Fast Image-Based Geometric Change Detection Given a 3D Model

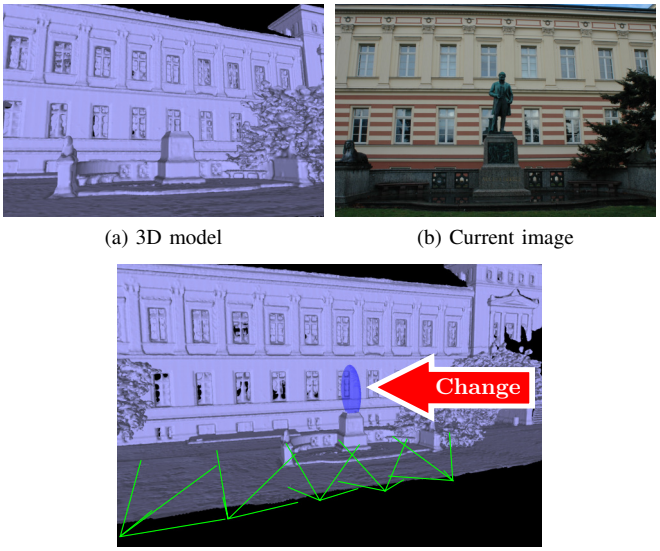
Emanuele Palazzolo Cyrill Stachniss

***Abstract*— 3D models of the environment are used in numerous robotic applications and should reflect the current state of the world. In this paper, we address the problem of quickly finding structural changes between the current state of the world and a given 3D model using a small number of images. Our approach finds inconsistencies between pairs of images by re-projecting an image onto another one by passing through the given 3D model. This process leads to ambiguities, which we resolve by combining multiple images such that the 3D location of the change can be estimated. A focus of our approach is that it can be executed fast enough to allow the operation on a mobile system. We implemented our approach in C++ and released it as open source software. We tested it on existing datasets as well as on self-recorded image sequences and 3D models, which we publicly share. Our experiments show that our method quickly finds changes in the geometry of a scene.**

I. Introduction

Building 3D models of the environment is a frequently addressed problem in robotics as they are needed for a wide range of applications. For most applications that include autonomous behavior, such models should correspond as well as possible to the current state of the environment. In case the environment changed substantially, existing models must be updated. For this purpose, the possibility of directing a mapping or exploring robot directly towards the possible regions that have changed instead of repeating the whole mapping process is advantageous. Therefore, it is important to reliably identify locations in a 3D model that have changed.

In this paper, we address the problem of finding changes between a previously built 3D model and its current state based on a small sequence of images (keyframes) recorded in the environment, see Fig. 1 for an illustration. Two aspects are important here: first, we want to reliably locate changes in the model and second, the approach should have a limited computational demand so that it can be executed on a mobile platform, allowing an exploring or mapping robot to plan its next action according to the location of change. Our approach seeks to find changes between the current state of the world and a previously recorded 3D model of the scene. For finding inconsistencies, we do not build a new 3D model from the newly obtained image data and compare the result to the existing one. Instead, we back-project the currently obtained image onto the 3D model and then project it to a viewpoint at which another image of the current sequence has been taken. Through a comparison between the re-projected images and the one observed in reality, we can identify possible regions of change. To eliminate ambiguities, this process is executed for multiple image pairs. Typically, 4-5 keyframe images are sufficient to find areas of change and then estimate the 3D location where the geometry has changed. Compared to existing approaches for visual change detection such as the work by Taneja et al. [20] or Ulusoy et al. [22], our method is substantially faster towards execution on a mobile robot. Note that our approach only compares images from the current sequence with each other, i.e., no current image needs to be compared to an old one. This makes our approach robust to seasonal changes, weather conditions, or any changes that are usually present when taking image sequences at two distinct points in time.



(a) 3D model (b) Current image

(c) Result

Fig. 1: Our approach aims at quickly finding changes in the environment based on an existing 3D model and a sequence of (currently recorded) images.

The main contribution of this paper is a new and fast approach for identifying differences between an existing 3D model and a small sequence of images recorded in the environment. Our approach identifies the approximate area of change fast enough to be executed on a navigating robot, which sets it apart from several related other techniques. We identify inconsistencies by comparing the acquired images to re-projected images that would have been obtained assuming the 3D model is correct, in combination with a forward intersection of the potentially inconsistent regions. We implemented our approach in C++ and tested it on both self-recorded and existing datasets. Our experiments show that our method quickly finds the approximate location of

All authors are with the University of Bonn, Institute of Geodesy and Geoinformation, Bonn, Germany.

This work has partly been supported by the DFG under the grant number FOR 1505: Mapping on Demand.

the change in the scene and is fast enough to potentially guide an exploring ground robot or UAV seeking to map the changes in the environment. We publicly share both our open source implementation[1] and our own dataset including the 3D models[2].

We make two key claims: our approach is able to (i) identify the location of changes in the environment, in the form of 3D volumes in the world coordinate frame, using a 3D model and a sequence of images, and (ii) it is fast enough to be executed on a mobile robot, i.e. analyzing a sequence of keyframe images does not take longer than recording it (e.g., a few seconds for a sequence of five keyframe images).

II. Related Work

Building 3D models can be an expensive process as it requires a good coverage of the environment [10] and potentially dedicated sensors or equipment. To reduce this cost, it is important to identify, on an existing model, the parts that have changed, and direct the exploration towards those locations. For this reason, 3D change detection is an increasingly popular topic, see [13].

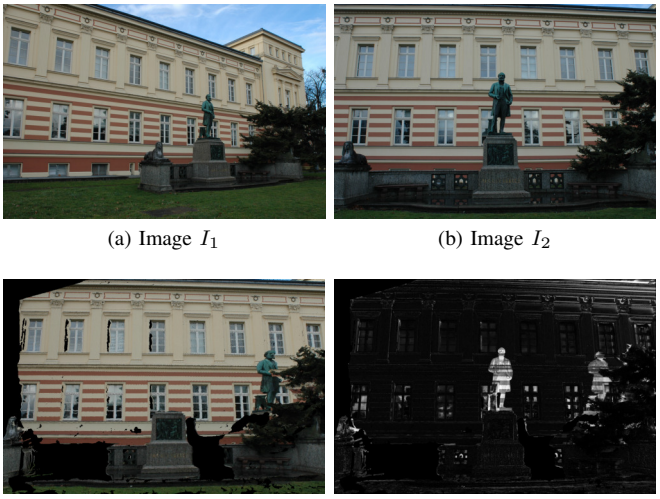
In the past, many 2D change detection algorithms have been proposed [14]. Several of such methods are affected by lighting conditions, seasonal changes, weather conditions, and other differences that may occur between the recording of the old and the new images. However, under certain conditions, the 2D approach can still be useful, e.g. for monitoring a tunnel surface, as proposed by Stent et al. [18]. Another limitation of 2D approaches is that the images often do not provide information on the actual 3D location of the change. Sakurada et al. [16] try to overcome these problems by estimating the probabilistic density of the depth from the old set of images and by comparing it with the depth computed from the new set of images. Eden et al. [4] compare 3D lines in the images instead of using color or intensity information. A more recent approach by Sakurada and Okatani [15] instead uses a deep convolutional neural network to detect changes in omnidirectional images. Alcantarilla et al. [1] also use a deep convolutional neural network, but they additionally combine it with a dense reconstruction technique.

Another approach to 3D change detection is to build a 3D model from the new images through Multi-View Stereo and then compare the new model with the old one. However, this is often a rather time consuming activity. Golparvar-Fard et al. [7] use this approach combined with a support vector machine classifier to obtain an updated voxelized model of the environment.

A popular and effective approach is to infer the changes of the environment using a previously built 3D model and a sequence of newly acquired images. One way to achieve this is to maintain a voxelized model of the environment and detect the probability of change in it by comparing the color of a voxel and the color of the pixels in the images onto which it projects. Examples of this approach are the one by Ulusoy et al. [22] or the one by Pollard et al. [11].

[1] github.com/Photogrammetry-Robotics-Bonn/fast_change_detection

[2] www.ipb.uni-bonn.de/data/changedetection2017



(a) Image I_1 (b) Image I_2

(c) Re-projection of I_1 onto I_2 (d) Inconsistencies

Fig. 2: A pair of images, the first image re-projected onto the second, and the inconsistencies between them.

Another relevant strategy that uses an existing 3D model and newly acquired images is to identify changes by re-projecting images onto each other by passing through the existing model and compare the inconsistencies in the re-projection. Taneja et al. [20] use this technique on pairs of images, and apply a graph cut minimization to label the changed area in 3D in a voxelized model. This technique is also effective for large scale change detection [21]. In addition, Qin et al. [12] combine the pairwise detected inconsistencies by counting the rays that hit every pixel for each image, in order to get rid of the ambiguities. They stop at the image level and do not estimate the 3D location of the change.

In this paper, we use a re-projection technique similar to [20] and [12] to identify the changed regions in the images. We resolve ambiguities by fusing multiple images and introduce a fast way for estimating the rough location of change in 3D. The whole process takes only a few seconds for an image sequence. In contrast to that, state-of-the-art approaches such as [20] or [22] have execution times in the order of minutes. This paper extends our previous workshop paper [9] presenting an optimized algorithm and an extended experimental evaluation.

III. Fast Image-Based Change Detection

Our approach aims at spotting areas in an environment that have changes with respect to a previously built 3D model. It does so by exploiting a sequence of around five images through evaluating how the projections of image content from one image to the model and back to another image look like. In terms of computational demands, this process is substantially more efficient than generating a new, dense 3D model and comparing it directly with the given one. The first step is to detect possible inconsistencies of an image with its neighboring images assuming that the 3D model is correct.

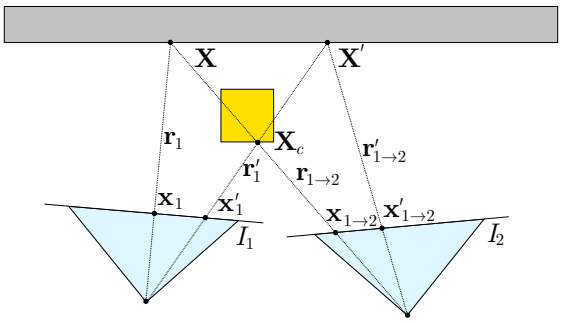


Fig. 3: Re-projection procedure. The gray rectangle represents the known 3D model, while the yellow square is a change not present in the original model. Using two images, a point $\mathbf{X}_c$, not present in the model, is re-projected onto two pixels $\mathbf{x}_{1\to2}$ and $\mathbf{x}'_{1\to2}$.

After computing pairwise inconsistency hypotheses, we fuse them to eliminate the intrinsic ambiguities and estimate the location of change by triangulation. Given that we look for inconsistencies between the 3D model and new images, our approach only finds changes from images where the rays corresponding to pixels intersect with the 3D model.

Note that we assume a good pose estimate for the camera. We obtain the (approximate) location of the 3D model and the viewpoint of the images as described in Sec. III-A below.

A. Camera Pose Estimate

Our algorithm requires an estimate of the viewpoints of the images w.r.t. the 3D model. We obtain this through direct georeferencing fusing GPS, IMU, and visual odometry, as described in [17]. The approach employs the iSAM2 algorithm, and provides uncertainty information about all sensor poses in form of a covariance matrix. In case no GPS information is available, approaches for camera to 3D model localization such as [2] can be used, although we did not directly try that here.

B. Inconsistencies Between Image Pairs

Given the calibration matrix and the pose at which the camera took an image I, we can compute the projection of an arbitrary 3D point $\mathbf{X}_{\text{world}}$ onto the image plane resulting in a 2D point at pixel $\mathbf{x}$:

$$\mathbf{x} \quad = \quad \mathbf{P}\mathbf{X}_{\text{world}}, \tag{1}$$

where $\mathbf{x}$ is expressed in homogeneous coordinates and $\mathbf{P} = \mathbf{K}[\mathbf{R}|-\mathbf{R}\mathbf{t}]$ is the camera projection matrix computed from the calibration matrix $\mathbf{K}$ of the camera and the rotation $\mathbf{R}$ and translation $\mathbf{t}$ that transform the world coordinates into camera coordinates.

By inverting Eq. (1), we compute the ray from the projection center of the camera through the pixel to the 3D world. This allows us to back-project each pixel of I onto the 3D model assuming the known intrinsic parameters $\mathbf{K}$ and the rotation matrix $\mathbf{R}$ from the extrinsic parameters:

$$\mathbf{r} \quad = \quad \mathbf{R}^\mathsf{T}\mathbf{K}^{-1}\mathbf{x}, \tag{2}$$

where $\mathbf{r}$ is the direction of the ray in world coordinates.

To detect inconsistencies between a pair of images consisting of the images I_1 and I_2, we create a new image $I_{1\to2}$ that represents the content of I_1 as seen from the view point of I_2 given the 3D model. Given that we know, from Eq. (2), the view direction $\mathbf{r}_1$, we compute the intersections $\mathbf{X}$ between the rays and the 3D model and project $\mathbf{X}$ onto the image plane of I_2 to obtain $I_{1\to2}$ (see Fig. 2c for a real example):

$$\mathbf{x}_{1\to2} \quad = \quad \mathbf{P}_2\,\mathbf{X}, \tag{3}$$

where $\mathbf{P}_2$ is the camera projection matrix corresponding to image I_2. In this way, we obtain a new image $I_{1\to2}$ that can be compared to I_2. Since the *exact* poses of the cameras are unknown and the 3D model is not perfect, the point $\mathbf{x}_{1\to2}$ has an uncertainty represented by the covariance matrix $\mathbf{\Sigma} := \mathbf{\Sigma}_{\mathbf{x}_{1\to2}\mathbf{x}_{1\to2}}$. To consider this uncertainty, we compute, for every pixel of I_2 the minimum Euclidean norm of the intensity difference to each pixel of $I_{1\to2}$ in a neighborhood $\mathcal{N}_{i,j}$ around the projected pixel. We compute the size of this neighborhood by propagating the pose uncertainty, obtained while recording the images, into the image points (see Sec. III-A). In detail, we search within the 3σ area given by $\mathbf{\Sigma}$ and select the pixel with the smallest difference:

$$D_{1\to2}(i,j) = \min_{k,l\in\mathcal{N}_{i,j}} ||I_2(i,j) - I_{1\to2}(k,l)||_2, \tag{4}$$

where i, j, k, l are pixel coordinates and the neighborhood $\mathcal{N}_{i,j}$ is defined as:

$$\mathcal{N}_{i,j} = \left\{ \forall (k,l) \in I_{1\to2} \;\middle|\; \begin{bmatrix} i-k \\ j-l \end{bmatrix}^\mathsf{T} \mathbf{\Sigma}^{-1} \begin{bmatrix} i-k \\ j-l \end{bmatrix} < d^2 \right\}, \tag{5}$$

where $d^2 = 11.82$ is the critical value of the χ^2_2 distribution corresponding to a probability of 99.73%, i.e. the 3σ boundary on the normal distribution. Finally, we normalize $D_{1\to2}$ to values between $[0, 1]$. Fig. 2d shows the result of this procedure.

If there is no change in the 3D model between the acquisition time and the time when the images have been taken, all pixels in I_1 should correctly re-project onto I_2. Therefore, I_2 and $I_{1\to2}$ should be identical and $D_{1\to2}$ should be small or equal to 0 for each pixel. If there is, however, a change in the model, pixels corresponding to the change re-project onto the wrong place in I_2. Thus, $D_{1\to2}$ allows us to identify the changes (as long as not all pixels in the current images have the same RGB value, i.e. represent a large homogeneous area)

The process, however, has ambiguities. As Fig. 3 illustrates, a single point $\mathbf{X}_c$ corresponding to a change in the 3D model generates two pixel locations, $\mathbf{x}_{1\to2}$ and $\mathbf{x}'_{1\to2}$, in $D_{1\to2}$, one corresponding to the change in I_1 re-projected onto I_2 and one corresponding to the change in I_2 re-projected onto I_1. To eliminate this ambiguity, we use multiple pair-wise image comparisons as described in the following section.

C. Inconsistency Detection using Multiple Images

The ambiguity produced by the re-projection of an image onto another one can be eliminated by considering multiple image pairs. Fig. 4 shows how a pixel belonging to the same change in a third image I_3 re-projects onto I_2 at two different

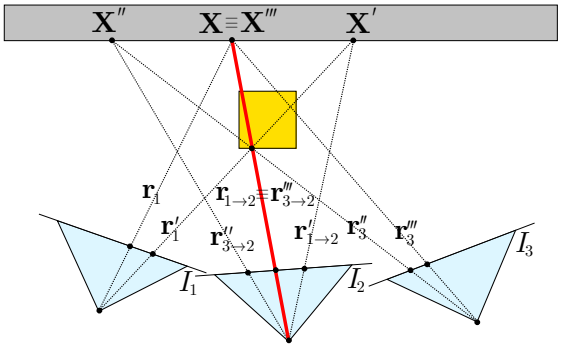


Fig. 4: Ambiguity elimination using multiple images. When re-projecting I_1 and I_3 onto I_2, only one ray (therefore one pixel) is coincident. The thicker red line represents that coincident ray.

locations. It is important to note that one of the two points is mapped to the same location as a change detected by re-projecting I_1 onto I_2. Thus, the pixels that re-project onto the same region of I_2 from the other images represent the real change.

To localize the changes, we therefore compare an image with its m neighboring keyframe images. For each image I_t, we store an inconsistency image D_t resulting from the pixel-wise minimum over all the inconsistency images obtained from the neighboring images re-projected onto I_t:

$$D_t(i,j) \quad = \quad \min\{D_{s\to t}(i,j),\ \forall s \in \mathcal{S}(t)\}, \qquad (6)$$

where $\mathcal{S}(t)$ is the set of m neighboring keyframe images of I_t. In our implementation, we typically use the four closest images in time to I_t. Fig. 5 depicts the output of Eq. (6), for $m = 1, 2, \text{and } 3$. Even though it is not easy to see in Fig. 5, the noise in the total inconsistency image is substantially smaller when using more than $m = 2$ neighboring images. Thus, we stick to $m = 4$, although using $m = 2$ is theoretically sufficient.

D. Segmentation and Data Association

The procedure explained so far enables us to identify the pixels in each image where changes occur. For reliably computing the regions of change, we first filter out the noise with an erosion-dilation procedure, then apply a standard border following algorithm [19]. We discard all the regions with a contour shorter than a threshold (in our implementation 50 px for images with horizontal resolution of 500 px) to filter out noise and changes that are too small. The next step is to associate the regions from the images with each other. To do that, we compute and compare hue-saturation histograms region-wise and perform standard cross-correlation together with a simple geometric consistency check using the epipolar lines.

E. Estimating the Location of Change

Once we obtain the segmented 2D regions and the association between them, we proceed to estimate the 3D location of the change. To simplify the notation in the remainder of this section, the following equations will refer to a single change in images, i.e. dropping an index referring to individual regions. The whole procedure is repeated for every region (of detected change).

To estimate the 3D volumes in which the changes occur, we first compute, for every region identified as a change, the mean location $\bar{\mathbf{x}}_t$ and spread in form of the covariance $\mathbf{\Sigma}_t$ in the image. We then compute, for each change, a 3D point $\overline{\mathbf{X}}$ in the 3D world coordinates by triangulating the mean location in each image [6]. Specifically, we setup a system of equations in the form

$$\mathbf{A}\overline{\mathbf{X}} = \mathbf{0}, \quad \text{with} \quad \mathbf{A} = \begin{bmatrix} \mathsf{S}(\bar{\mathbf{x}}_1)\mathbf{P}_1 \\ \vdots \\ \mathsf{S}(\bar{\mathbf{x}}_n)\mathbf{P}_n \end{bmatrix}, \qquad (7)$$

where $\mathbf{A}$ is a $3n \times 4$ matrix composed by 3×4 blocks, n is the number of images, $\mathbf{P}_t$ is the projection matrix relative to image I_t, and $\mathsf{S}(\bar{\mathbf{x}}_t)$ is the skew symmetric matrix corresponding to the mean pixel $\bar{\mathbf{x}}_t$, in homogeneous coordinates, i.e.:

$$\bar{\mathbf{x}}_t = \begin{bmatrix} x_t \\ y_t \\ w_t \end{bmatrix}, \quad \mathsf{S}(\bar{\mathbf{x}}_t) = \begin{bmatrix} 0 & -w_t & y_t \\ w_t & 0 & -x_t \\ -y_t & x_t & 0 \end{bmatrix}. \qquad (8)$$

We solve this system using singular value decomposition and retrieve $\overline{\mathbf{X}}$ by taking the right-singular vector of $\mathbf{A}$ belonging to its smallest singular value (see Fig. 6a for an example of triangulation). For each change in the image, we additionally compute the K sigma points $\mathbf{v}_t^{(k)}$ $(k = 1 \dots K)$ corresponding to $\bar{\mathbf{x}}_t$ and $\mathbf{\Sigma}_t$ and project the sigma points to the 3D space to estimate the region of change in 3D. Using the sigma points allows for a better propagation of a Gaussian through a non-linear function than first-order error propagation, see [8] for details. To compute the 3D position of the sigma points, we define for each image a plane $\widehat{A}_t$ passing through $\overline{\mathbf{X}}$ with normal equal to the direction of the ray $\bar{\mathbf{r}}_t$ obtained through Eq. (2) for $\bar{\mathbf{x}}_t$.

We define the plane in homogeneous coordinates as a 4-dimensional vector:

$$\widehat{A}_t = \begin{bmatrix} \bar{\mathbf{r}}_t \\ d \end{bmatrix}, \qquad (9)$$

where the last element $d = \bar{\mathbf{r}}_t^{\mathsf{T}}\overline{\mathbf{X}}$ is the distance between the camera and $\overline{\mathbf{X}}$. The projection of $\mathbf{v}_t^{(k)}$ on $\widehat{A}_t$ is the intersection $\mathbf{V}_t^{(k)}$ between the plane and the ray $\mathbf{r}_t^{(k)}$ generated from $\mathbf{v}_t^{(k)}$. We compute $\mathbf{V}_t^{(k)}$ by expressing $\mathbf{r}_t^{(k)}$ in Plücker coordinates as a line $\mathbf{L}_t^{(k)}$ joining the camera projection center $\mathbf{C}_t$ and a point $\mathbf{p} = \mathbf{C}_t + \mathbf{r}_t^{(k)}$ along the ray:

$$\mathbf{L}_t^{(k)} = \begin{bmatrix} \mathbf{L}_h \\ \mathbf{L}_0 \end{bmatrix} = \begin{bmatrix} \mathbf{C}_t - \mathbf{p} \\ \mathbf{C}_t \times \mathbf{p} \end{bmatrix}. \qquad (10)$$

From $\mathbf{L}_t^{(k)}$, we compute the transposed Plücker matrix

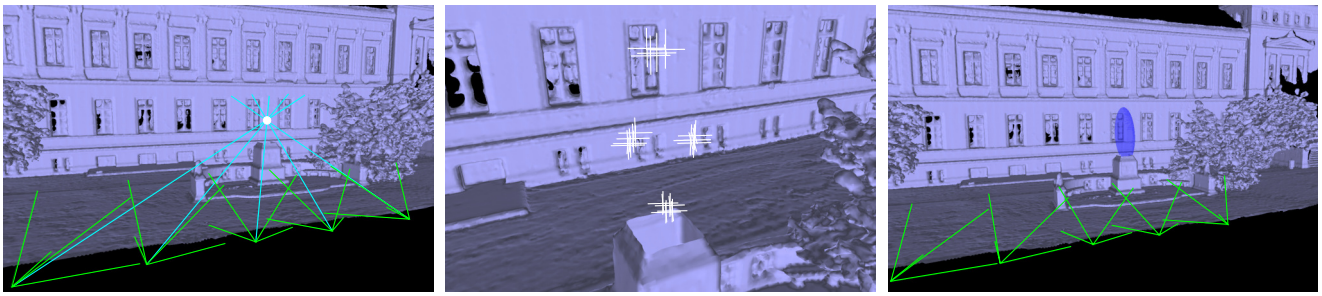
$$\mathbf{\Gamma}^{\mathsf{T}}(\mathbf{L}_t^{(k)}) \quad = \quad \begin{bmatrix} \mathsf{S}(\mathbf{L}_0) & \mathbf{L}_h \\ -\mathbf{L}_h^{\mathsf{T}} & 0 \end{bmatrix}, \qquad (11)$$

where $\mathsf{S}(\mathbf{L}_0)$ is the skew symmetric matrix corresponding to $\mathbf{L}_0$. Finally, we obtain $\mathbf{V}_t^{(k)}$ as

$$\mathbf{V}_t^{(k)} \quad = \quad \mathbf{\Gamma}^{\mathsf{T}}(\mathbf{L}_t^{(k)})\hat{A}_t. \qquad (12)$$

(a) (b) (c) (d) (e)

Fig. 5: (a) The statue (here manually marked in green) is not in the model. (b) Inconsistencies between 2 images ($m = 1$). (c) Inconsistencies between 3 images ($m = 2$). (d) Inconsistencies between 4 images ($m = 3$). (e) Original image masked with the segmented area obtained from the inconsistency image with $m = 3$. (best viewed on screen)



(a) (b) (c)

Fig. 6: (a) Example triangulation with five images. The white lines are the back-projected rays and the white point represent the triangulated point. (b) Sigma points projected in 3D. (c) The result of our algorithm, i.e. the 3D region where the change is. (best viewed in color)

We repeat this procedure for the sigma points from each mean and covariance matrix of the same region in every image. In this way, we can quickly estimate the approximate 3D location of the change without computing a dense reconstruction of the scene, see Fig. 6b. The mean and the covariance of the position of these points represent the 3D area where the change occurs, see Fig. 6c.

IV. Experimental Evaluation

The focus of this work is a comparably fast approach to identify changes in a given 3D model using a sequence of new images. Thus, our experiments are designed to show the performance of our approach and to support the two claims that we made in the beginning of the paper, i.e., our method: (i) can localize changes in the environment using a 3D model obtained in the past and a sequence of new keyframe images, and (ii) can be executed fast enough to run on an exploring robot, i.e., the average execution time should be in the order in which the sequence is recorded, here in the order of a few seconds for around five keyframe images.

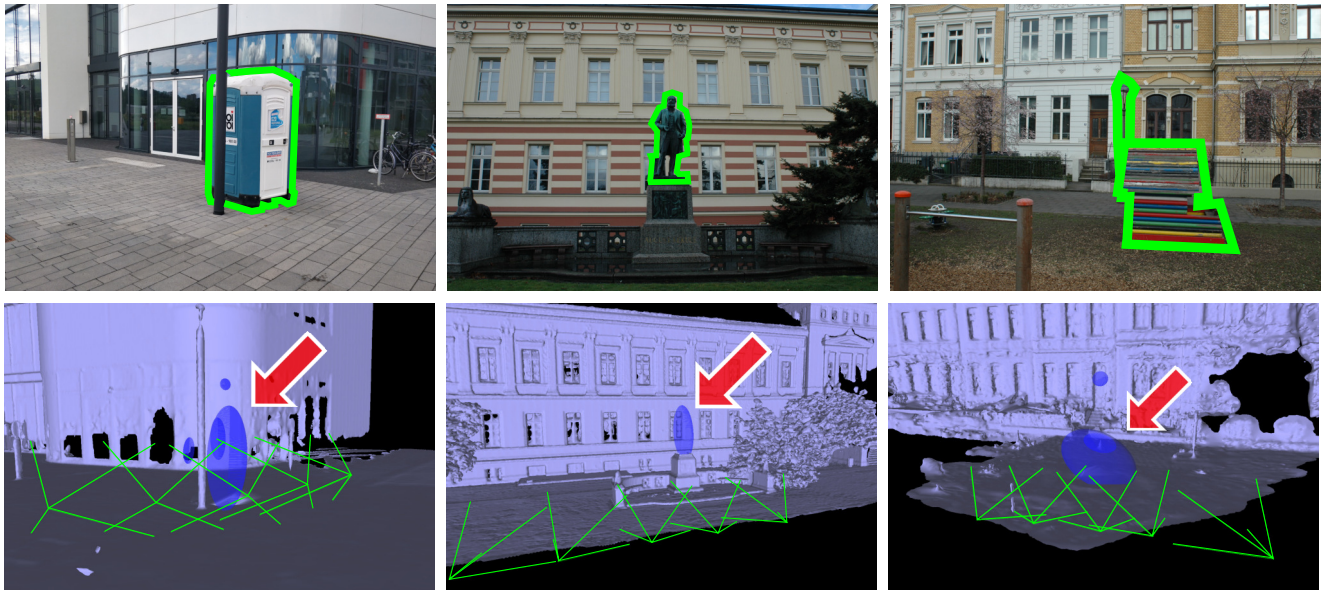
We perform the evaluations on own datasets, as well as on a subset of the ScanNet dataset by Dai et al. [3]. Furthermore, we use the dataset by Taneja et al. [20] to provide a comparison with their method. Throughout all experiments, we use a sequence of $n = 5$ images and for each image of the sequence, we compute the inconsistencies with $m = 4$ neighboring images, i.e. for these sequences all the neighboring images. Before the execution of the algorithm, we resize every image to a fixed width of 500 pixels.

A. Qualitative Evaluation

The first experiment is designed to illustrate the capability of our approach to localize a change in 3D given a model and a small sequence of images. Fig. 7 depicts the results of the algorithm on three different outdoor datasets, while Fig. 8 shows the results on three indoor scenes from the ScanNet dataset [3]. In all our tests, the localized 3D regions reflect the actual position of the changes. This information can allow an exploring or mapping robot to inspect the changed regions in more detail and collect more observations to update the previously built model. Note that the exploration itself is not part of this work but it is enabled by it.

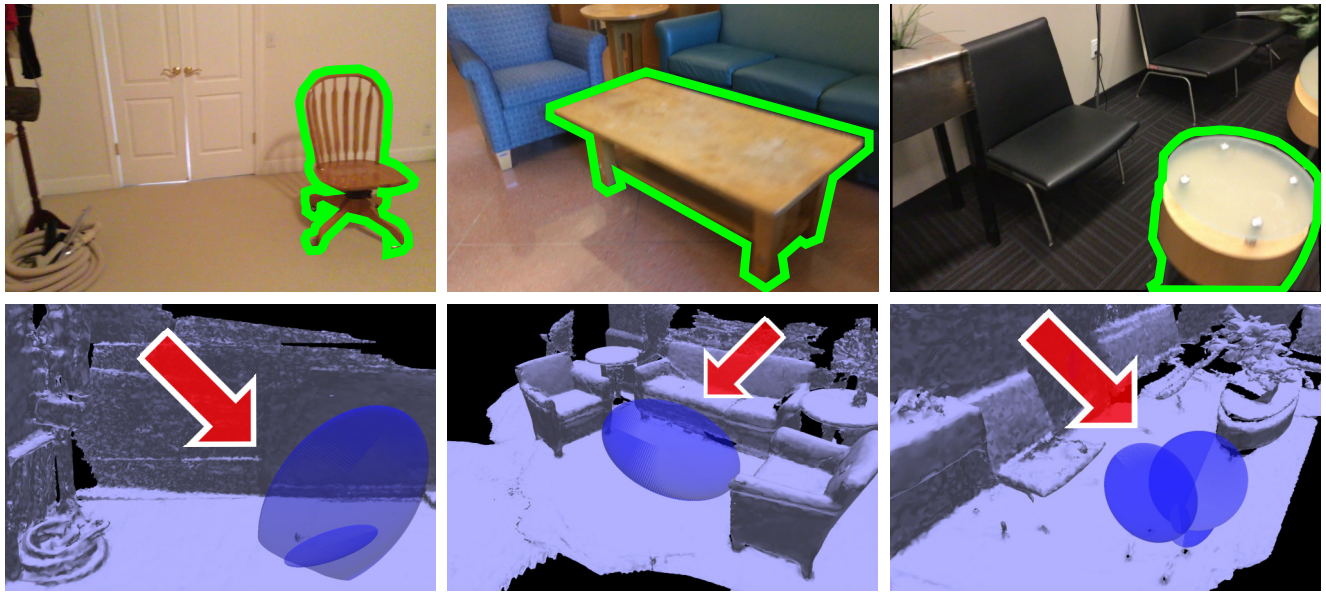
B. Quantitative Evaluation

To provide a quantitative evaluation, we project the 3D results of our approach onto the original 2D images in the sequence and we compare the 2D projection to a manually labeled ground truth. To get the final score for a dataset, we compute the average score among all the images in the sequence. We use two different evaluation criteria. The first one is the *intersection over union* (IoU), which is one of the most commonly used metrics for image segmentation [5]. The common approach is to compute the IoU using bounding boxes. However, since we are evaluating the changes projected in a sequence of images, occlusions may happen between multiple changes and that makes it impossible to evaluate the bounding box of a single change, see Fig. 9 for an example. Therefore, we compute the intersection over union directly between the segmented ground truth and the projected ellipsoid used to indicate the change in our



(a) (b) (c)

Fig. 7: Results of our experiments on three different outdoor datasets. For each dataset, the top image shows the changes (here manually marked in green), while the bottom image shows the 3D region, identified by our algorithm, where the changes are. (best viewed in color)



(a) (b) (c)

Fig. 8: Results of our experiments on three different indoor scenes from the ScanNet dataset [3]. For each dataset, the top image shows the changes (here manually marked in green), while the bottom image shows the 3D region, identified by our algorithm, where the changes are. (best viewed in color)

approach. As Fig. 10 shows, the intersection over union is not always representative of the quality of our detection. In this case, the algorithm successfully identified the change, but the IoU score is only 44%. This results from the fact that we correctly identify the location of change, but have a substantial discrepancy between the two shapes, as we only compute an ellipsoid. Thus, we additionally provide the measurement of *coverage* (or True Positive Rate) of our results, i.e. the intersecting area between our detection and the ground truth, over the full area of the ground truth. This allows us to measure whether the change is properly covered or not. In the case of Fig. 10 the coverage score is 97%, meaning that the algorithm is able to detect the change appropriately.

We tested our algorithm on five self-recorded outdoor datasets. To also consider other datasets, we additionally used the ScanNet dataset by Dai et al. [3], which provides indoor scenes recorded with an RGB-D camera. Here, we selected 15 scenes, we removed some objects from the 3D models and run our algorithm on a sequence of RGB images

(a) (b)

Fig. 9: (a) The house does not occlude the lamp. (b) The lamp is occluded: it is impossible to guess its bounding box.

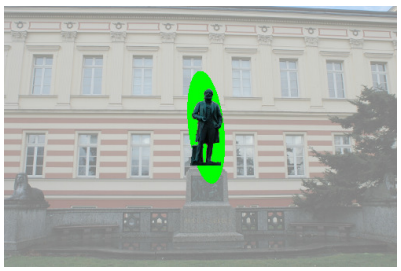
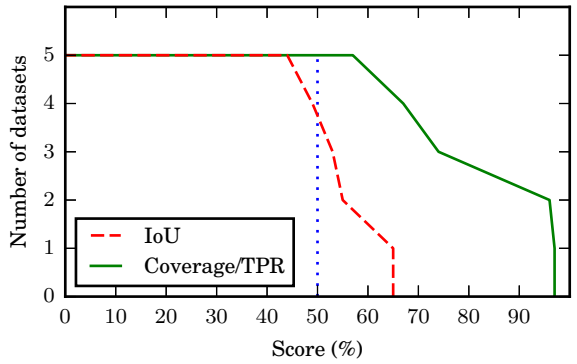


Fig. 10: Result of our algorithm projected on the original image. The statue (manually segmented from the image) is not present in the model. The green ellipse is the projection of the result of our algorithm on the image. (best viewed in color)

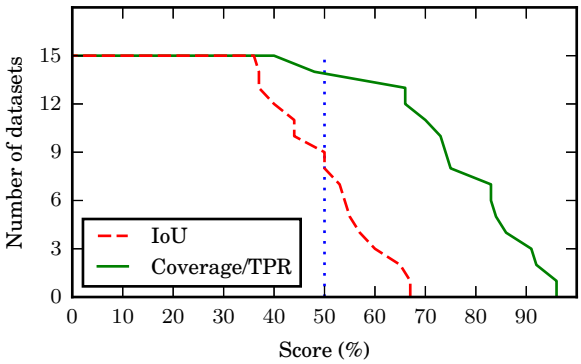
containing the missing objects. Fig. 11 shows the number of datasets (on the y axis) over a certain threshold (on the x axis). The blue dotted line is the 50% threshold, which is the one commonly used in the image segmentation literature [5]. On all our datasets and on 12 out of 15 ScanNet datasets, the IoU score is over 40%. This shows that our algorithm is generally able to detect the position of the changes, although with a significant approximation in terms of shape (given that we only compute the ellipsoid and not the exact 3D structure). The coverage score, considerably higher, further supports this claim.

C. Execution Time

The next experiment is designed to support the claim that our approach runs fast enough for processing on an exploring robot. We therefore measured the execution time of our approach on a common, lightweight laptop with an Intel Core i7 processor and an embedded Intel GPU. All the operation were executed on a single core of the CPU except for the re-projection operations, which are implemented in OpenGL and therefore executed by the integrated GPU of the Intel processor. Tab. I shows the average execution time needed to process sequences of five images from different datasets as well as the standard deviation. The numbers support our second claim, namely that the computations can be executed fast enough for operation on an exploring robot. On all datasets, the whole process takes less than 2 s, which is shorter than the time needed to record the five keyframe images. Even though the process is clearly not real-time in a strict sense, it is fast enough to be executed on a real robot at a low frequency to trigger exploration or additonal mapping actions.



(a) Evaluation on our outdoor datasets



(b) Evaluation on the indoor ScanNet datasets

Fig. 11: Quantitative evaluation of our algorithm. The plots represent the number of datasets on which our approach achieved a score above a certain threshold.

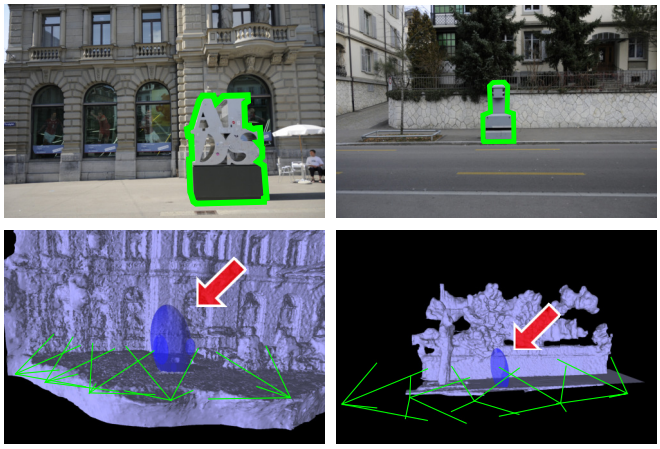
TABLE I: Average execution time for different datasets.

Dataset	Average execution time [s]
outdoor data	1.64±0.19
ScanNet (indoor)	1.95±1.08
all	1.88±0.94

D. Comparison to an Existing Approach

Finally, we want to briefly compare our results with those obtained by Taneja et al. [20]. The comparison is done based on two of the datasets that they provide and report on. We chose the "Speedcam" dataset, as it is the only one for which the authors provide the ground truth, and the "Structure" dataset, as it is the one that appears more often in their paper. We created the ground truth for the second dataset by segmenting the pictures manually, as it is not provided by the authors. Fig. 12 and Tab. II illustrate the results of our algorithm on the datasets. Their approach uses a computationally expensive graph cut labeling on a 3D voxelization of the scene. Their method typically provides a more accurate estimate of the region of change (in the order of $25\times25\times25\,\mathrm{cm}^3$ voxels) than our estimate using the mean and covariance.

The disadvantage of their method, however, is the computational demands as they require computation times in the order of 1 min per region (reported by the authors [20]), whereas we can process the same datasets in about 1 second. Thus, for most robotics applications, where an online feedback is expected, our approach is better suited.



(a) Structure (b) Speedcam

Fig. 12: Results of our approach on the datasets by Taneja et al. For each dataset, the top image shows the changes (here manually marked in green), while the bottom image shows the 3D region, identified by our algorithm, where the changes are. (best viewed in color)

TABLE II: Results for the datasets by Taneja et al.

	Structure	Speedcam
IoU	51%	52%
Coverage	86%	88%
Time [s]	1.43	1.13

To summarize, our evaluation shows that our method can estimate the 3D location of changes in the environment. At the same time, the algorithm is fast enough to be used by an exploring robot to focus on the areas that have changed.

V. Conclusion

In this paper, we presented a novel approach to identify geometric changes between the current state of the environment and a previously built 3D model using a short sequence of images. Our approach operates by identifying the changes in the images by re-projecting them onto each other, passing through the 3D model. We eliminate the ambiguities about possible changes by combining the inconsistencies from multiple pairs of images. We are then able to estimate the locations of changes in 3D and identify the changed region through a mean 3D point and a covariance matrix. The computational time of the whole process using multiple images is in the order of seconds. We implemented and evaluated our approach on different datasets. The experiments show that our method can correctly identify the changes in the environment with only five images and a total computational time of less than 2 s, which make the algorithm suitable for running on mobile robots.

Acknowledgments

We thank Johannes Schneider and Jens Behley for the fruitful discussions and valuable help during the realization of our approach. We furthermore thank Taneja et al. and Dai et al. for sharing their datasets.

References

[1] P.F. Alcantarilla, S. Stent, G. Ros, R. Arroyo, and R. Gherardi. Street-View Change Detection with Deconvolutional Networks. In *Proc. of Robotics: Science and Systems (RSS)*, 2016.

[2] T. Caselitz, B. Steder, M. Ruhnke, and W. Burgard. Monocular Camera Localization in 3D LiDAR Maps. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2016.

[3] A. Dai, A.X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner. ScanNet: Richly-Annotated 3D Reconstructions of Indoor Scenes. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[4] I. Eden and D.B. Cooper. Using 3D Line Segments for Robust and Efficient Change Detection from Multiple Noisy Images. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, pages 172–185, 2008.

[5] M. Everingham, L. Van Gool, C.K. Williams, J. Winn, and A. Zisserman. The Pascal Visual Object Classes (VOC) Challenge. *Intl. Journal of Computer Vision (IJCV)*, 88(2):303–338, 2010.

[6] W. Förstner and B. Wrobel. *Photogrammetric Computer Vision – Statistics, Geometry, Orientation and Reconstruction*. Springer Verlag, 2016.

[7] M. Golparvar-Fard, F. Pena-Mora, and S. Savarese. Monitoring Changes of 3D Building Elements from Unordered Photo Collections. In *Proc. of the Int. Conf. on Computer Vision (ICCV) Workshops*, pages 249–256, 2011.

[8] S.J. Julier and J.K. Uhlmann. A New Extension of the Kalman Filter to Nonlinear Systems. *Proc. of the SPIE Conf. on Reconnaissance and Electronic Warfare System*, 3068:182–193, 1997.

[9] E. Palazzolo and C. Stachniss. Change Detection in 3D Models Based on Camera Images. In *9th Workshop on Planning, Perception and Navigation for Intelligent Vehicles at the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2017.

[10] E. Palazzolo and C. Stachniss. Information-Driven Autonomous Exploration for a Vision-Based MAV. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, IV-2/W3:59–66, 2017.

[11] T. Pollard and J.L. Mundy. Change Detection in a 3D World. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 1–6, 2007.

[12] R. Qin and A. Gruen. 3D Change Detection at Street Level Using Mobile Laser Scanning Point Clouds and Terrestrial Images. *ISPRS Journal of Photogrammetry and Remote Sensing (JPRS)*, 90:23–35, 2014.

[13] R. Qin, J. Tian, and P. Reinartz. 3D Change Detection – Approaches and applications. *ISPRS Journal of Photogrammetry and Remote Sensing (JPRS)*, 122:41–56, 2016.

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

[15] K. Sakurada and T. Okatani. Change Detection from a Street Image Pair using CNN Features and Superpixel Segmentation. In *Proc. of British Machine Vision Conference (BMVC)*, pages 61–1, 2015.

[16] 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. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 137–144, 2013.

[17] J. Schneider, C. Eling, L. Klingbeil, H. Kuhlmann, W. Förstner, and C. Stachniss. Fast and Effective Online Pose Estimation and Mapping for UAVs. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2016.

[18] S. Stent, R. Gherardi, B. Stenger, and R. Cipolla. Detecting Change for Multi-View, Long-Term Surface Inspection. In *Proc. of British Machine Vision Conference (BMVC)*, pages 127–1, 2015.

[19] S. Suzuki and K. Abe. Topological Structural Analysis of Digitized Binary Images by Border Following. *Computer vision, graphics, and image processing*, 30(1):32–46, 1985.

[20] A. Taneja, L. Ballan, and M. Pollefeys. Image Based Detection of Geometric Changes in Urban Environments. In *Proc. of the IEEE Intl. Conf. on Computer Vision (ICCV)*, pages 2336–2343, 2011.

[21] A. Taneja, L. Ballan, and M. Pollefeys. City-Scale Change Detection in Cadastral 3D Models Using Images. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 113–120, 2013.

[22] A.O. Ulusoy and J.L. Mundy. Image-Based 4D Reconstruction Using 3D Change Detection. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, pages 31–45, 2014.