from Motion, in: CVPR, 2011, pp. 3001–3008.
[3] M. Pollefeys, D. Nistér, J.-M. Frahm, A. Akbarzadeh, P. Mordohai, B. Clipp, C. Engels, D. Gallup, S.-J. Kim, P. Merrell, C. Salmi, S. Sinha, B. Talton, L. Wang, Q. Yang, H. Stewénius, R. Yang, G. Welch, H. Towles, Detailed Real-Time Urban 3D Reconstruction from Video, IJCV 78 (2-3) (2008) 143–167.
[4] N. Snavely, S. M. Seitz, R. Szeliski, Modeling the World from Internet Photo Collections, IJCV 80 (2) (2007) 189–210.
[5] C. Zhang, L. Wang, R. Yang, Semantic Segmentation of Urban Scenes Using Dense Depth Maps, in: ECCV, 2010, pp. 708–721.
[6] G. Zhang, J. Jia, W. Xiong, T.-T. Wong, P.-A. Heng, H. Bao, Moving Object Extraction with a Hand-held Camera, in: ICCV, 2007, pp. 1–8.
[7] T. Pollard, J. L. Mundy, Change Detection in a 3-d World, in: CVPR, 2007, pp. 1–6.
[8] D. Crispell, J. Mundy, G. Taubin, A Variable-Resolution Probabilistic Three-Dimensional Model for Change Detection, Geoscience and Remote Sensing 50 (2) (2012) 489–500.
[9] D. C. Ibrahim Eden, Using 3D Line Segments for Robust and Efficient Change Detection from Multiple Noisy Images, in: ECCV, 2008, pp. 172–185.
[10] J. Košecka, Detecting Changes in Images of Street Scenes, in: ACCV, Springer, 2012, pp. 590–601.
[11] R. J. Radke, S. Andra, O. Al-Kofahi, B. Roysam, Image Change Detection Algorithms: A Systematic Survey, Transactions on Image Processing 14 (3) (2005) 294–307.
[12] A. Huertas, R. Nevatia, Detecting Changes in Aerial Views of Man-Made Structures, in: ICCV, 1998, pp. 73–80.
[13] A. Taneja, L. Ballan, M. Pollefeys, Image Based Detection of Geometric Changes in Urban Environments, in: ICCV, 2011, pp. 2336–2343.
[14] G. Schindler, F. Dellaert, Probabilistic Temporal Inference on Reconstructed 3D Scenes, in: CVPR, 2010, pp. 1410–1417.
[15] K. Matzen, N. Snavely, Scene Chronology, in: Proc. European Conf. on Computer Vision, 2014.
[16] A. Taneja, L. Ballan, M. Pollefeys, City-Scale Change Detection in Cadastral 3D Models Using Images, in: CVPR, 2013, pp. 113–120.
[17] P. Alcantarilla, S. Stent, G. Ros, R. Arroyo, R. Gherardi, Street-View Change Detection with Deconvolutional Networks, in: Robotics: Science and Systems (RSS), 2016.
[18] K. Sakurada, T. Okatani, K. M. Kitani, Hybrid macro–micro visual analysis for city-scale state estimation, CVIU 146 (2016) 86–98.
[19] B. Triggs, P. McLauchlan, R. Hartley, A. Fitzgibbon, Bundle Adjustment - Modern Synthesis, in: ICCV, 1999, pp. 298–372.
[20] D. Nistér, An Efficient Solution to the Five-Point Relative Pose Problem, Pattern Analysis and Machine Intelligence, IEEE Transactions on 26 (6) (2004) 756–770.
[21] R. Hartley, A. Zisserman, Multiple View Geometry in Computer Vision Second Edition, Cambridge University Press, 2004.

[22] H. Bay, T. Tuytelaars, L. Van Gool, Surf: Speeded Up Robust Features, in: ECCV, Springer, 2006, pp. 404–417.
[23] R. Kaminsky, N. Snavely, S. Seitz, R. Szeliski, Alignment of 3D Point Clouds to Overhead Images, in: CVPR Workshops, 2009, pp. 63–70.
[24] Q. Shan, C. Wu, B. Curless, Y. Furukawa, C. Hernandez, S. M. Seitz, Accurate Geo-Registration by Ground-to-Aerial Image Matching, in: 3D Vision, 2015, pp. 525–532.
[25] B. Klingner, D. Martin, J. Roseborough, Street View Motion-from-Structure-from-Motion, in: ICCV, 2013, pp. 953–960.
[26] A. Krizhevsky, I. Sutskever, G. E. Hinton, ImageNet Classification with Deep Convolutional Neural Networks, in: Advances in Neural Information Processing Systems 25, Curran Associates, Inc., 2012, pp. 1097–1105.
[27] Y. Bengio, Learning deep architectures for ai, Foundations and Trends® in Machine Learning 2 (1) (2009) 1–127.
[28] M. D. Zeiler, R. Fergus, Visualizing and Understanding Convolutional Networks, in: ECCV, Springer, 2014, pp. 818–833.
[29] A. Babenko, A. Slesarev, A. Chigorin, V. Lempitsky, Neural Codes for Image Retrieval, in: ECCV, Springer, 2014, pp. 584–599.
[30] D. Hoiem, A. Efros, M. Hebert, Geometric Context from a Single Image, in: ICCV, 2005, pp. 654–661.
[31] K. Simonyan, A. Zisserman, Very Deep Convolutional Networks for Large-Scale Image Recognition, CoRR.
[32] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, A. Oliva, Learning Deep Features for Scene Recognition using Places Database., NIPS.
[33] C. Liu, J. Yuen, A. Torralba, J. Sivic, W. T. Freeman, SIFT Flow: Dense Correspondence across Scenes and its Applications, in: ECCV, Springer, 2008, pp. 28–42.
[34] D. G. Lowe, Distinctive Image Features from Scale-Invariant Keypoints, IJCV 60 (2) (2004) 91–110.
[35] A. Vedaldi, B. Fulkerson, Vlfeat: An Open and Portable Library of Computer Vision Algorithms, in: International Conference on Multimedia, MM '10, ACM, 2010, pp. 1469–1472.
[36] J. Yang, Y.-G. Jiang, A. G. Hauptmann, C.-W. Ngo, Evaluating Bag-of-visual-words Representations in Scene Classification, in: International Workshop on Workshop on Multimedia Information Retrieval, ACM, 2007, pp. 197–206.
[37] F. Perronnin, C. Dance, Fisher Kernels on Visual Vocabularies for Image Categorization, in: CVPR, IEEE, 2007, pp. 1–8.
[38] F. Perronnin, J. Sánchez, T. Mensink, Improving the Fisher Kernel for Large-Scale Image Classification, in: ECCV, Springer, 2010, pp. 143–156.
[39] P. F. Felzenszwalb, D. P. Huttenlocher, Efficient graph-based image segmentation, International Journal of Computer Vision 59 (2) (2004) 167–181.

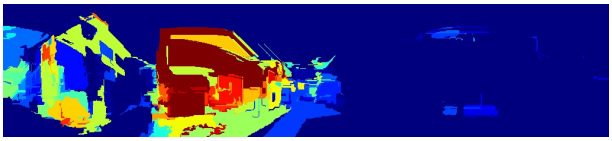

Input image (query)

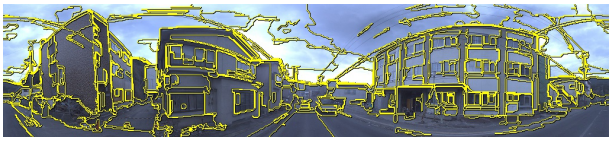

Input image (database)

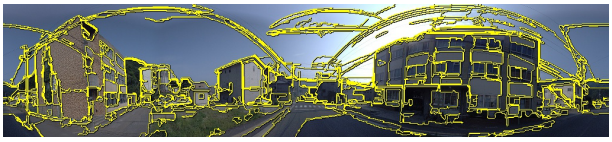

Ground-truth (superimposed)

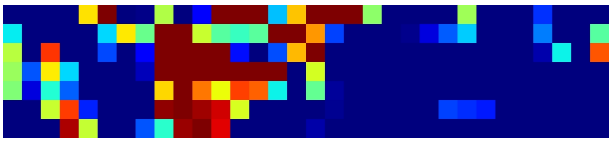

Ground-truth (mask)

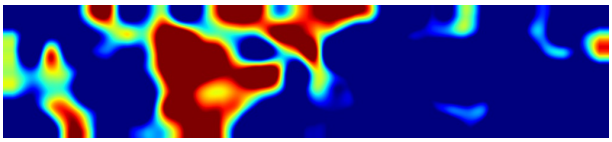

Change estimation (binarized)

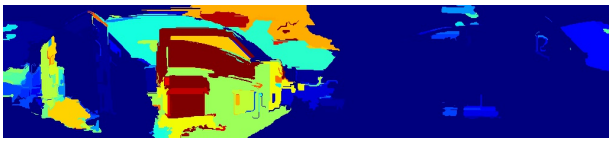

Change estimation (distance)

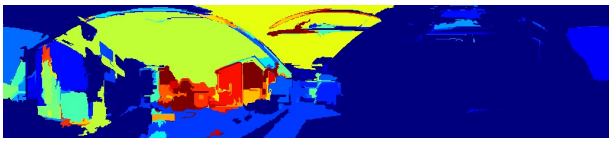

Superpixel segmentation (query)

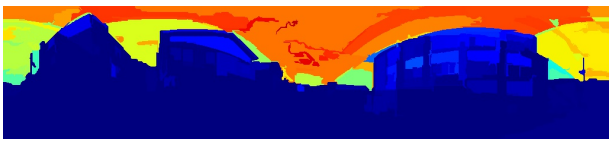

Superpixel segmentation (database)

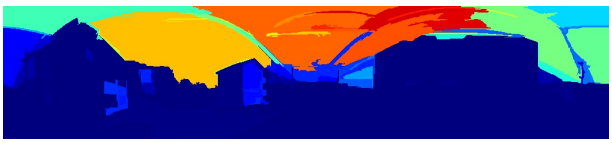

Feature distance (each grid)

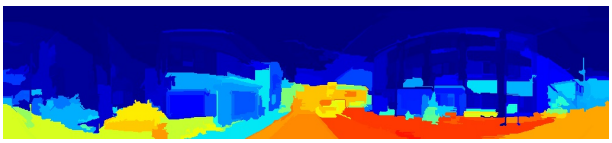

Feature distance (interpolation)


Feature distance in superpixel (query)


Feature distance in superpixel (database)


Sky probabilities (query)


Sky probabilities (database)


Ground probabilities (query)

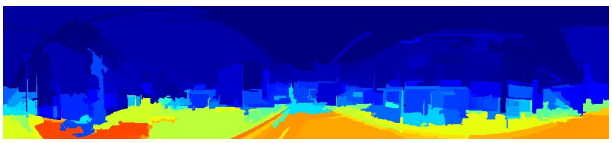

Ground probabilities (database)

Figure 15. Results of change detection using the CNN pool-5 feature (Frame No. 1/100 in the TSUNAMI change detection dataset)

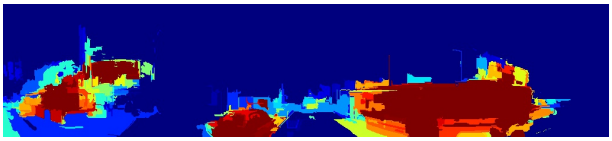

Input image (query)


Input image (database)

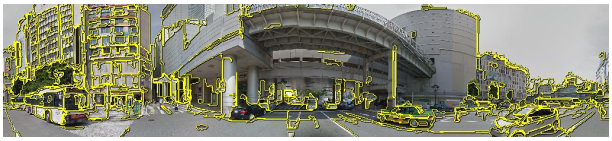

Ground-truth (superimposed)

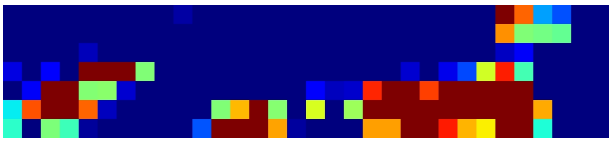

Ground-truth (mask)

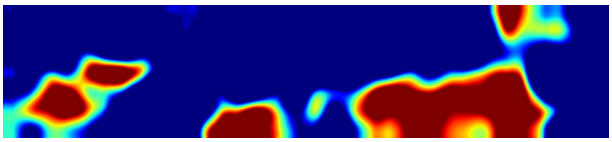

Change estimation (binarized)

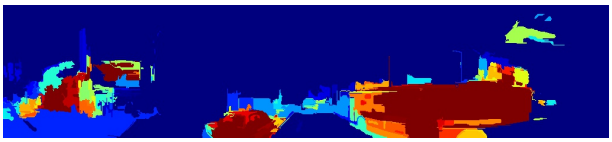

Change estimation (distance)

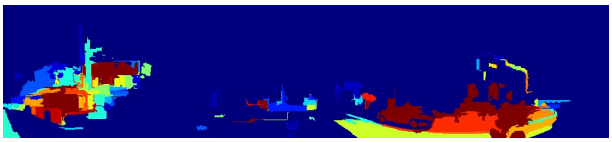

Superpixel segmentation (query)

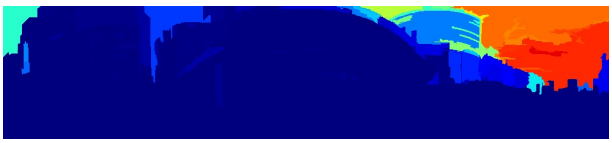

Superpixel segmentation (database)


Feature distance (each grid)

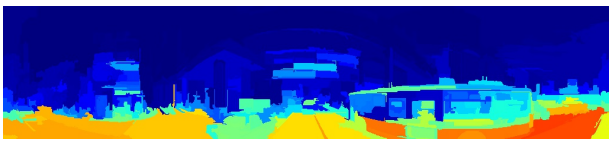

Feature distance (interpolation)


Feature distance in superpixel (query)


Feature distance in superpixel (database)


Sky probabilities (query)


Sky probabilities (database)


Ground probabilities (query)

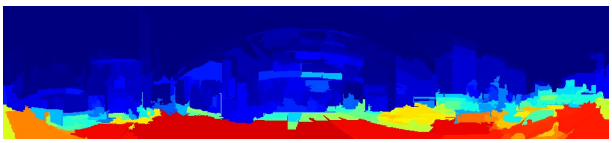

Ground probabilities (database)

Figure 16. Results of change detection using CNN pool-5 features (Frame No. 1/100 in the GSV change detection dataset)

Input image (query)

Input image (database)

Ground-truth (superimposed)

Change estimation (binarized)

(a)

Input image (query)

Input image (database)

Ground-truth (superimposed)

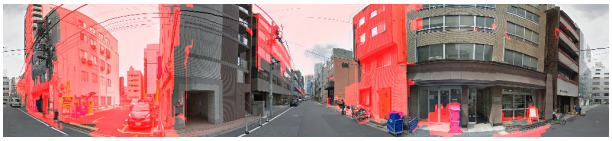
Change estimation (binarized)

(b)

Input image (query)

Input image (database)

Ground-truth (superimposed)



Change estimation (binarized)

(c)

Input image (query)

Input image (database)

Ground-truth (superimposed)

Change estimation (binarized)

(d)

Figure 17. Additional results of change detection using CNN pool-5 features

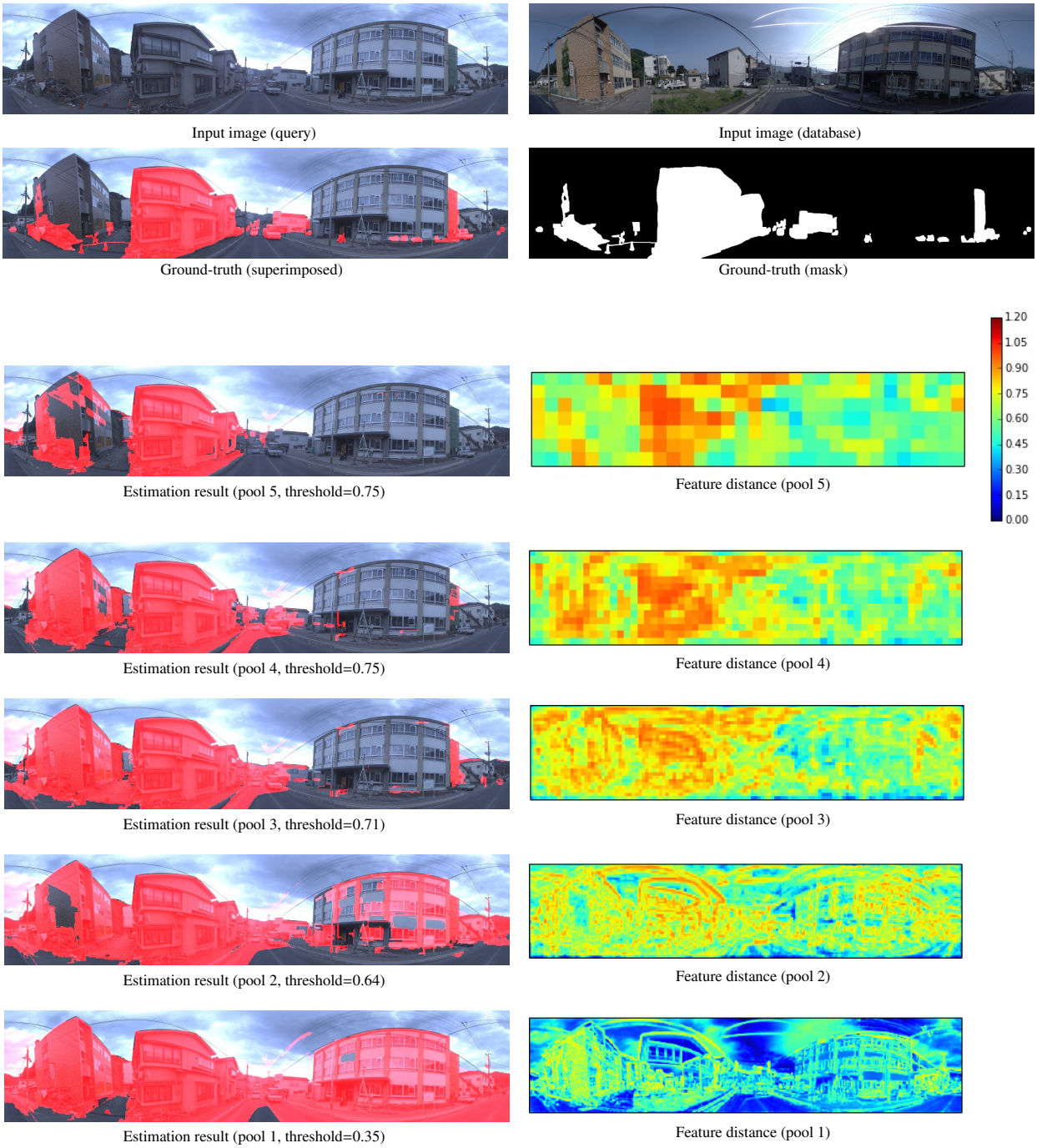


Figure 18. Feature distance of each grid (CNN pooling layers). The distance of the normalized features between each grid $d_i \in \mathbf{d}_g$ takes a value within the range $0 \leq d_i \leq \sqrt{2}$ because all elements of the pooling layer feature are non-negative values.