Improving Multi-view Object Recognition by Detecting Changes in Point Clouds

Martin Velas[1], Thomas Fäulhammer[2], Michal Spanel[1], Michael Zillich[2], Markus Vincze[2]

***Abstract*—This paper proposes the use of change detection in a multi-view object recognition system in order to improve its flexibility and effectiveness in dynamic environments. Multi-view recognition approaches are essential to overcome problems related to clutter, occlusion or camera noise, but the existing systems usually assume a static environment. The presence of dynamic objects raises another issue – the inconsistencies introduced to the internal scene model. We show that by incorporating the change detection and correction of the inherent scene inconsistencies, we reduce false positive detections by 70% in average for moving objects when tested on the publicly available TUW dataset. To reduce time required for verifying a large set of accumulated object pose hypotheses, we further integrate a clustering approach into the original multi-view object recognition system and show that this reduces computation time by 16%.**

I. INTRODUCTION

Industrial as well as service robot applications include the tasks of detecting changes in the environment while navigating, recognizing objects and estimating the object pose for grasping or similar tasks. In recent years, the accessibility of reliable depth sensors, mainly RGB-D cameras, contributed to significantly improve the 3D localization precision of objects detected in the environment.

Nevertheless, several challenges remain. For example, the standard techniques of object recognition, e.g., [1], [2], often fail to detect the objects correctly when objects are partially occluded, the camera view is not favorable or cannot be altered (which is often the case for mobile robots) or when the environment is cluttered. To overcome these problems, recent works [3], [4], [5], [6] exploited techniques combining multiple observations of the same scene from different viewpoints. Since these works assume a static environment, there naturally arises the problem of false detections in environments with moving objects (Fig.1). Removal of objects from the scene is not reflected in the internal model (e.g., scene reconstruction) or the model is not flexible to cope with changes. In this paper, we address the issue of a flexible reconstruction and update of a dynamic scene model using change detection in point clouds.

The research was funded by the TACR project V3C (no. TE01020415), the IT4IXS – IT4Innovations Excellence in Science project (LQ1602), the European Community's Seventh Framework Programme FP7/2007-2013 (No. 600623), STRANDS and the AKTION cooperation project 72p7.

[1]Department of Computer Graphics and Multimedia, Faculty of Information Technology, Brno University of Technology, Czech Republic {ivelas|spanel} at fit.vutbr.cz

[2]Vision4Robotics group (ACIN), Faculty of Electrical Engineering and Information Technology, Vienna University of Technology, Austria {faeulhammer|zillich|vincze} at acin.tuwien.ac.at

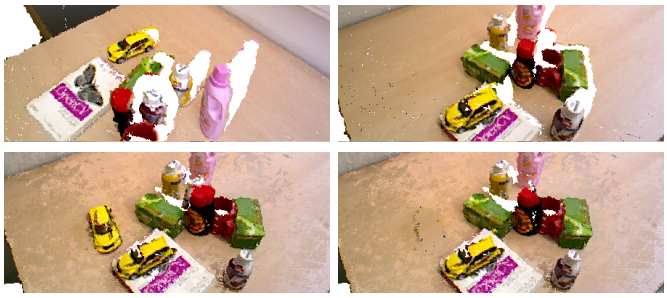


Fig. 1: Two views of a scene (top row) observed in time from left to right. In between the observations an object (yellow toy car) is moved in position. The bottom row shows the reconstructed 3D scene after the two observations, subsequently used as input for object recognition. In [6], the scene incorrectly contains two object instances of the yellow toy car in the reconstructed scene (bottom left). Using our proposed change detection module, only the instance in the current position is preserved (bottom right).

We show how this can be used to build a reliable multi-view 3D object recognition pipeline for mobile robots.

The key contributions are summarized as follows: (i) extension of the change detection method proposed by [7] for non-omnidirectional RGB-D sensors, (ii) improved precision of a multi-view object recognition system in changing environments by incorporating the change detection module, (iii) reduced complexity of the object verification stage by clustering object hypotheses, and (iv) publicly available source code and an extended RGB-D dataset including annotations of objects and changes in the scene.

The paper proceeds as follows. After reviewing related work regarding multiview object recognition and change detection, we present the proposed system in Section III with object recognition and change detection modules deeply elaborated. The impact of our work is evaluated in Section IV and concluded in the final section.

II. RELATED WORK

Detection of changes in 3D scenes proved to be useful in many robotic applications. Although there is a lot of literature related to object retrieval such as unsupervised object discovery, to our knowledge, change detection has not been applied to improve object recognition yet.

A. Multi-view object recognition

While there is a vast amount of literature related to object recognition (see [8] for an extensive review), only a small

fraction of it deals with multiple viewpoints of the scene.

Pillai and Leonard [5] proposed a multi-view object recognition approach on top of a monocular SLAM system. Using a reconstruction of the observed environment, they cluster points based on density generating object candidates described by efficient feature encodings. They then classify these candidates in each view and infer the most probable object class across multiple views. Lai *et al.* [4] build a voxel grid for 3D reconstruction of the scene from multiple RGB-D frames. They classify each point in the scene by two separate classifiers (a sliding window detector in each RGB-D image and hierarchical matching pursuit for 3D voxel data). Their responses are then combined using a Markov Random Field.

Collet *et al.* [9] uses a similar approach to our work as they (i) generate object hypotheses from clustered feature correspondences within each single images, (ii) merge similar hypotheses across views and (iii) refine their pose by an Iterative Clustering Estimation. Our work is different in following aspects. First, we use a multi-pipeline recognizer [10] for extracting features of multiple modalities (both the shape and the appearance). Next, we use a graph-based correspondence grouping which is computationally more complex but eases the detection of multiple object instances appearing next to each other. Furthermore, we reject potential false positives by an additional hypotheses verification [11].

All of the approaches mentioned above assume an environment where objects are *static*. Our work particularly addresses situations where this is not the case (e.g. objects move, appear or disappear). The multi-view method in [12] projects object candidates generated from immediate observations into the currently observed RGB-D frame and tries to cope with dynamic changes by an additional global verification step which checks for outliers and other cues [11]. While this rejects false object detections, the verification (i) does not exploit the full information available from multiple views, and (ii) has to check all possible configurations of accumulated object candidates which becomes computationally very expensive. In this work, we verify the accumulated hypotheses not only against a single observation of the current scene but against a fully reconstructed 3D scene considering all available viewpoints.

B. Change detection

Mason *et al.* [13] exploited information of object disappearance for automatic and unsupervised object discovery and tracking. To detect disappearing objects, they built a sparse feature map by employing visual features (ORB) over RGB-D data. Missing feature points in new observations are clustered and removed objects are discovered by supporting plane based segmentation. Herbst *et al.* [14] discover objects unsupervised by building a dense surfel-based scene representation and comparing two states of the scene. To reconstruct and compare the scene, a probabilistic model over the dense surfel-based map is adopted. Although such system has proved to successfully find moving objects especially in the clear tabletop scenes, it is computationally quite expensive.

A continuous object segmentation for dynamic environments was published by Finman *et al.*[15]. Objects discovered by the change detection are used as a training data for unsupervised learning of the segmentation algorithm. Since both the map and the observation are processed in the form of raw point clouds, change detection is based on *point cloud differencing* (described in III-B1). The authors of the *Meta-rooms* system [7] proposed an iterative environment scanning from predefined vantage points. Once the autonomous robot had scanned the environment (usually office, room, storage, etc.), a new observation is compared to the meta-room by change detection also adopting a point cloud difference. Discovered objects are described by VFH features and clustered into the classes without any supervision.

An important aspect in detecting changing objects is the way of reasoning about occlusion. Since there is no evidence about the presence of occluded objects, these objects can not be considered as a change. While [15] ray-traces voxel grid from the position of camera looking for occlusions in occupied voxels, approaches [16] and [7] use reasoning in the spherical coordinates. Since its superior computational performance achieved by efficient spherical structure, we decided to base our work on [7] and relax the assumption of omnidirectional scans of the scene (see details in III-B1).

In this work, we show that integration of this proposed change detection into an existing multi-view recognition system [6] significantly improves recognition of objects appearing and/or changing position in-between different scene observations. Since there is (to our best knowledge) no related method able to cope with dynamic scenes in terms of multi-view recognition, we evaluate this proposed solution with respect to the original recognition system to demonstrate the improvement.

III. SYSTEM DESCRIPTION

Our goal is to detect the identity o and pose $\boldsymbol{P}$ of pre-trained object models in a test scene $\boldsymbol{S}$. The scene is hereby observed from multiple view points k and represented by a set of point clouds $\mathcal{S} = \{\boldsymbol{S}_k\}$ sensed by an RGB-D camera (ASUS Xtion Primesense used in our experiments). Each of these views potentially contains one or multiple instances of the objects which can (dis-)appear and/or change position during the observation period K. For each view, we assume the camera pose is known (e.g. by tracking the camera with Structure-from-Motion techniques like [17] or using common object hypotheses [6]) which allows us to integrate the observations into a common coordinate system. The overall recognition system used to achieve this goal is shown in Fig.2 where novel components are highlighted in red. The following sections describe the individual components in more detail.

A. Multi-view object recognition

Our multi-view recognition system *(MVR-CHD)* is based on the existing framework *(MVR)* [6] briefly explained in this section. In particular, we divide the steps in training the object models, generating object hypotheses $\hat{\mathcal{H}}_k$ in each test view,

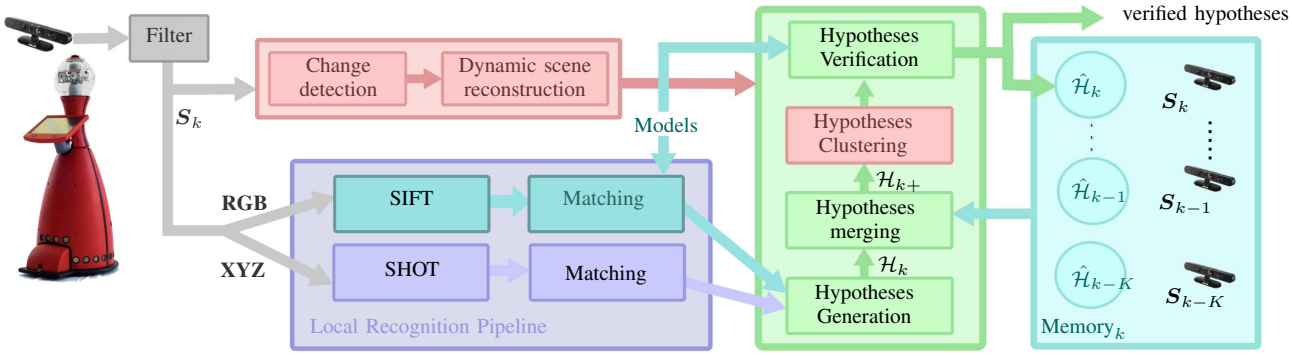


Fig. 2: Overview of our multi-view object recognition system with the proposed change detection and hypotheses clustering (red boxes). The scene $\boldsymbol{S}_k$ is observed by an RGB-D sensor at different points in time k. A local recognition pipeline (purple box) extracts local features and matches them with pre-trained features from the object models. Hypotheses $\hat{\mathcal{H}}_k$ are generated from the corresponding keypoints of these feature matches and accumulated over multiple scene observations into $\mathcal{H}_{k+}$. Finally, these hypotheses are clustered and verified against a scene reconstruction from the last K observations with dynamic segments already removed.

merging these hypotheses into a common coordinate system $\hat{\mathcal{H}}_{k+}$ and verifying a subset to give the final result.

In the first step, we *train* models for each object of interest using [17]. While the object is presented from 360° to the camera, this approach tracks the position of the object and extracts training views after significant viewpoint change (15° in our experiments). The training stage outputs (i) a registered and organized RGB-D point clouds and (ii) a set of keypoints with associated feature descriptors. We use two different local features; uniformly sampled SHOT [2] describing the geometrical traits and the SIFT [1] extracted in DoG keypoints capturing the appearance.

To *generate object hypotheses*, a multi-pipeline recognizer [11] extracts features from the current test view $\boldsymbol{S}_k$ and matches them to the corresponding model features by a k-Nearest Neighbor search in both feature spaces. To allow detecting multiple object instances within a scene, keypoint correspondences between model and scene are stored as nodes within a graph and connected to each other if they are geometrically consistent, i.e. if point distance and respective surface normals hold consensus [11]. This allows to compute cliques and for each clique we estimate a 6DoF rigid body transformation $\boldsymbol{P} = [\boldsymbol{R}|\boldsymbol{t}]$ aligning scene and model keypoints. The output is a set of object hypotheses $\hat{\mathcal{H}}_k = \left\{o_k^j, \boldsymbol{P}_k^j\right\}$ with j indexing the detected objects.

To exploit the information gain from observing the scene from multiple view points, we *merge* all object hypotheses generated for the last K test views into a common set $\hat{\mathcal{H}}_{k+}$. Given we know the camera pose to each test view, we are able to transfer generated object hypotheses from previous test views into a merged hypotheses set $\hat{\mathcal{H}}_{k+}$. At this end, the set contains all objects detected in the last K views which elements can be highly redundant. To reduce the computational cost of final verification stage which grows with the number of hypotheses, we additionally *cluster* nearby hypotheses based on their position and orientation. In particular, starting with randomly selected seed object hypotheses $h^j \in \hat{\mathcal{H}}_{k+}$ we iteratively cluster hypotheses $h^i \in \hat{\mathcal{H}}_{k+}$ *iff*

$$o^j = o^i \quad \text{and} \quad \|\boldsymbol{t}^j - \boldsymbol{t}^i\| < \delta_t \quad \text{and} \quad \boldsymbol{R}^{j\to i}(\alpha), \boldsymbol{R}^{j\to i}(\beta), \boldsymbol{R}^{j\to i}(\gamma) < \delta_r, \tag{1}$$

where $\boldsymbol{R}^{j\to i} = \boldsymbol{R}^j \boldsymbol{R}^{i^T}$ gives the relative orientation of the two hypotheses (hypothesis i with respect to j) and $\boldsymbol{R}(\alpha)$, $\boldsymbol{R}(\beta)$ and $\boldsymbol{R}(\gamma)$ are yaw, pitch and roll angles of rotation matrix $\boldsymbol{R}$. The threshold parameters δ_t and δ_r define the maximum allowed relative distance and orientation for two object hypotheses to be clustered together; they influence computation time and accuracy. As a trade-off between these two, we empirically chose $\delta_t = 2$cm and $\delta_r = 10°$. We additionally refine the pose of each clustered object hypothesis by a final RANSAC based rigid body transformation estimation which considers all keypoint correspondences within a cluster that were used for generating the individual hypotheses.

Finally, we *verify* the set of clustered object hypotheses $\hat{\mathcal{H}}_{k+}$ against the registered point clouds of the last K test views by a global optimization function [18]. It tries to maximize the number of *visible* model points that can be explained by nearby scene points and vice versa. At the same time it tries to penalize false detections by geometrical cues described in detail in [18]. Different to [18] which assumes a static scene, this work only checks nearby points for test views where the object is indicated as present by our change detection method influencing the set of *explained* model and scene points. The optimization function is solved by local search and outputs the final set of object hypotheses.

B. Incorporating change detection into recognition system

The approach used in *MVR* [6] assumes a static environment and verifies generated object hypotheses against a scene reconstructed from all points from previous observations. As the scene reconstruction gets corrupted for environments changing during the observation period K (see Fig. 1 for instance), this method tends to either falsely detect objects disappearing or (due to an underestimated ratio of explained points) to falsely

reject objects appearing during this observation period. To address these issues, we integrate a change detection module which is based on the *Meta-Rooms* framework [7]. Since this work assumes omnidirectional observations and in our work we have to deal with general RGB-D camera positions, we additionally include a *view frustum test* described in the following section.

Given the previous scene reconstruction $\boldsymbol{\Psi}_{k-1}$ for the observations $\boldsymbol{S}_{k-1}, \ldots, \boldsymbol{S}_{k-K}$, we find changes with respect to each current scene observation $\boldsymbol{S}_k$ as a set of *novel* $\boldsymbol{N}$ and *removed* $\boldsymbol{D}$ points. These changes are taken into account for reconstructing the current 3D scene $\boldsymbol{\Psi}_k$. The change detection and reconstruction algorithm is summarized in Alg. 1 and described in detail in the following sections.

1) Change detection: The goal of our change detection is the retrieval of novel $\boldsymbol{N}$ and removed $\boldsymbol{D}$ points as shown step by step in Fig. 3. Analogously to [7], we first compute the raw difference between the current scene reconstruction and the new observation. The difference of two point clouds $\boldsymbol{A}$ and $\boldsymbol{B}$ is defined by

$$\boldsymbol{A} \setminus \boldsymbol{B} = \{a \in \boldsymbol{A} \,|\, \forall b \in \boldsymbol{B} : \|a - b\| > \epsilon\}. \tag{2}$$

Assuming all points are transformed into a common coordinate system, we define novel points $\boldsymbol{N}$ and estimate the initial set of removed points $\boldsymbol{D}''$ by

$$\boldsymbol{N} = \boldsymbol{S}_k \setminus \boldsymbol{\Psi}_{k-1} \qquad \boldsymbol{D}'' = \boldsymbol{\Psi}_{k-1} \setminus \boldsymbol{S}_k. \tag{3}$$

Since the initial estimation $\boldsymbol{D}''$ also contains points occluded in the current frame $\boldsymbol{S}_k$, potentially causing false detections of changes (see Fig.3c), we deploy an additional *occlusion test* to deal with such cases.

As in [7], we reason about occlusion of points using a *spherical z-buffer* (Fig.4). Without loss of generalization, we assume the camera is positioned at the origin of the coordinate system (this can be always arranged since the camera position is known) and transform each point $\boldsymbol{p} = [p_x, p_y, p_z]$ into spherical coordinates $[p_\phi, p_\theta, p_d]$ with

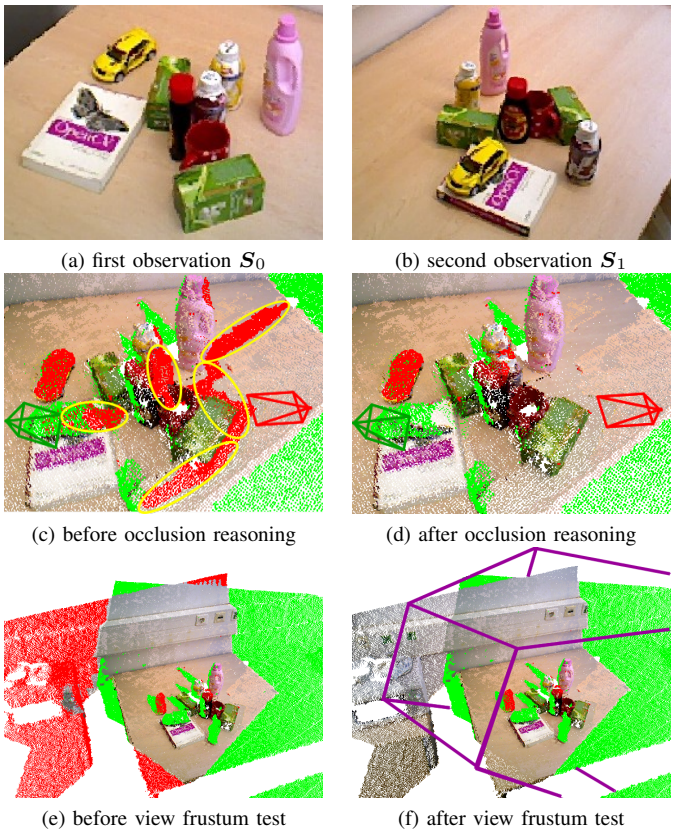
$$p_d = \|\boldsymbol{p}\|, \quad p_\theta = \pi + \text{acos}(p_z/p_d), \quad p_\phi = \pi + \text{atan}(p_y/p_x). \tag{4}$$

The spherical coordinate system is divided into regular spherical bins (see Fig.4). In our experiments, we used bins of size 2° in both θ and ϕ directions.

To find occluded points in cloud $\boldsymbol{A}$ with respect to another cloud $\boldsymbol{B}$ (the one which causes the occlusion), we assign each point into a bin. A point $\boldsymbol{q} \in \boldsymbol{A}$ is then considered *occluded* if $p_d < q_d$ for each point $\boldsymbol{p} \in \boldsymbol{B}$ sharing the same bin as $\boldsymbol{q}$. In our case, points obtained by the previous scene reconstruction difference $\boldsymbol{\Psi}_{k-1} \setminus \boldsymbol{S}_k$ can be occluded by newly observed points $\boldsymbol{S}_k$. Such points can not be considered as changed.

Similar to the case of occlusion, change detection can not reason about points outside the camera's field of view. Therefore, we add a *view-frustum test* to check for each point whether it is within the camera's view frustum (see Fig. 3f and step 3 of Alg. 1). Given the camera intrinsic and extrinsic parameters $\boldsymbol{C}_k$ (position $\boldsymbol{t}$, orientation $\boldsymbol{R}$, horizontal α_h and vertical α_v field of view, minimal z_{min} and maximal z_{max} view distance), the point cloud can be transformed such that the camera is at the origin of the coordinate system and facing parallel with the z-axis by multiplying each point by $[\boldsymbol{R}|\boldsymbol{t}]^{-1}$. A transformed point is within the current field of view *iff*

$$\left(\frac{|p_x|}{\sqrt{p_x^2 + p_z^2}} < \alpha_h\right) \wedge \left(\frac{|p_y|}{\sqrt{p_y^2 + p_z^2}} < \alpha_v\right) \wedge p_z \in (z_{\text{min}}, z_{\text{max}}). \tag{5}$$



(a) first observation $\boldsymbol{S}_0$ (b) second observation $\boldsymbol{S}_1$

(c) before occlusion reasoning (d) after occlusion reasoning

(e) before view frustum test (f) after view frustum test

Fig. 3: A dynamic scene observed from two different viewpoints (a),(b) with their respective camera position depicted as red and green frustums in (c),(d). The point cloud difference is shown in (c) with *removed* points marked red and *novel* points green. Points occluded in the new viewpoint are incorrectly marked as removed (yellow ellipses). These false change detections are excluded by the occlusion test (d). The view frustum test shown in (e), (f) excludes points outside the camera view frustum (purple).

2) Scene reconstruction: In the original system, each point in $\mathcal{S}$ is associated with a noise term which estimates the lateral and axial noise level according to Nguyen *et al.* [19] as well as the distance to the closest depth discontinuity. In a nutshell, the noise term is large for points far away from the sensor's origin or principal axis or close to any depth discontinuities. The scene is then reconstructed by putting all points in a voxel grid structure and only selecting the points with the lowest noise term in each voxel.

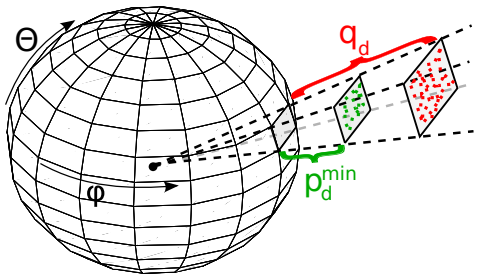


Fig. 4: Occlusion reasoning using a spherical z-buffer. First, the newly observed points (green) are transformed into spherical coordinates with its origin located at the current camera position. For each spherical bin, the minimal distance p_d^{min} is stored. Then distance q_d of each potentially removed point (red) is compared against the z-buffer value of the respective bin. If $p_d^{min} < q_d$, a point from the previous frame is occluded and preserved for the subsequent reconstruction.

Algorithm 1 Dynamic scene reconstruction

Input: Observations $\mathcal{S} = \{\boldsymbol{S}_k, \boldsymbol{S}_{k-1}, \ldots, \boldsymbol{S}_{k-K}\}$, previous reconstruction $\boldsymbol{\Psi_{k-1}}$ and current camera parameters $\boldsymbol{C_k}$
Output: Current scene reconstruction $\boldsymbol{\Psi_k}$

1: $\boldsymbol{D''} := \boldsymbol{\Psi_{k-1}} \setminus \boldsymbol{S}_k$
2: $\boldsymbol{D'} :=$ NOTOCCLUDED($\boldsymbol{D''}$, $\boldsymbol{S}_k$)
3: $\boldsymbol{D} :=$ INVIEWVOLUMEOF($\boldsymbol{D'}$, P_k^c)
4: **for** $i = k, k-1, \ldots, k-K$ **do**
5: $\quad \boldsymbol{S}_i := \boldsymbol{S}_i \setminus \boldsymbol{D}$
6: $\boldsymbol{\Psi_k} :=$ NOISEMODELBASEDINTEGRATION($\mathcal{S}$)

After the changes have been identified, the most straightforward way of using them for the reconstruction improvement would be their direct removal from the reconstruction. However, to ensure that the detected changes will be reflected in the assignment of the noise terms to the respective points, the filtering of removed points is done individually for each frame before modeling the noise. The solution used in this work is summarized in lines $4-6$ of Alg.1. It processes each observation individually and discards all points which has been marked as removed by the change detection. (Since the implementation of noise model we used requires organized point clouds, the points are only set to *NaN* values).

IV. EVALUATION

We evaluated the impact of the change detection described in previous chapters on an extended TUW dynamic dataset[3]. The dataset consists of 9 sequences (203 keyframes in total) of captured office environments with objects of everyday use (bottles, books, toys, ...). Some of these objects (dis-)appear or are moved in-between observations which causes the aforementioned issues in state-of-the-art multi-view recognition systems. Using [18], we annotated all objects with their 6DoF pose in each RGB-D view. Moreover, we have developed publicly available tools for annotation of changes in the presence of object instance[4].

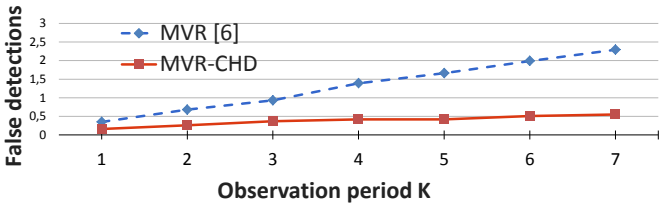


Fig. 5: Average number of false detections caused by the change (removal or move) of the object instance. Both the original MVR and the improved MVR-CHD system were evaluated for different values of K (number of previous scene observations used for recognition).

	MVR [6]			**MVR-CHD**		
K	Precision	Recall	F-score	Precision	Recall	F-score
0	0.93	0.50	0.65	0.93	0.50	0.65
1	0.89	0.66	0.76	**0.90**	**0.66**	**0.77**
2	0.86	0.67	0.76	**0.89**	**0.68**	**0.77**
3	0.82	0.68	0.74	**0.87**	**0.68**	**0.76**
4	0.78	0.67	0.72	**0.84**	**0.68**	**0.75**
5	0.76	0.67	0.71	**0.84**	**0.67**	**0.75**
6	0.74	0.65	0.69	**0.84**	**0.67**	**0.75**
7	0.73	0.65	0.68	**0.84**	**0.68**	**0.75**

TABLE I: Overall results of the object recognition in the dynamic scenes with and without change detection for different observation periods K. Significant improvement can be observed in terms of precision for multiple views, since the change detection module reduces number of false positives. It also illustrates the benefits of a multi-view recognition system compared to a single-view approach $(K = 0)$. In a single-view system, the change detection reasoning can not bring any improvement.

A. Integration of change detection

A qualitative example of a scene reconstructed by the original (MVR) system and by our proposed system incorporating change detection (MVR-CHD) is shown in Fig. 6.

The following experiment quantitatively evaluates the effect of the change detection module on the recognition of objects disappearing from the scene or being moved to another position. In particular, we are interested in the number of views the object is falsely detected at its original position after the change. On average, incorporating change detection decreases false detections of objects which disappeared or were moved to a different position in the scene by 70%.

To demonstrate the effect of our proposed change detection module on the overall results, we evaluated precision and recall on the extended TUW dataset. As summarized in Table I, the change detection improves both precision and recall for all values of K (observation period). On average, there is a 8.3% improvement in precision, 1.4% in recall and 4.3% in f-score (i.e., harmonic average of precision and recall).

[3] repo.acin.tuwien.ac.at/tmp/permanent/dataset_index.php

[4] github.com/martin-velas/v4r

Fig. 6: Multiple observations of a scene observed in time from left to right with an object (yellow toy car) being moved twice within the sequence (top). Initial and new position of the object are marked red and green, respectively. The static environment assumed by the MVR [6] framework (middle row), incorrectly reconstructs the scene such that points from the object's original position remain present during the whole observation period K (number of frames used) which was set to 3 in this experiment. Therefore, it took four observations for invalid points of the moved object to be discarded. Using our proposed change detection (bottom), the scene is correctly reconstruced at each time step. It removes points belonging to the old position of the object and keeps all points observed during the observation period that belong to the new object's position or are static.

The improvement of the recognition performance averaged over the whole dataset (Table I) is not as significant as evaluated on the dynamic objects only (Fig. 5). This is because many of the object instances are static (or at least temporarily static) in the dataset. For these static objects, the recognition performance is, as expected, the same with and without change detection.

Since the false detection rate increases with the observation period (the removed object is falsely detected for a longer time), the precision drops accordingly. This is significantly reduced in MVR-CHD but can not be totally eliminated. For cases, when the location of a previously removed object is not well observed afterwards (e.g. because of occlusions), the reasoning about the changes and the object presence can not be done.

B. Time performance

Regarding the computational cost, the change detection adds only minor overhead. In average, the recognition of original MVR system took 28.6s per frame, comparing to 29.8s per frame for MVR-CHD system. The computational cost was estimated using whole TUW dynamic dataset and the hypotheses were accumulated over $K = 5$ previous views.

The most demanding part of the recognition pipeline is hypotheses verification module, which is optimized by hypotheses clustering evaluated in the next chapter.

C. Clustering of object hypotheses

As shown in Table II and III our proposed clustering method significantly reduces the computation time of the verification stage at approximately the same overall recognition rate. In fact, the average f-score is even increased by 1%. This can be explained by the lower dimensional problem that needs to be solved during the global hypotheses verification which decreases the chance the local search gets stuck in an unfavourable local minima (details can be found in [11]). The additional computation time needed to cluster the object hypotheses is on average just a fraction (6%) of the time saved for verifying the object hypotheses. The trend of the computation time needed for hypotheses verification over time is shown in Fig.7. In this experiment we accumulate object hypotheses over $K = 5$ views. Therefore, the computation time does not significantly increase after the fifth view.

	Precision	Recall	F-score
w/o clustering [6], [11]	0.71	0.68	0.69
with clustering	0.72	0.69	0.70

TABLE II: Recognition rate with and without hypotheses clustering.

clustering time	216ms
preserved object hypotheses after clustering	66.0%
reduction of verification time	3.6s (15.7%)

TABLE III: Influence of our clustering approach on the hypotheses verification ($\delta_d = 2cm$, $\delta_r = 10°$) averaged over all sequences in the TUW dataset.

V. CONCLUSION

In this paper we propose improvements over the existing multi-view object recognition system that are able to deal with both the occlusion of objects and with dynamic objects in the scene. The improvements have been mainly achieved by integration of the change detection module, playing a role of "dynamic segments filtering" in the scene observations. After this filtering, a consistent scene reconstruction is built and used for the verification of hypotheses. The result is a significant drop of the false positives rate for dynamic objects by 70%. This has been proved by the experimental evaluation using a newly collected and annotated dataset that is publicly

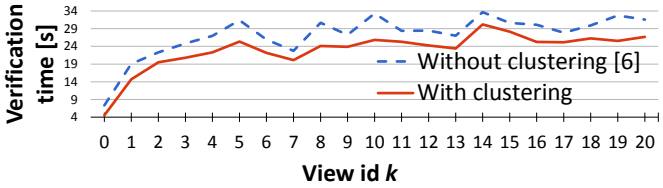


Fig. 7: Computation time of the hypotheses verification with and without clustering of object hypotheses for frames of single data sequence.

available together with source code of the implementation. Moreover, the novel hypotheses clustering module has been also introduced into the system to reduce the hypotheses verification time by 16%.

Future research should attempt a deeper incorporation of the change detection method into the mobile robot navigation and scene reconstruction scheme to fully benefit from its potential to speed-up the recognition process. Those areas of the scene where novelties are discovered will be considered as salient regions such that computational effort is concentrated and computing time significantly reduced.

REFERENCES

[1] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *IJCV*, vol. 60, no. 2, pp. 91–110, Nov. 2004.

[2] F. Tombari, S. Salti, and L. Di Stefano, "Unique signatures of Histograms for local surface description," in *ECCV*, 2010.

[3] A. Collet and S. Srinivasa, "Efficient multi-view object recognition and full pose estimation," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, May 2010, pp. 2050–2055.

[4] K. Lai, L. Bo, X. Ren, and D. Fox, "Detection-based object labeling in 3d scenes," in *ICRA*. IEEE, 2012.

[5] S. Pillai and J. Leonard, "Monocular slam supported object recognition," *arXiv preprint arXiv:1506.01732*, 2015.

[6] T. Fäulhammer, M. Zillich, and M. Vincze, "Multi-view hypotheses transfer for enhanced object recognition in clutter," in *IAPR Conference on Machine Vision Applications (MVA)*, 2015.

[7] R. Ambrus, N. Bore, J. Folkesson, and P. Jensfelt, "Meta-rooms: Building and maintaining long term spatial models in a dynamic world," in *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, Sept 2014, pp. 1854–1861.

[8] Y. Guo, M. Bennamoun, F. Sohel, M. Lu, and J. Wan, "3d object recognition in cluttered scenes with local surface features: A survey," *PAMI*, vol. 36, no. 11, pp. 2270–2287, 2014.

[9] A. Collet, M. Martinez, and S. S. Srinivasa, "The moped framework: Object recognition and pose estimation for manipulation," *The International Journal of Robotics Research*, vol. 30, no. 10, pp. 1284–1306, 2011.

[10] A. Aldoma, F. Tombari, L. Di Stefano, and M. Vincze, "A global hypothesis verification method for 3d object recognition," in *European Conference on Computer Vision (ECCV)*, 2012.

[11] A. Aldoma, F. Tombari, L. Di Stefano, and M. Vincze, "A global hypothesis verification framework for 3d object recognition in clutter," *PAMI*, 2015.

[12] T. Fäulhammer, A. Aldoma, M. Zillich, and M. Vincze, "Temporal integration of feature correspondences for enhanced recognition in cluttered and dynamic environments," in *ICRA*. IEEE, 2015.

[13] J. Mason, B. Marthi, and R. Parr, "Object disappearance for object discovery," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, Oct 2012, pp. 2836–2843.

[14] E. Herbst, P. Henry, X. Ren, and D. Fox, "Toward object discovery and modeling via 3-d scene comparison," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, May 2011, pp. 2623–2629.

[15] R. Finman, T. Whelan, M. Kaess, and J. Leonard, "Toward lifelong object segmentation from change detection in dense rgb-d maps," in *Mobile Robots (ECMR), 2013 European Conference on*, Sept 2013, pp. 178–185.

[16] J. Underwood, D. Gillsjo, T. Bailey, and V. Vlaskine, "Explicit 3d change detection using ray-tracing in spherical coordinates," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, May 2013, pp. 4735–4741.

[17] J. Prankl, A. Aldoma, A. Svejda, and M. Vincze, "Rgb-d object modelling for object recognition and tracking," in *IROS*. IEEE, 2015.

[18] A. Aldoma, T. Fäulhammer, and M. Vincze, "Automation of ground truth annotation for multi-view RGB-D object instance recognition datasets," in *IROS*. IEEE, 2014.

[19] C. V. Nguyen, S. Izadi, and D. Lovell, "Modeling kinect sensor noise for improved 3d reconstruction and tracking." in *3DIMPVT*. IEEE, 2012, pp. 524–530.