Detecting Building-level Changes of a City Using Street Images and a 2D City Map

Daiki Tetsuka Takayuki Okatani
Graduate School of Information Sciences, Tohoku University
okatani@vision.is.tohoku.ac.jp

Abstract

This paper presents a method for detecting city-scale changes of a city from its street images and a 2D map. Using SfM to reconstruct point cloud of the structures of the city, the method estimates the existence of each building by matching the point cloud with the 3D building structures recovered from the map. There are multiple difficulties, such as inaccuracy of the recovered building structures, large differences in observation and thus in point cloud size of individual buildings, and mutual dependency of building existences due to potential occlusions. To solve these, we develop a model of how point cloud is generated in the sequential processes of SfM, an observation model of a building wall, and a greedy iterative approach to cope with the mutual dependency. We experimentally apply the method to the cities damaged by the tsunami that struck Japan in 2011. The results show the effectiveness of the method.

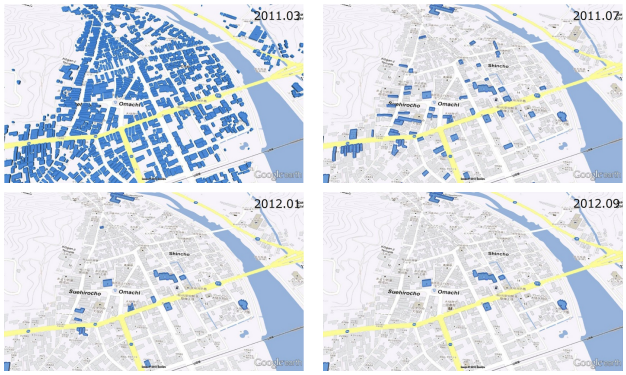

Figure 1. Visualization of the temporal changes of a city. The upper-left image shows the buildings before the disaster. The images in upper-right, lower-left, and lower-right show the results obtained by our method using several image sequences captured at different times. These images visualize how buildings are initially removed by the disaster and then the remaining buildings are gradually demolished in recovery operation.

1. Introduction

This paper considers the problem of detecting large-scale, building-level changes of a city using street images and a map of the city. To be specific, using the image sequence of a city captured by a vehicle-mounted camera while running the vehicle on its streets and a 2D map of the city, we judge whether each building in the map is existing or not.

This study is motivated by a demand to visualize widespread and drastic changes of a city, such as those caused by a natural disaster, in an automatic manner only by running a vehicle with cameras. Its application is not limited to the assessment of disaster damages. We are even more interested in the visualization of how a city is recovering from such a disaster. We can visualize the damages by using a pair of a pre-disaster city map and the post-disaster images. Instead, by using a pair of a future city map (i.e. a post-disaster map created at a point in the future) and the images captured at an earlier time point, we can also create the snapshot of the recovering process of the city.

This study has a specific target, which is the cities severely damaged by the tsunami caused by the massive earthquake that struck a north-eastern part of Japan on March 11th, 2011. The disaster can be said to be quite a rare incident in the history of mankind, as many modern cities over a wide area are forced to change their shapes in such a short time. Thus, the visualization of their damages as well as recovering processes has a lot of applications. Figure 1 shows the visualization of changes of a city before and after the tsunami, which is obtained by the method presented in this paper. It should be noted from a point of view beyond this study that there is a plenty of room for utilizing computer vision techniques in such disaster-related applications. Furthermore, targeting at the tsunami-damaged area of Japan is also beneficial to the development of computer vision techniques, since there are many varieties in damages and recoveries being taken place, which provides a number of ideal test cases for the development of new techniques.

The goal of this study is to detect the changes of a city in the above sense from its images and map. The map we consider is a 2D map in which each building is represented as a

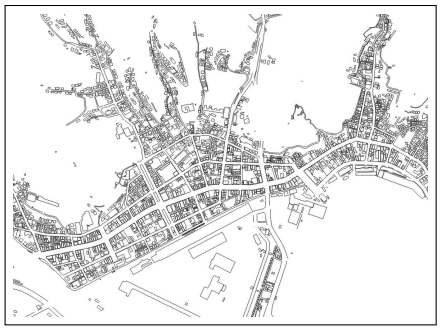

Figure 2. An example of 2D city maps.

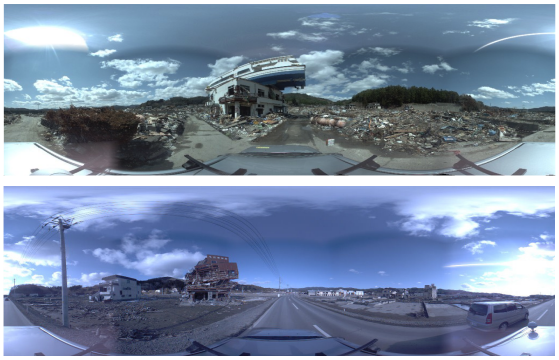

Figure 3. Examples of omnidirectional images of the tsunami-damaged cities of Japan.

polygon. Although 3D maps storing 3D structure are available for some cities in the world, they are rather unavailable for most areas in the world. On the other hand, 2D maps are available for most cities and residential areas in the world. This is indeed the case with the tsunami-damaged area of Japan. It should be noted here that such 2D maps, including the ones we are using, represent each building as an approximated polygon of its ground projection; it is impossible to obtain accurate 3D structures of the buildings from them.

The other input is the sequence of images of the city which is captured by an omnidirectional camera mounted on the roof of a vehicle while running it on city streets. The sequence consists of images captured at every few meters on the streets and GPS data synchronously recorded with each image. In our experiments we used an archive of the images created in this way for the above tsunami-damaged area; see Fig.3. Instead of (or in addition to) this ground-level imagery, aerial imagery could also be used for our purpose. In this study, however, we consider using only the ground-level imagery captured by a vehicle-mounted omnidirectional camera, since they are low-cost and provide high-resolution, ground-level information that are unavailable in aerial imagery.

That said, there are a few disadvantages in the ground-level imagery. A major one is the occurrence of occlusions in the images. If a building is occluded by other buildings, it becomes impossible to judge its existence. This problem will be mitigated by running the vehicle on every street of the city. However, it is costly and is often impossible due to some reasons (e.g., under construction work). Thus, we wish to maximize the accuracy of the judgment as well as the number of buildings being judged, given an image sequence with a limited travel distance of the vehicle.

To achieve the goal outlined above, we propose a method that first reconstructs point cloud from the images by SfM and then compares it against the map to estimate the existences of the buildings on the map. The basic idea is to match the point cloud with the 3D structure of each building recovered from the map and then judge its existence by evaluating how points emerge around to the building. Despite the simplicity of the idea itself, there are several difficulties with this approach, such as inaccuracy of the recovered building structures, large differences in observation and thus in point cloud size of individual buildings, and mutual dependency of building existences due to potential occlusions. To cope with these, we model how point cloud is generated by SfM and also how each building wall is observed in the images, based on which we evaluate the probability that each building is existing in a greedy, iterative fashion.

2. Related work

The most closely related to our study is that of Taneja et al. [12]. It considers the same problem of estimating the building-level changes by using images and a map. However, the city maps used in their study are 3D maps. This is natural since the motivation is to provide a low-cost method of maintaining the 3D model of a city [11]. Thus, it basically considers detecting small changes, not drastic changes such as those caused by a natural disaster. This difference in motivation also results in technical differences. In our study, we use 2D city map. Thus we recover the 3D building structures from them, which have to be necessarily inaccurate. Although the method of Taneja et al. is designed to deal with a certain level of inaccuracy in 3D building models, the inaccuracy in our case is beyond the ranges that their method can handle. Moreover, to deal with the case where the image sequence does not cover every street, our method models potential occlusions among buildings and attempts to judge the existences of the maximum number of buildings with maximum accuracy.

Our study is also close to that of Schindler et al. [10] in that city-scale changes are attempted to be detected by using the point cloud recovered from images by SfM. Given a number of photographs of a city lacking precise temporal information, their method creates the 4D model of the city as well as estimates when these photographs were captured. Their method needs 3D models of buildings (although they could be created from the point cloud if conditions are met). As in our method, it considers potential occlusions among buildings and thus can deal with the mutual dependency

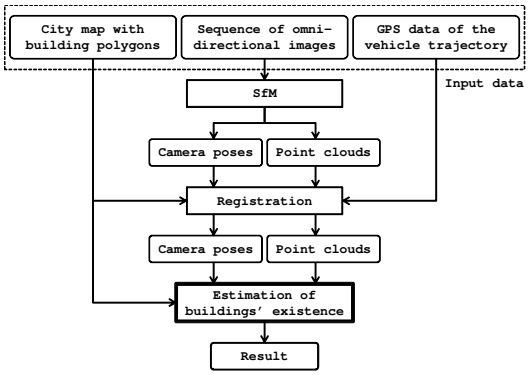

Figure 4. Data Flow diagram of the proposed method.

of building existences. However, since the images it uses are sparse snapshots, their method employs a brute force approach (i.e., MCMC) to resolve the dependency. Our method attempts to resolve it more efficiently by leveraging the nature of the images we use, i.e., a continuous sequence of images captured from a vehicle running in city streets.

Besides these, there are a number of studies of the detection of scene changes from its images [8, 9], most of which use aerial imagery because of the scale of the problem. A study close to ours among them is the method of detecting changes by matching the edges extracted in aerial images with the 3D models of buildings [4]. A number of methods for a more general setting of detecting scene changes from aerial images have been proposed [7, 2].

3. Problem formulation

Figure 4 shows the data flow diagram of the proposed method. It takes three inputs: a sequence of omnidirectional images, associated GPS data, and a 2D map. The method then returns whether each building in the map exists or not, either as a binary value or a continuous probability.

The details of the three inputs are as follows. The omnidirectional image sequence is captured by a camera (Ladybug 3 of Point Grey Research Inc.) mounted on the roof of a vehicle. While running the vehicle on city streets, an image is captured at every two meters. The GPS data recorded synchronously give the position of the camera when capturing each image. We render the omnidirectional images as cylindrical panoramic images for SfM; examples are shown in Fig.3.

The last one of the three inputs is a 2D map, which contains the ground projections of buildings as polygons, as shown in Fig.2. There is no three-dimensional information, or the height data of the buildings. To recover the 3D structures, we assume that each polygon represents the outer walls of a building and that the walls are at least five meters high. The 3D structures thus obtained are not so accurate for several reasons, which needs to be considered in the subsequent processing.

Figure 5. Illustration of differences in observation of individual buildings (best viewed in color). A building close to the camera trajectory (the green curve) will have 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 point cloud emerged. Observe also that potential occlusion makes the building existences mutually dependent.

These three inputs are processed as shown in Fig.4. Firstly, structure-from-motion (SfM) [3, 5, 13] is performed using the sequence of omnidirectional images, yielding the point cloud of various structures of the city as well as the camera trajectory. Next, they are registered to the map using the GPS data of the camera trajectory. The details of these "preprocessing" steps are described in the supplementary material. Finally, for each building in the map, its existence is estimated from the registered point cloud and camera trajectory as well as the map.

The problem we consider in what follows is the final part, a thick line box in Fig.4. In this part, we match and compare the registered point cloud with the building structures recovered from the map to estimate existence of each building. If a building exists, a number of points should be reconstructed around its walls; otherwise, there should be no point. As mentioned above, each building is represented as the polygon whose lines represent the walls comprising the building. We decompose each building into these walls, for each of which we perform the matching and comparison with the point cloud. We then combine the results for the walls to estimate the existence of the building.

4. Details of the proposed method

4.1. Notation

Suppose there are K buildings on the map. We denote the existence of the k-th building by a binary variable $b_k = \{0, 1\}$ ($k = 1, \cdots, K$). We denote the existence of all the buildings by a vector $\mathbf{b} = [b_1, \ldots, b_K]$ and the existence of all but the k-th building by $\mathbf{b}_{-k}$. Let J_k be the number of the outer walls of the k-th building. We denote the existence of the j-th wall of this building by a binary variable $w_{kj} = \{0, 1\}$ ($j = 1, \cdots, J_k$). Let L be the number of camera poses (or equivalently images) in the sequence. We denote the pose (six DOFs) of the l-th camera by $\mathbf{c}_l$ ($l = 1, \cdots, L$).

We will use the angle spanned by a wall of a building viewed from a camera. In its evaluation, we take into ac-

count occlusion by other buildings (the blue triangles of Fig.5). Let $\omega_{kj}^{(l)}$ be this angle of the j-th wall of the k-th building viewed from l-th camera pose. Note that the angle can be evaluated when the existences of all other buildings (i.e., $\mathbf{b}_{-k}$) are known; thus, it may be written as $\omega_{kj}^{(l)}(\mathbf{b}_{-k})$. We set $\omega_{kj}^{(l)} = 0$ if the wall is self-occluded by itself when viewed from the l-th camera.

4.2. Modeling the size of the point cloud emerged from a wall

The images are captured while running the vehicle on streets, and thus individual buildings appear differently in the images. For example, a building closer to the running trajectory will appear larger, and the inverse is true. Furthermore, a building can be occluded by other buildings either partly or completely, depending on its relative position to the trajectory; Fig.5 shows an illustration. If a building appears large and without occlusion in the images, its existence should be determined with a high confidence, and the inverse is true. Thus, the estimation must reflect the quality and quantity of the observations, such as the distance from the trajectory as well as the occlusion. To do so, we model the mechanism of how the point cloud emerges.

Each point comprising the point cloud originates from an image feature point. Thus, we start with modeling how many feature points are extracted for a wall in a single image. We assume here it to be proportional to the area of the wall occupying the image. Although the number should also depend on the texture of the wall, we neglect the dependency by extracting feature points as uniformly as possible in images.

These feature points are attempted to be matched between consecutive images. Successfully matched points form a trajectory of a scene point in the images, from which its 3D coordinates are computed. Thus, we next consider modeling how frequently the feature points are succesfully (or unsuccessfully) matched through the image sequence. However, the behavior is too complicated to precisely model, since the matching can fail due to a number of reasons. Thus, we consider instead evaluating a lower bound of the point cloud size.

We consider here an ideal, simplified model of how point cloud is generated for a wall. In the model, the point cloud size is proportional to the largest area of the wall seen in the image sequence. The underlying assumption is that new trajectories always start before the peak of the area, and they always end after the peak, as shown in Fig.6. As this is an ideal model, it should give a lower bound; the truth is always in the higher side, since a point could be lost before and emerge after the peak, always resulting in the increase in the number of point trajectories.

We calculate the area of a wall in an image as follows. As our map does not contain the height information, we assume each wall to be at least five meters high (and thus will consider only the part of the point cloud below this height). As we are dealing with cylindrical panoramic images, the area of this part of the j-th wall of the k-th building seen from the l-th camera is given by

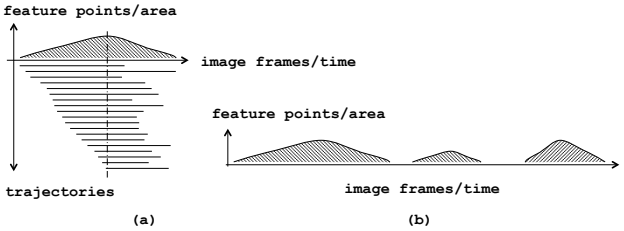

Figure 6. (a) A simplified model of how the image trajectories of the points belonging to a wall emerge and vanish in the image sequence. In this model, their number is proportional to the maximum image area of the wall. (b) A wall could be observed multiple times, which increase the number of points.

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

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

As discussed above, we assume the maximum of $v_{kj}^{(l)}$ for $l = 1, 2, \ldots$ to give the bound of the point cloud size. More precisely, the maximum is taken for the period since the area begins to have a non-zero value and until it becomes zero. Thus, if there are multiple such periods, as shown in Fig.6(b), we need to sum over the largest areas for those periods. This reflects the nature of our SfM implementation; once a scene point is lost in the tracking, the same point will be reconstructed as a different scene point when it is rediscovered. To represent this summation of the largest areas, we divide 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, 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 $\mathbf{b}_{-k}$.

4.3. Observation model of a wall

To estimate the existence of a wall, we first decide which points belong to a wall. A point is judged to belong to a wall if its orthogonal projection lies inside the wall (the rectangle of the polygon line of the wall × five meters high) and whose distance is less than a threshold (we used 2.5 meters in the experiments to deal with the 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}

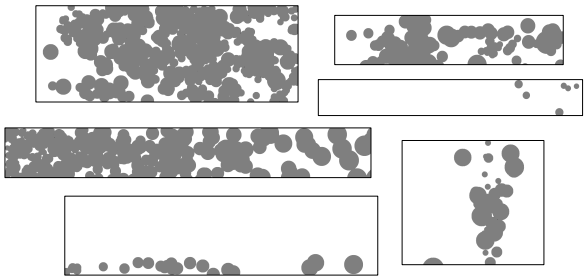

Figure 7. Definition of r_{kj}. Each rectangle represents a wall. For the point cloud emerged around the wall, each point is projected onto it, at which a filled circle is drawn on the wall. The radius of the circle is chosen to be proportional to the distance from the camera observing the point to the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

of the point cloud size as q_{kj}/v_{kj} and use it to estimate the existence of the wall.

However, points are emerged not only from building walls but also from other objects. There could be irrelavant objects nearby a wall such as telegraph poles, electric wires, and (remaining) building fonudatations. As these objects often stand close to building walls, merely counting the number of points could lead to erroneous estimation.

To mitigate this, we propose an observation model of walls. It evaluates not only the population size but the distribution of the points. If a wall exists and generates points, they should distribute over most of its area in a 2D manner. We could use this fact for our purpose, as the above irrelevant objects basically generate points only in a one-dimensional manner.

A possible approach to this is to assume the point cloud to obey a uniform distribution and then perform some statistical test of the uniformity. However, this will work only when there are only the following two cases: the wall exists and there is no other object nearby or the wall does not exist and there either are or are not other objects nearby. There could be in reality the case where a wall and other objects nearby both exist. This complicates the point distribution and makes the statistical test more difficult.

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 of the projected points, as shown in Fig.7. The ratio of the filled area a'_{kj} to the total area a_{kj} of the wall is calculated as

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

We regard this as an observation for the wall. We vary the radius of the circles depending on the distances to the wall from the cameras observing the point. To be specific, letting d be the minimum distance (between the wall to the cameras observing the point), we set the radius α for each point as

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

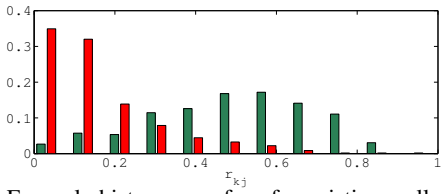

Figure 8. Example histograms of r_{kj} for existing walls (in green) and non-existing walls (in red). These are created for the selected walls that have a large v_{kj} value.

where c_r is a constant; we set it to 0.033 radians throughout the experiments. It corresponds to an approximated 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.

We judge the existence of the wall (k, j) from the observation r_{kj}. To do this, we consider and 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 if $\mathbf{b}_{-k}$ is given and thus should be treated as a random variable, is treated here as a parameter. Figure 8 shows example histograms created from the data of a city with its ground truth of building existences. These histograms are created for the selected walls that have high visibility, i.e., have a large v_{kj} value. It is seen that the density of r_{kj} for existing walls has a volume around a large value and the density for non-existing walls has a volume around zero.

These two densities are considered to vary their shapes depending on the value of v_{kj} and are functions of two variables. To model them, we need a vast amount of training data, which requires ground truths of building existences for a number of cities. However, such ground truths are costly to create because of their scale. Therefore, we employ the simplest model here, in which each density changes only once depending on v_{kj} as

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

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

We set the thresholding value c_v to 0.1 in the experiments. It determines which walls to be judged since they are well observed and which ones not to be judged since they are insufficiently observed. Assuming a wall to be observed once in the image sequence, $c_v = 0.1$ is roughly equivalent to that the wall occupies at maximum about 5% width of the cylindrical panoramic image. The two densities p_{exist} and p_{vanish} model the green and the red histograms shown in Fig.8, respectively. We choose truncated Gaussian densities for them; $N(0.5, 0.03)/0.9961$ for p_{exist} $N(0.0, 0.01)/0.5$ for p_{vanish}. (The denominators normalize the density.) The remaining $p_{invis}(r_{kj})$ indicates the density for walls that have low visibility; there is only a single density regardless of the

existence of the wall. This density needs not to be explicitly modeled in the estimation of building existences described next.

4.4. Estimation of building existences

We now consider how to estimate the existence of each building. To do this, we consider evaluating the posterior $p(b_k = 1|r_{k1}, \ldots, r_{kJ_k})$ of the existence of the k-th building given all the associated observations r_{kj}'s. Although the variables $r_{k1}, \ldots, r_{kJ_k}$ are mutually dependent on each other in a complicated way, we introduce a naive independence assumption and approximate it as

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

The second equality holds since $b_k = 1$ 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 there is no prior on b_k and set $p(b_k = 1) = p(b_k = 0) = 1/2$, it becomes

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

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.

4.5. Greedy iterative estimation

Eq.(7) gives the existence probability of the k-th building. However, we cannot evaluate it straightforwardly because of the mutual dependency of the building existences; 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 only be evaluated if the existences $\mathbf{b}_{-k}$ of other buildings are given.

To cope with this, we propose a greedy iterative approach. Starting with assuming all the buildings to be existing ($b_k = 1$), we iteratively make a binary decision ($b_k = 0/1$) one by one for the buildings in order of decreasing confident level. The confidence level is given by $p(b_k|\cdot)$ itself of Eq.(7) (that is, how close to either 0 or 1 it is). In accordance with the determination of b_k's for some buildings, we recalculate v_{kj}, which will change the values of $p(b_k|\cdot)$'s.

This iteration should work, because the existence of a building facing a street on which the vehicle run should be able to be correctly estimated independently of other buildings. After the existence of this building is determined, there will be more chance of correctly determining those of the buildings potentially occluded by this building.

To be more specific, we introduce a binary variable $d_k = \{0, 1\}$ which indicates whether or not the decision has been made for the k-th building; $d_k = 1$ means decided and $d_k = 0$ means undecided. We recompute v_{kj}'s at each iteration step,

Algorithm 1 The algorithm for estimating building existences.

Input: Points and camera trajectories generated by SfM and building polygons. **Output:** Existences b_k's for the buildings judged ($d_k = 1$) and existence probabilities of 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 the buildings with $d_k = 0$ and those with $d_k = 1$ and $b_k = 1$ are existing, evaluate v_{kj} for any pair (k, j) according to Eq.(2).
5: Compute the probability of Eq.(7) for each of the buildings with $d_k = 0$ using the precomputed r_{kj}'s and the latest v_{kj}'s.
6: Set b_k and d_k according to Eq.(8).
7: **until** Existence is judged for no new building.

where occlusion is evaluated by using the existences b_k's for the decided buildings with $d_k = 1$ and by assuming the buildings with $d_k = 0$ to exist. In the first iteration, we set $d_k = 0$ for all the buildings resulting in evaluating v_{kj}'s by assuming all the buildings to exist (but the k-th one). The buildings facing a street on which the vehicle run will appear large without occlusion in the images, and thus its facing wall(s) has a large v_{kj} value.

The straightforward implementation of the above process requires large computational time, as v_{kj}'s are updated at the decision of every building. To reduce the cost, we incorporate two approximation. One is to introduce a threshold c_m for confidence and make a decision immediately for all the buildings with confidence levels beyond this threshold. This significantly reduces the count of v_{kj} updates, or equivalently the number of iteration counts. 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:

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

Note that we do nothing if $1/2 - c_m \leq p(b_k|\cdot) \leq 1/2 + c_m$. The parameter c_m controls how aggressive the decision is made; it ranges from 0.0 to 0.4 in the experiments. The algorithm is summarized as Algorithm 1.

5. Experimental results

We conducted experiments to test the effectiveness of the proposed method. We used an image archive created for the tsunami-damaged area of the north-eastern coastline of Japan, from which we chose three cities that differ in the amount of damages: *Otsuchi Town, Miaygino Ward,* and *Kamaishi City*. Figure 9 shows their SfM reconstructions registered with the maps. The number of images used for SfM are 2870, 18002, and 1266, respectively.

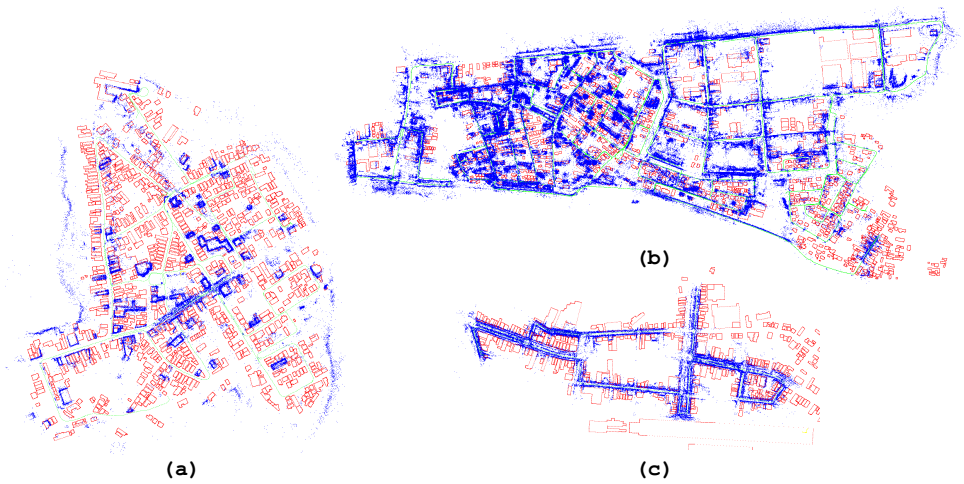

Figure 9. SfM reconstructions for the three cities tested in the experiments; best viewed in color. The reconstructed points and camera trajectories are shown as blue dots and green curves, respectively. The building polygons are shown as red lines. (a) Otsuchi town (reconstructed from 2870 images). (b) Miyagino ward (18002 images). (c) Kamaishi city (1266 images).

As there is no previous study dealing with the same problem setting as our study, we implement two variants of the proposed method by omitting the main ingredients, and compare them in performance to evaluate their effectiveness.

Apart from the basic idea of comparing SfM point cloud with building walls, there are three ingredients in the proposed method that we think are contributions of this study. The first one is v_{kj} introduced in Sec.4.2 to quantify how each wall appears in images. The second is r_{kj} introduced in Sec.4.3 to quantify the uniformity of points on each wall. The third is the greedy iterative scheme to cope with the mutual dependency of building existences. To test the effectiveness of the first two, we consider a variant that does not use v_{kj} and one that does not use r_{kj}. To be specific, the first variant is implemented by simply setting $c_v = 0$ in Eq.(5), which is evaluated in Eq.(7). The second variant is implemented by replacing r_{kj} with a simpler quantity q_{kj}/v_{kj}, which is first introduced in Sec.4.3; the densities for the quantity are modeled as in Eq.(5) for r_{kj}. To examine the effectiveness of the greedy iterative scheme, we varied the margin c_m controlling how many buildings to be judged for each of the three methods.

Table 1 shows the results. The method with all the proposed ingredients, 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. It is observed from the results that "v and r" achieves the best accuracy for the two cities; "v and q" is the best for the third city but "v and r" is comparatively good; and "q alone" yields a very inaccurate result for the third city. The reason for the last observation may be due to the differences among the cities that a large number of buildings are demolished in the first two cities, whereas a relatively small number of buildings are demolished in the third city.

Figure 10 visualizes how the building existences are decided with the iteration counts. The buildings that have not been judged, those judged to be existing, and those judged to be non-existing are shown in green, blue, and red, respectively. It is seen that the existences of a certain number of buildings are judged to be either existing or non-existing at each step. It is also seen that the buildings that are distant from the camera trajectory (shown in blue curves) and/or are occluded from others are left undecided until the end. These show that the algorithm works as we intended. More results for other cities and also the analyses of erroneous estimations of building existences are given in the supplementary material.

Figure 1 shows the results of applying our method to the image sequences captured at different times after the disaster. It demonstrates that our method can be used to visualize how a city changes its structure in the temporal axis.

6. Summary

We have described a method for detecting building-level changes of a whole city by using its 2D map and image sequence captured by a vehicle-mounted camera. It uses SfM to reconstruct 3D point cloud of the structure of the city and matches it with the 3D structures of buildings recovered from the 2D map. To cope with multiple difficulties with the approach, introducing the model of the mechanism of how point cloud is generated and also the observation model of each building wall, the method evaluates the existence probability of each building in a greedy, iterative fashion. The experimental results show the effectiveness of the method. A future work will be further improvement of

Table 1. Recognition rates of the compared methods. The numbers in the leftmost column is (the number of existing buildings)/(the total number of buildings). In each cell, the upper number is the recognition rate and the lower number is the answering rate, both in percentages(%).

	v and r					v and q					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

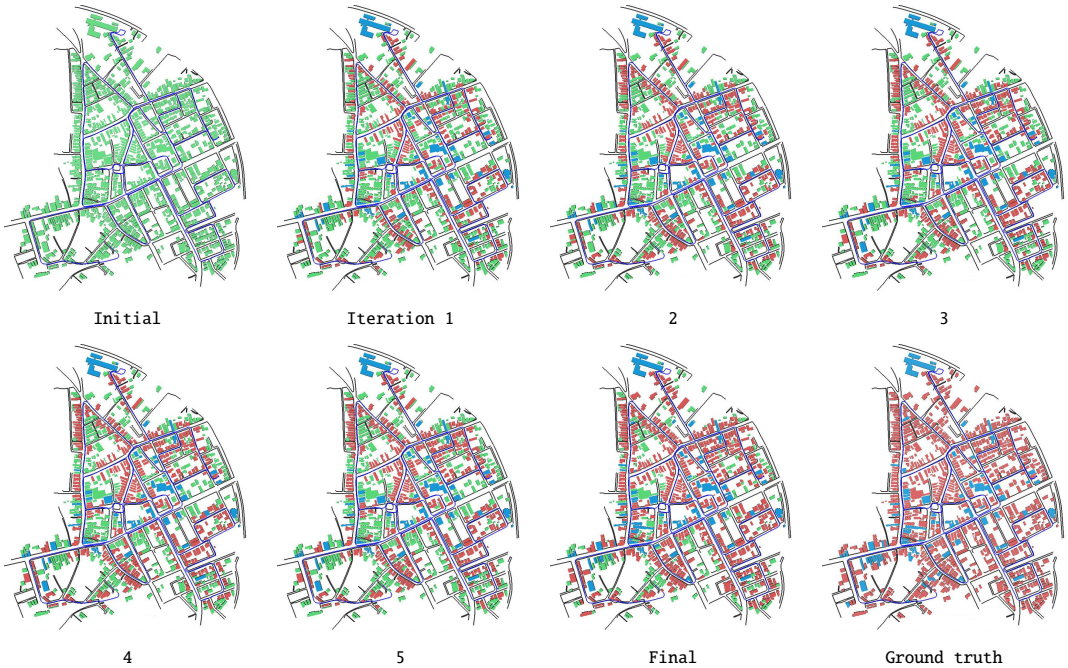

Figure 10. Intermediate results at different iteration counts for Otsuchi Town; best viewed on a color monitor. Green, blue, and red polygons indicate the buildings that have not been judged yet, those judged to be existing, and those judged to be non-existing, respectively. The camera trajectory is displayed in dark blue.

the estimation accuracy.

References

[1] H. Bay, A. Ess, T. Tuytelaars, L .Van Gool. SURF: Speeded Up Robust Features. *Computer Vision and Image Understanding*, 110(3):346–359, 2008.

[2] D. Crispell, J. Mundy, and G. Taubin. A Vairable-Resolution Probabilistic Three-Dimensional Model for Change Detection. *Geoscience and Remote Sensing*, 50(2):489–500, 2012.

[3] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision Second Edition*. Cambridge University Press, 2004.

[4] A. Huertas and R. Nevatia. *Detecting Changes in Aerial Views of Man-Made Structures*. In *Proc. International Conference on Computer Vision*, pages 73–80, 1998.

[5] D. Nistér. An Efficient Solution to the Five-Point Relative Pose Problem. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 26(6):766- -777, 2004.

[6] 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, and H. Towles. Detailed Real-Time Urban 3D Reconstruction from Video. *International Journal of Computer Vision*, 78(2-3):143–167, 2008

[7] T. Pollard and J. L. Mundy, Change Detection in a 3-D World. In *Proc. Computer Vision and Pattern Recognition*, pages 1–6, 2007.

[8] R. J. Radke, S. Andra, O. Al-Kofahi, and B. Roysam. Image Change Detection Algorithms: A Systematic Survey. *IEEE Trans. Image Processing*, 14(3):294–307, 2005.

[9] K. Sakurada, T. Okatani, and K. Deguchi. Detecting Changes in 3D Structure of a Scene from Multi-view Images Captured by a Vehicle-Mounted Camera. In *Proc. Computer Vision and Pattern Recognition*, pages 137–144, 2013.

[10] G. Schindler and F. Dellaert. Probabilistic temporal inference on reconstructed 3D scenes. In *Proc. Computer Vision and Pattern Recognition*, pages 1410–1417, 2010.

[11] A. Taneja, L. Ballan, and M. Pollefeys. Image based detection of geometric changes in urban environments. In *Proc. International Conference on Computer Vision*, pages 2336–2343, 2011.

[12] A. Taneja, L. Ballan, and M. Pollefeys. City-Scale Change Detection in Cadastral 3D Models using Images. In *Proc. Computer Vision and Pattern Recognition*, 2013.

[13] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon. Bundle Adjustment - Modern Synthesis. In *Proc. International Conference on Computer Vision*, pages 298–372, 1999.