Change Detection in 3D Models Based on Camera Images

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 reprojecting an image onto the other by passing through the 3D model. Ambiguities about possible inconsistencies resulting from this process are resolved 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 tested it on an existing dataset for change detection as well as on self recorded images sequences. Our experiments suggest 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 was substantially changed, 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 can greatly reduce the required efforts. Therefore, it is important to reliably identify locations in the environment or 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 for us: 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. Our approach seeks to find changes between the current state of the world and a previously recorded, existing 3D model of the scene. For finding inconsistencies, we do not build another 3D model from the newly obtained image data. Instead, we project the currently obtained image onto the 3D model and then back to a view-point at which another image of the current sequence has been taken. Through a comparison between the back-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

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.

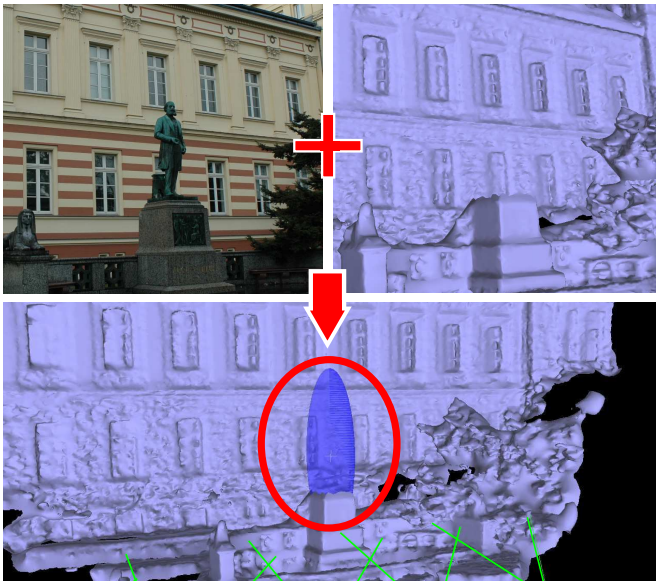


Fig. 1: Our approach aims at quickly finding changes in the environment based on an existing 3D model and a sequence of (currently recorded) images.

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. [13] or Ulusoy et al. [14], our method is substantially faster towards execution on a mobile robot.

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 back-projected images that would have been obtained assuming the 3D model is correct, in combination with a forward intersection of the potentially inconsistent regions. Our experiments suggest that our method quickly finds the approximate location of 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 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., 10 s for five keyframe images with a size of 1500 by 1000 pixels).

II. Related Work

Building 3D models can be an expensive process as it requires a good coverage of the environment 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 [8].

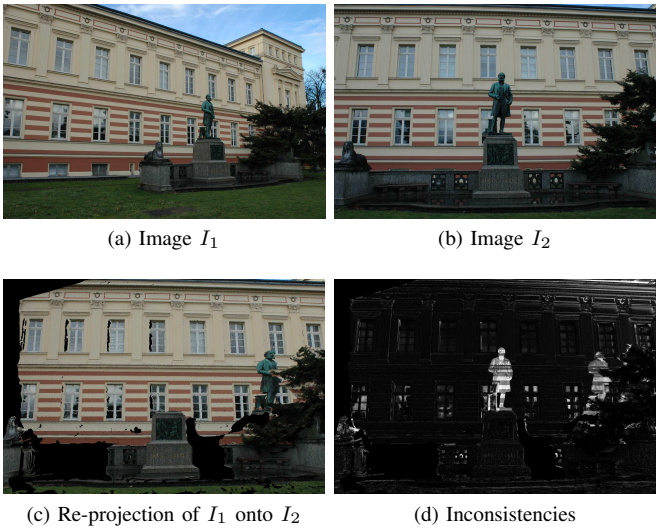
In the past, several 2D change detection algorithms have been proposed [9]. Several of such methods are affected by lighting conditions, seasonal changes, weather conditions, and other differences that may occur between the recording of the old and the new images. Moreover, the images often do not provide information on the actual 3D location of the change. Sakurada et al. [10] try to overcome these problems by estimating the probabilistic density of the depth from the oldset of images and by comparing it with the depth computed from the new set of images. Eden et al. [3] compare 3D lines in the images instead of using color or intensity information. A more recent approach by Alcantarilla et al. [1], instead use a deep convolutional neural network combined 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. This is, however, often a rather time consuming activity, at least when using cameras. Golparvar-Fard et al. [4] 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. [14] or the one by Pollard et al. [6].

Another relevant strategy that use an existing 3D model and newly acquired images is to identify changes by reprojecting images onto each other by passing through the existing model and compare the inconsistencies in the reprojection. Taneja et al. [13] use this technique on pairs of images, and apply a graph cut minimization to label the changed area in 3D in a voxelized model. In addition, Qin et al. [7] 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 reprojection technique similar to [13] and [7] to identify the changed regions in the images. We resolve ambiguities by fusiung 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 [13] or [14] have execution times in the order of minutes.



(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 reprojected onto the second, and the inconsistencies between them.

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 looks 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. Note that we assume a good pose estimate for the robot. We obtain the (approximate) location of the 3D model and the viewpoint of the images as described in Sec. III-A below. The first step is to detect possible inconsistencies of an image with its neighboring images assuming that the 3D model is correct. 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 inconsitencies 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.

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 [11]. 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 Images Pairs

To detect inconsistencies between a pair of images consisting of the images I_1 and I_2, we create a new image $I_{1\to 2}$ that represents the content of I_1 as seen from view point of I_2 given the 3D model. Given the calibration matrix and

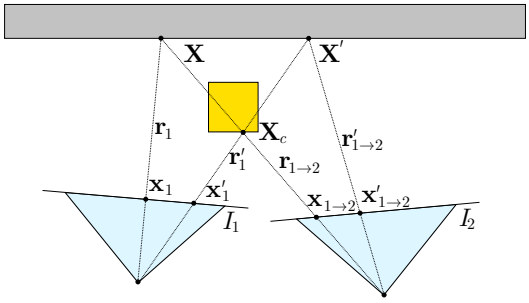


Fig. 3: Re-projection procedure. The gray rectangle represents the known 3D model, while the yellow square is a change not present in the original model. Using two images, a point $\mathbf{X}_c$, not present in the model, is reprojected onto two pixels $\mathbf{x}_{1\to2}$ and $\mathbf{x}'_{1\to2}$.

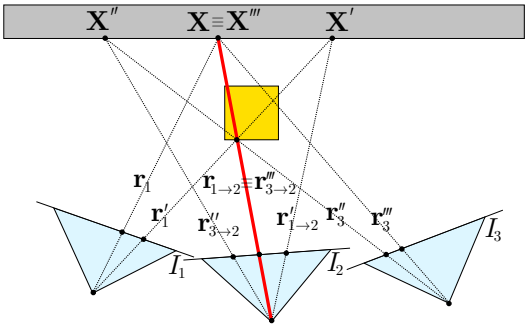


Fig. 4: Ambiguity elimination using multiple images. When re-projecting I_1 and I_3 onto I_2, only one ray (therefore one pixel) is coincident. The thicker red line represents that coincident ray.

the pose at which the camera took I_1, we can compute the projection of a 3D point $\mathbf{X}$ onto the image plane resulting in a 2D point at pixel $\mathbf{x}_1$:

$$\mathbf{x}_1 \quad = \quad \mathbf{P}_1\mathbf{X}, \tag{1}$$

where $\mathbf{x}_1$ is expressed in homogeneous coordinates and $\mathbf{P}_1 = \mathbf{K}_1[\mathbf{R}_1|-\mathbf{R}_1\mathbf{t}_1]$ is the camera projection matrix computed from the calibration matrix $\mathbf{K}_1$ of the camera and the rotation $\mathbf{R}_1$ and translation $\mathbf{t}_1$ 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_1 onto the 3D model assuming the known intrinsic parameters ($\mathbf{K}_1$) and the rotation matrix $\mathbf{R}_1$ from the extrinsic parameters:

$$\mathbf{r}_1 \quad = \quad \mathbf{R}_1^\mathsf{T}\mathbf{K}_1^{-1}\mathbf{x}_1, \tag{2}$$

where $\mathbf{r}_1$ is the direction of the ray in world coordinates.

In the next step, we project the intersections $\mathbf{X}$ between the rays and the 3D model 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 overcome this, 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 $\mathcal{N}$ 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:

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

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

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

where $d^2 = 11.82$ is the critical value of the χ^2_2 distribution corresponding to a probability of 99.73%, i.e. a 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 reproject 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)

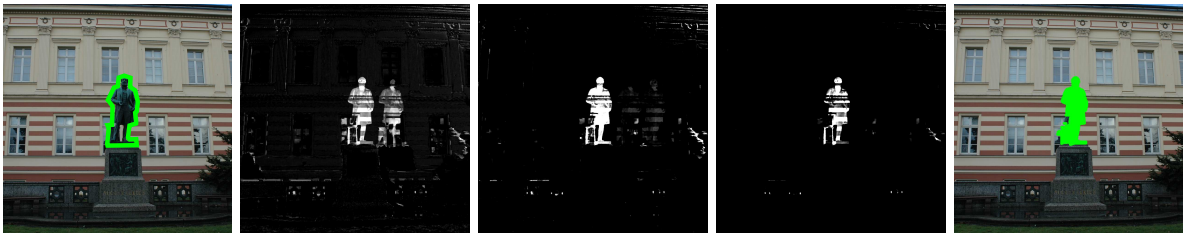
The process, however, leads to ambiguities. As Fig. 3 illustrates, a single point $\mathbf{X}_c$ corresponding to a change in the 3D model generates two pixel locations, $\mathbf{x}_{1\to2}$ and $\mathbf{x}'_{1\to2}$, in $D_{1\to2}$, one corresponding to the change in I_1 reprojected onto I_2 and one corresponding to the change in I_2 reprojected 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 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 product of all the inconsistency images obtained from the neighboring images reprojected onto I_t:

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



(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 in color)

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 $m = 1, 2, \text{and } 3$.

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 [12]. We discard all the regions with a contour shorter than a threshold (in our implementation 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. Specifically, we setup a system of equations in the form

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

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.:

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

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 (Fig. 6a). For each change in the image, we additionally compute the K sigma points [5] $\mathbf{v}_t^{(k)}$ $(k = 1 \dots K)$ 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. To compute the 3D position of the sigma points, we define for each image a plane $\mathcal{A}_t$ 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 can define the plane in homogeneous coordinates as a 4-dimensional vector:

$$\mathcal{A}_t = \begin{bmatrix} \bar{\mathbf{r}}_t \\ d \end{bmatrix}, \tag{9}$$

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 $\mathcal{A}_t$ 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:

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

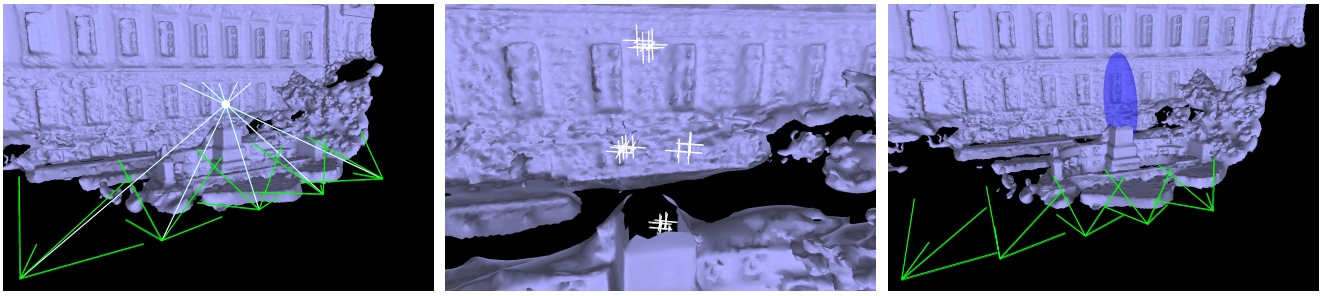
From $\mathbf{L}_t^{(k)}$, we compute the transposed Plücker matrix

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

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

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

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.



(a) (b) (c)

Fig. 6: (a) Example triangulation with 5 images. The white lines are the backprojected 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)

IV. Experimental Evaluation

The focus of this work is a comparably fast approach to identify changes in a previously obtained 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 central claims that we made in the beginning of the paper, i.e. that 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 5 keyframe images.

We perform the evaluations on own datasets, which we publicly share including the 3D models at http://www.ipb.uni-bonn.de/data/changedetection2017/, as well as the dataset used by Taneja et al. [13], which can be obtained from: https://cvg.ethz.ch/research/change-detection/Datasets/Structure.zip. Throughout all experiments, we use a sequence of $n = 5$ images and for each image of the sequence, we compute the inconsistencies with $m = 4$, i.e. for these sequences all the neighboring images. We found out that using higher values of m does not improve the results substantially and that is why we used this value on all the experiments.

A. Change Identification

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 4 different datasets. 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 this works enables it.

The "Playground" dataset shown in Fig. 7c is particularly challenging. In this dataset, the house, which is not present in the model, is composed by separate wooden pieces, each one in a different color. Our algorithm is able to correctly identify the lamp and the bar as changes, but recognizes the house as multiple, separate changes. This does not constitute a real problem, but shows a possible limitation of our approach.

TABLE I: Execution time for different datasets. The images in our datasets have resolution 1504×1000 pixels, the ones by Taneja et al. [13] have resolution 1072×712 pixels.

Dataset name	Execution time without uncertainty [s]	Execution time with uncertainty [s]
A/C Unit	4.598	10.54
Statue	7.486	12.571
Playground	8.529	13.989
Taneja et al. [13]	2.47	5.525
Average time	5.77±2.758	10.656±3.702

B. Execution time

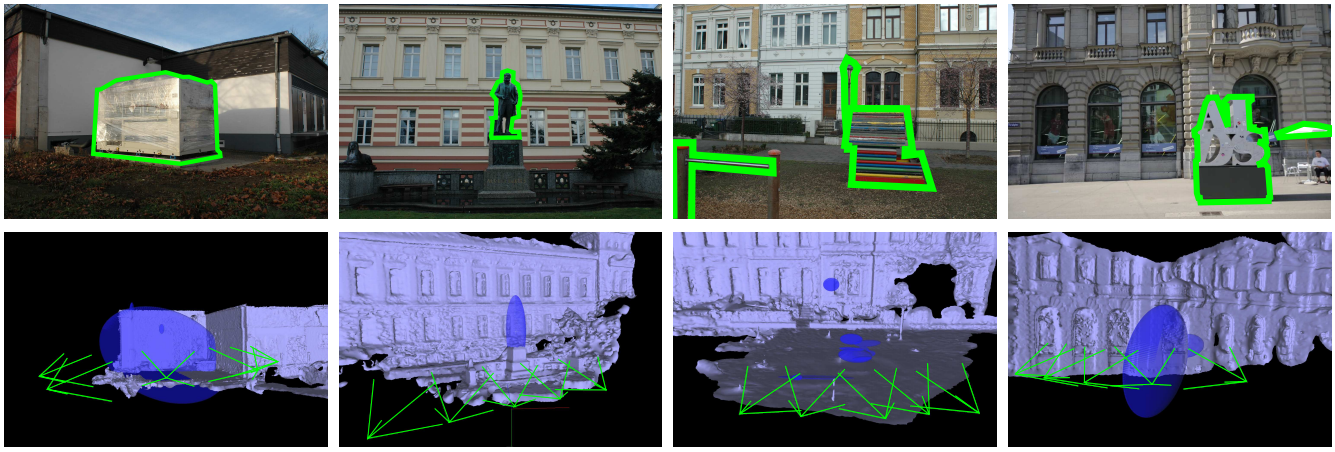
The next experiment is designed to support the claim that our approach runs fast enough for processing on an exploring robot. We therefore measured the execution time of our approach on a common, lightweight laptop with an Intel Core i7 processor and an embedded Intel GPU.

Tab. I shows the average execution time needed to process sequences of 5 images from different datasets as well as the standard deviation, both with and without taking into account the uncertainty on the camera poses. The numbers support our second claim, namely that the computations can be executed fast enough for operation on an exploring robot. On our datasets, the whole process, taking into account the uncertainties, takes about 10 s, which is shorter than the time needed to record the 5 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.

The computation time is influenced by both the number of images as well as their resolution. This is evident from our test on the dataset by Taneja et al., which took approximatively half of the time for processing images with a lower resolution.

C. Comparison to an Existing Approach

Finally, we want to briefly compare our results with those obtained by Tanjea et al. [13]. The comparison is done based on the dataset that they provide and report on (Fig. 7d). 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



(a) A/C Unit (b) Statue (c) Playground (d) Dataset by Taneja et al.

Fig. 7: Results of our experiments on 4 different 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)

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, whereas we can process the same dataset in about 5 seconds. Thus, for most robotics applications, where an online feedback is expected, our approach is better suited.

To summarize, our evaluation suggests that our method can estimate the 3D localization 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. Thus, we supported all our claims made in the introduction with this experimental evaluation.

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 reprojecting 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 suggest that our method can correctly identify the changes in the environment with only 5 images and a total computational time of around 10 s, which make the algorithm suitable for running on mobile robots.

As future work, we plan to conduct more effective tests of our method on different types of datasets and extend the quantitative comparisons.

Acknowledgments

We thank Johannes Schneider and Jens Behley for the fruitful discussions and valuable help during the realization of our approach. We furthermore thank Tanjea et al. for sharing their dataset from Zurich.

References

[1] P.F. Alcantarilla, S. Stent, G. Ros, R. Arroyo, and R. Gherardi. Streetview 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 Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 1926–1931. IEEE, 2016.

[3] 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.

[4] 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.

[5] 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.

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

[7] 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*, 90:23–35, 2014.

[8] R. Qin, J. Tian, and P. Reinartz. 3D change detection – Approaches and applications. *ISPRS Journal of Photogrammetry and Remote Sensing*, 122:41–56, 2016.

[9] R.J. Radke, S. Andra, O. Al-Kofahi, and B. Roysam. Image change detection algorithms: a systematic survey. *IEEE Transaction on Image Processing*, 14(3):294–307, 2005.

[10] 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.

[11] 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 Int. Conf. on Robotics & Automation (ICRA)*, pages 4784–4791, 2016.

[12] 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.

[13] A. Taneja, L. Ballan, and M. Pollefeys. Image based detection of geometric changes in urban environments. In *Proc. of the Int. Conf. on Computer Vision (ICCV)*, pages 2336–2343, 2011.

[14] A.O. Ulusoy and J.L. Mundy. Image-based 4-d reconstruction using 3-d change detection. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, pages 31–45, 2014.