Unsupervised learning of spatial-temporal models of objects in a long-term autonomy scenario

Rares Ambrus[1], Johan Ekekrantz[1], John Folkesson[1] and Patric Jensfelt[1]

***Abstract*— We present a novel method for clustering segmented dynamic parts of indoor RGB-D scenes across repeated observations by performing an analysis of their spatial-temporal distributions. We segment areas of interest in the scene using scene differencing for change detection. We extend the Meta-Room method and evaluate the performance on a complex dataset acquired autonomously by a mobile robot over a period of 30 days. We use an initial clustering method to group the segmented parts based on appearance and shape, and we further combine the clusters we obtain by analyzing their spatial-temporal behaviors. We show that using the spatial-temporal information further increases the matching accuracy.**

I. Introduction

Autonomous mapping is becoming a mature technology. Most approaches to date assume that the environment is static and build a model to support localization. Dynamics are often dealt with as noise to be filtered out, i.e. something to get rid of. If robots are to shoulder the tasks that the future holds for them, they need to do more than localize in static environments. The dynamics of space carry a lot of information which will need to be studied, modeled and exploited. The highly dynamic parts, such as humans, can be studied relatively easy. Many other objects in our environment move much slower and often not at all while the robot is observing but rather between observations and sometimes very seldom. Studying these dynamics requires running experiments over weeks or months but this is needed for a robot to become truly autonomous. All objects cannot be known in advance. They must be learned by the robot from experience and separating dynamic parts of the environment is a very useful tool for this.

Once dynamic parts have been identified they can be matched over time and space to develop more complete models of the parts. For example, filling in initially unobserved sections of the part, or modeling its movements over space and or time, or trying to classify the part as a known object class. In other words, identifying these parts opens up a world of self-learning possibilities for robots operating over extended periods of time.

Spatial-temporal models of dynamic elements are a higher level representation of the world that can be used for a variety of tasks such as predicting where objects are likely to be at various points in time for the purpose of finding objects or activity planning, or as input to an object recognition pipeline that could also take into account the spatial-temporal features. Moreover, such representations would help facilitate great data reduction. Instead of storing the raw data for each instance of an element, the robot could simply store one complete model of the element and its poses over time.

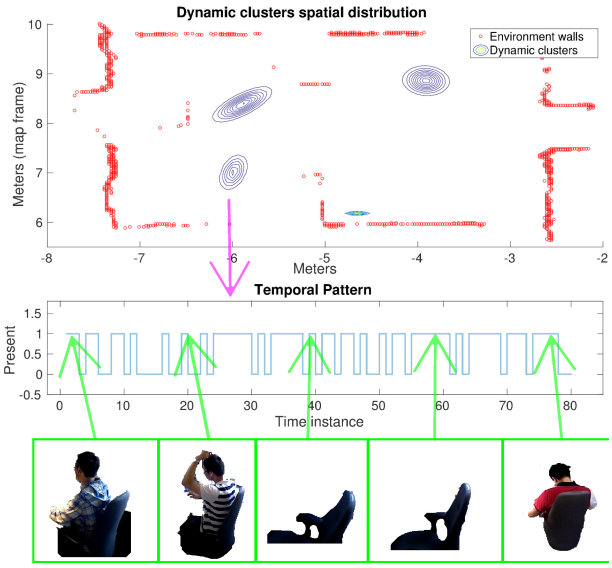

Fig. 1: Modeling the spatial distribution of dynamic elements and matching them across time. Top - environment outline (red) and spatial distributions of four dynamic elements (blue). Middle - Temporal pattern for one of the elements that is matched across time showing when it is has been detected. Bottom - Snapshots of the segmented element at different points in time.

The problem of matching the dynamic elements across observations collected at different points in time is inherently hard. The data is collected at different times during the day, and even during the night. The dynamic elements are usually seen in different relative poses meaning different sections of the element will be seen at different times leading to very small overlap between observations; for an example see the bottom row of images in Fig. 1. Dynamic elements may be composed of several separately dynamic parts, a stack of books for example. Some dynamic elements can change shape, like a laptop or may even be truly non-rigid like a coat.

In this paper we build on [1] where a method for modelling the static part of the environment based on repeated observations was introduced. Preliminary results showed that the static model could be used to extract dynamic elements from the scene. In this paper, we investigate this further and look at segmentation of elements and clustering of such elements

[1] The authors are all with the Centre for Autonomous Systems at KTH Royal Institute of Technology, Stockholm, SE-100 44, Sweden. {raambrus}@kth.se

across time. We address a subset of the challenges mentioned above and assume that the dynamic objects are rigid. Figure 1 illustrates some basic concepts. The upper part shows a 2D slice of the environment with the red outline representing what is static. The ellipses show a Gaussian estimate of the spatial distribution of four dynamic elements. The middle part shows when a certain element has been observed and in the bottom we see five example observations of the dynamic element, a chair. Notice how the chair both changes pose and sometimes includes a person.

Our contributions are:

- improved registration to allow segmentation of dynamic elements in more challenging scenarios which is quantified with real data from the robot.
- clustering of repeated observations of dynamic elements over time by exploiting spatial-temporal constraints.
- a large dataset with repeated observations over 30 days[1].

II. Related work

Dynamics in the context of mapping has been studied along several directions in the literature. Grid maps are extended to model the dynamics of occupancy in [2], [3]. Bieber and Duckett use several sample grid maps to capture the changes in the environment at different rates and time scales [4]. Walcott-Bryant et al. [5] maintain two separate maps - active and dynamic - to better represent a changing environment with a Pose-Graph SLAM system. Modeling spatial-temporal dynamics in the environment has been used successfully to improve robot localization in long term autonomy scenarios by Krajnik et al. [6]. The SLAM++ system of Moreno et al. [7] uses an a-priori constructed object database both for object recognition and for camera localization. Similarly, Li et al. [8] use a database of object models which replace segmented objects in the scene, with an emphasis on maintaining and matching against large databases of objects. In our work we focus on the individual dynamic elements and we make no assumptions about them being known ahead of time.

Gunther et al. [9] build large scale 3D maps augmented with recognized objects. Mason et al. [10] perform experiments over extended periods of time like us and build semantic maps for object query as well as change detection. They study objects lying on top of tables and other planar surfaces and the matching of these objects is done on the basis of the overlap between their convex hulls in 2D. In our work we do not require any assumptions regarding geometry of space but focus on objects that are dynamic and use that via scene differencing as the queue for segmentation.

The scene differencing technique has been used successfully by other researchers as well. In [11] Herbst et al. use scene differencing and surface patches to segment out shapes on table tops. In [12], they present a method that clusters the detected shapes into object clusters using 2D features (kernel features) and ICP as similarity measures which are then clustered using multiclass spectral clustering. In [13] they show that additional methods, such as a human moving an object can be used in conjunction with scene differencing to further improve the segmentation and classification of objects. Finman et al. [14] use change detection to segment objects in a long term autonomy scenario, and they show how the segmentation can be improved by re-identifying the segmented objects multiple times over the lifetime of the robot. Many feature representations exist in the literature within the context of re-identifying objects extracted through various methods from RGB- D data, and substantial research has been invested in coming up with better descriptors that handle occlusions or partial data - e.g. Aldoma et al [15] or Tombari et al. [16]. We also deal with partial observations of objects collected under very different circumstances. However we only rely on features extracted over the point cloud for an initial clustering phase. Contrary to previous work, we look at the spatial and temporal behavior of the elements we segment to form the clusters. We have access to this information as our data is captured over very long time periods, and we show that by using it we can improve the classification accuracy over only point cloud features.

Other work that make no or little assumptions about the environment and that do not require any training or previously defined data include Endres et al. [17]. They employ a variant of the spin-image local feature extracted from the input point cloud in conjunction with Latent Dirichlet Allocation to autonomously discover and classify object classes by modeling the distribution of the features in different object classes. Shin et al. [18] segment the input point cloud using a superpixel segmentation algorithm and match groups of segments using a modified version of the branch and bound algorithm. Their method however relies on the fact that the scene contains multiple instances of an object, as a prerequisite for that object to be segmented. Mattausch et al. [19] segment the input point cloud into planar patches and define a patch similarity measure that takes into account geometric properties of the patch as well as spatial relationships with other patches, and which is used to cluster the patches in a higher dimensionality Euclidean space. They show that their method scales well with very large amounts of data. We also segment objects in an unsupervised fashion and without requiring any training data as input. Our work differs in that it focuses on building models of individual objects observed multiple times over extended time rather than trying to find multiple instances of the same type of object.

III. System Overview

At predefined waypoints we incrementally update a 3D model of the static structure of the environment through the Meta-Room method presented in [1]. For completeness we briefly describe the method from [1] here: given a number of observations collected at a waypoint over some time period, we iteratively estimate the static structure of the environment, called the ”Meta-Room”, by removing what is dynamic from one observation to the next and at the same time adding

[1]The dataset can be found on-line at: https://strands.pdc.kth.se/public/KTH-longterm-dataset/

what was previously occluded. Changes to the static structure such as the addition of furniture are also taken into account. Examples of typical observations collected at one of the waypoints can be seen in Fig. 2.

Fig. 2: Three observations collected at the same waypoint

Each new observation is then compared to the constructed Meta-Room to segment out dynamic elements from the scene. We then match segmented elements across observations at a particular waypoint. The main difficulty in achieving this arises from the complex nature of the segmented elements, being seen from different viewpoints, under different illumination or at night, only partially visible, etc.

Our matching algorithm has two main stages: first we match elements using point features - this gives rise to sets of matched elements. Some sets correspond to the same physical object detected under such different conditions that the point features were not enough to group the elements together in one set. Second, we reason about sets of elements, trying to understand whether they represent the same physical object, by taking into account their temporal distributions, as well as calculating a similarity measure of their spatial distributions.

IV. Dynamic Element Detection/Segmentation

In [1] the focus was on reconstructing the static structure, and convergence of the method. Here we focus on the segmentation of dynamic elements, and propose two improvements that increase the accuracy of the segmentation.

A. Registration

The registration process has two stages. First, we register the individual RGB-D frames making up an observation to form a single, dense point cloud of a whole scene. Second, we register observations acquired at the same waypoint with each other, so that they are in the same frame of reference and can be compared. In [1] the Iterative Closest Point (ICP) [20] was used to register individual frames, and the Normal Distribution Transform (NDT) [21] to register observations together. Each pair of frames were registered individually, even though the motion of the pan-tilt unit is always the same in all the observations. This results in small alignment errors between frames which does not affect the creation of the static model, but which nevertheless leads to inaccuracies and noise when segmenting dynamic elements.

We propose here a different registration method which exploits the fact that the pan-tilt unit always executes exactly the same motion, even if this motion is known only roughly in advance (explained in more detail in section VI). One can say that we use all our collected data to calibrate this motion. That is, given that the relative poses between the frames are fixed we can propagate the constraints of all frames into the same estimation. For the collected dataset this is frames from over 700 observations leading to a very accurate registration. The registration is based on sparse bundle adjustment of ORB [22] keypoints augmented with depth values. Keypoints are matched between frames using the visual descriptors and then filtered using RANSAC [23]. In order to improve the results the matches are then filtered using ICP [20]. The optimization engine used is the Ceres Solver [24]. After the individual frames have been registered, the relative poses of the observations are found using the same technique of keypoint matching using RANSAC and ICP for filtering as above.

B. Dealing with low-dynamics changes

In [1] a sliding window approach was used for incorporating new elements (e.g. new furniture) into the static structure. We propose here a new formulation of this algorithm using a Bayesian approach that explicitly takes into account the temporal aspect of our data:

$$P(dynamic|\tau) \propto P(\tau|dynamic)P(dynamic)$$

where τ represents the amount of time the element has not moved. The factor $P(\tau|dynamic)$, the probability that an element has not moved for τ given that the element is dynamic, is modelled as a Poisson distribution, while $P(dynamic)$ is the prior probability that an element is dynamic and is estimated using a heuristic. The flatter an element is the more likely it is part of some large piece of furniture or the walls, and hence the lower the probability that it is dynamic. If the probability, $P(dynamic|\tau)$, exceeds a threshold we classify the element as dynamic, else we conclude it is static and add it to the static structure. Thus for each element our model gives a duration of time that the element must not move for it to be considered static.

V. Clustering Dynamic Elements

In this section we focus on autonomously separating the detected dynamic elements into clusters corresponding to the same physical structure seen at different poses over time. In our data there are multiple instances of the same type of object, and thus we want to recognize that elements belong to a particular repeated structure, but also to be able to separate multiple instances of the same structure. This allows us to build, for each such instance, a spatial-temporal map of when the instance was seen and in what pose. We first cluster the dynamic elements based on appearance and second merge the clusters based on a spatial-temporal analysis of their behavior.

A. Initial clustering

The initial clustering is done primarily through features extracted over the point cloud making up the dynamic element. The nature of our environment is such that the objects corresponding to the dynamic elements detected move in fairly constrained spaces (e.g. an office chair moves in a certain space around a desk, a laptop or a monitor moves roughly around the same position on desks). Moreover, observations collected at a particular waypoint are captured from roughly the same position, thus motivating our choice of a global, viewpoint dependent feature to describe the dynamic elements detected - the Visual Feature Histogram (VFH) [25].

Here we must reject a fair number of true positives as a result of the difficult nature of the data we are working with (partial views with very little overlap and severe changes in illumination). We are matching elements based on appearance only at this step. As we will merge these clusters in the next step false positives are more of a problem than false negatives. The result after this stage is clusters of elements which are similar in appearance, and multiple such clusters could correspond to the same physical object, seen in different circumstances.

Algorithm 1 Initial Clustering

```
1:  procedure INITIAL(allEl, newEl, clMap)
2:      for e ∈ elements do
3:          matches ← getKNearestNeighbours(e, allEl)
4:          matchFound ← False
5:          while (matches ≠ ∅) do
6:              m ← findClosestElement(matches, e)
7:              matches.pop(m)
8:              if poseMatch(m, e) then
9:                  clMap[e] ← clMap[m]
10:                 matchFound ← True
11:                 break
12:             end if
13:         end while
14:         if not matchFound then
15:             clMap[e] ← e
16:         end if
17:     end for
18:     allEl.push_back(newEl)
19: end procedure
```

The initial clustering algorithm is described in Alg. 1; we execute it once for each new observation acquired at a waypoint. The input to the initial clustering algorithm consists of all the detected dynamic elements from previous observations (variable *allEl*), the newly detected dynamic elements (variable *newEl*) and a map (variable *clMap*) that keeps track of which elements belong to which cluster. We define the cluster C_i as the set of elements c from the map such that:

$$C_i = \{\, c \in clMap \,|\, clMap[c] = i \,\}$$

For each new dynamic element, we find K nearest neighbors, in terms of the Chi-square distance of their VFH descriptors. We do not limit ourselves to just one nearest neighbor due to the fact that VFH sometimes returns wrong matches and also our data contains multiple instances of the same object, in which case we would want to pick the closest one. We select from the nearest neighbors the closest one spatially to the new dynamic element and we perform a spatial alignment step using ICP - this helps discard false positives, as our matching criteria is a fitness measure that considers two-way alignment. If a match is found we update the map to reflect the fact that the current dynamic element belongs to the same cluster as the matching dynamic element (line 9 of Alg. 1), otherwise we initialize a new cluster for the current dynamic element.

The initial clustering step is based on state of the art methods for matching features extracted over point clouds with the purpose of object recognition / clustering (see [15] [16]). In the results section we show that such methods are successful only up to an extent, but that it is possible to increase the clustering accuracy using spatial-temporal information about the elements.

B. Spatial temporal clustering

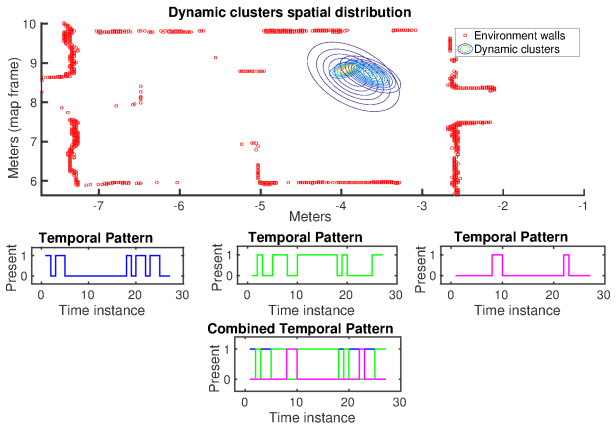

Fig. 3: The spatial and temporal distributions of three dynamic element clusters obtained after initial clustering. The bottom row shows the temporal patterns of the three clusters rendered on the same graph.

After the initial clustering step, we are left with a number of clusters of dynamic elements, with some corresponding to the same physical object. Note that at this stage, if two clusters represent the same physical object, it means that they represent two different embodiments of that object (i.e. seen under different lighting conditions or in different orientations), otherwise the initial clustering step based on visual features would have grouped all their elements together. The next step is to identify which clusters represent the same object and to group them together. The underlying assumptions are that clusters representing the same object will define the same spatial distribution, i.e. they will be found in the same place(s), and if two clusters represent

the same physical object, none of their elements should overlap temporally, i.e. in that case they must represent two different objects. Assuming a cluster is made up of n dynamic elements d_i with centroids c_i, we model the spatial distribution of the n dynamic elements making up a cluster as a Gaussian with mean and covariance:

$$\mu = \frac{1}{n}\sum_{i=1}^{n} c_i$$

$$\Sigma = \frac{1}{n-1}\left([c_i - \mu][c_i - \mu]^T\right)$$

Fig. 3 shows an example of the spatial distributions of three clusters of dynamic elements that correspond to the same physical object, along with the outline of the environment where the observations were taken. Given probability distributions P and Q, the Kullback Liebler Divergence denoted as $D_{KL}(P\|Q)$ is a measure of the information lost when Q is used to approximate P. However, since the KL Divergence is non-symmetric, we use the Symmetric Kullback Liebler Divergence $D_{SKL}(P\|Q) = D_{KL}(P\|Q) + D_{KL}(Q\|P)$ to measure similarity and dissimilarity at the spatial distribution level between the clusters. To compute the KL Divergence between two normal distributions $N_0(\mu_0, \Sigma_0)$ and $N_1(\mu_1, \Sigma_1)$ we use:

$$\begin{aligned} D_{KL}(N_0\|N_1) \quad = \quad & \frac{1}{2} tr\left(\Sigma_1^{-1}\Sigma_0\right) \\ & + (\mu_1 - \mu_0)^T \Sigma_1^{-1} (\mu_1 - \mu_0) \\ & - k + ln\left(\frac{\det\Sigma_1}{\det\Sigma_0}\right) \end{aligned}$$

Using the SKL Divergence we find pairs of compatible clusters, from a spatial distribution point of view, as those clusters D_i, D_j for which $SKL(D_i\|D_j) < \delta$, where δ is empirically chosen such that only clusters with very similar spatial distributions match the criterion. Given two clusters D_i and D_j that fit the previously defined criterion, we conclude they represent the same physical object if the individual dynamic elements making up the clusters do not overlap temporally, i.e.:

$$Match(D_i, D_j) \iff \forall d_k \in D_i, \forall d_l \in D_j\, (time(d_k) \neq time(d_l))$$

The middle row of Fig 3 shows the temporal patterns of three clusters of dynamic elements which represent the same physical object, and in the bottom row of Fig. 3 one can see that when rendered on the same graph, the temporal patterns of the three clusters do not overlap.

During the spatial-temporal clustering phase, we iteratively group pairs of clusters that match spatially and temporally, according to the criteria described above; we stop when no such pairs exist any more. In the results section we quantify the true and false positive rates of both the initial clustering and the spatial-temporal clustering on an individual object basis, as compared with labelled data.

VI. Experimental setup

The data was collected autonomously by a Scitos G5 robot with an RGB-D camera on a pan-tilt unit (PTU), navigating through our office environment over a period of approximately 30 days. Each observation consists of a set of 51 RGB-D images obtained by moving the pan-tilt in a pattern, in increments of 20 degrees horizontally and 25 vertically. We have chosen this specific pattern for good coverage of the environment around the robot, however any such configuration is possible. The requirement here is that initially the PTU visits each one of the 51 positions when acquiring a new observation. The 51 camera poses are optimized after executing the registration step described in section IV-A. Once the camera poses are known, the PTU does not need to stop at each of the 51 positions and a subset of these can be covered during the acquisition of subsequent observations.

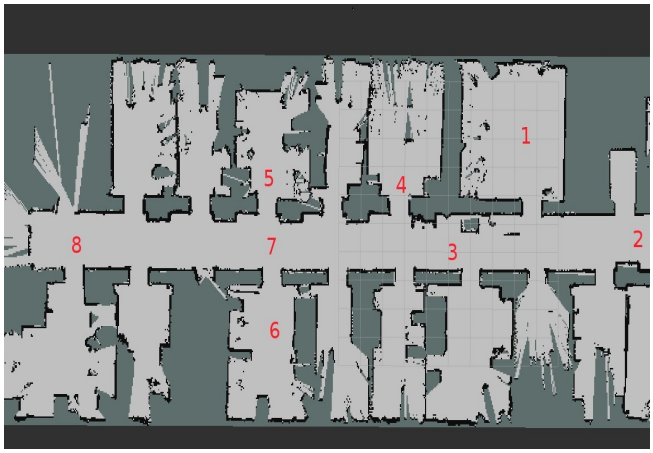

Fig. 4: 2D map with the waypoints at which the data was collected

Waypoints are visited between 80 and 100 times and a total of approximately 720 observations are collected at the eight waypoints that can be seen in Fig. 4. The robot typically starts operating at 9 AM and finishes at 7 PM every weekday, and patrol runs are usually scheduled every hour. However, depending on factors such as doors being open or closed, or other tasks with higher priority being scheduled instead, not all the waypoints are visited during each patrol run. This leads to a temporally non-uniform set of observations. We make available the data collected (images, poses and the results of our registration algorithm) for easier use and/or comparison.

To evaluate the segmentation and matching algorithms presented in this paper we have manually segmented and labelled the observations collected at one of the waypoints, using the GrabCut algorithm described in [26]. Since our data was collected at different times during the day and sometimes during the night as well, we have labelled objects both on the depth and the RGB images (roughly 3400 images for 100 observations collected at the waypoint).

VII. Results

We perform two evaluations, one of the segmentation of dynamic elements and one of clustering of dynamics elements.

A. Dynamic element segmentation

In this section we compare the performance on segmentation of dynamic elements in a scene against the original Meta-Room method [1]. We use the manually segmented and labelled elements as ground truth for the evaluation. The results are shown in Fig. 5 and Fig 6. Note that we are not showing the standard precision vs recall plot. Instead we show the precision and recall of the detected dynamic elements for each observation. In this way one can see the variation in the values that go into the averages shown in Table I.

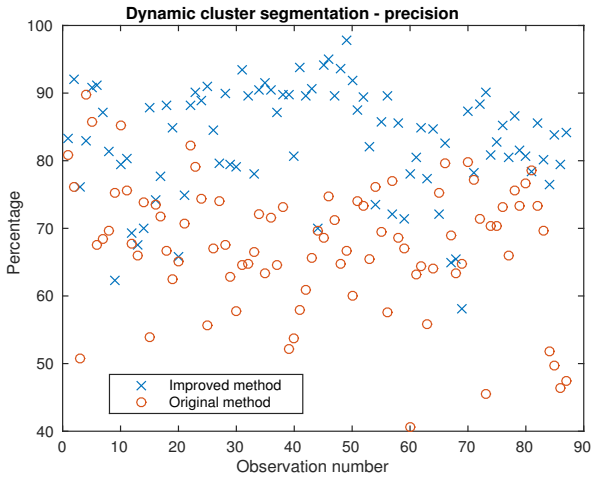

Fig. 5: Dynamic element segmentation - precision

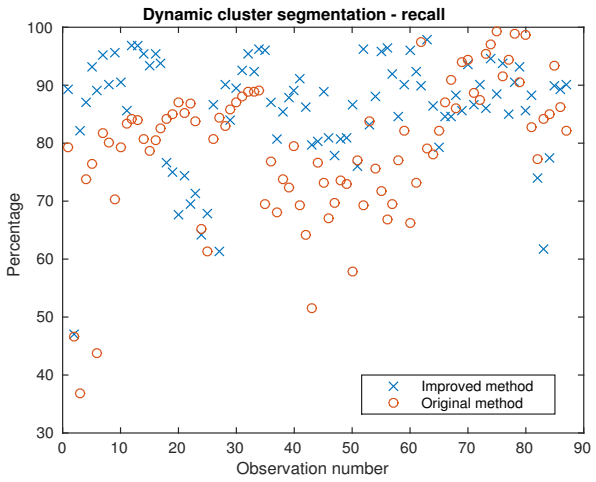

Fig. 6: Dynamic element segmentation - recall

To generate precision and recall values at each observation, we compared the reported dynamic elements with the true dynamic elements, as given by the labelled data for that particular observation, (i.e. precision = true positives / total reported dynamic and recall = true positives / total dynamic in labelled data).

TABLE I: Average precision recall - original method and improved method

	Precision (%)	Recall (%)
Original method	67.6	79.4
Improved method	82.7	85.9

We achieve an increase of 15 percentage points in the precision of the dynamic elements detected, which in practice means we are detecting less elements corresponding to noise, while the 6 percentage points increase in recall means we are segmenting dynamic elements more reliably. It must be noted here that we were not working with an engineered dataset, where the dynamic elements were under our control. As a result the labelling was not perfect. We have labelled most of the important dynamic elements, such as chairs, people, backpacks, jackets, laptops, monitors, etc, however sometimes we detect other dynamic elements, such as cups or mugs, or bags on the floor which we have overlooked while labelling - this gives rise to some of the unevenness in the precision curve. One way to address this limitation would be to discard smaller dynamic clusters based on a size threshold, however this would also result in discarding partial matches of larger objects.

Another factor affecting the precision and recall figures is the motion of low dynamics objects (such as couches, desks, etc.). Since these objects are mostly static in our observations we have not labelled them as dynamic. Whenever these objects move, they are segmented as dynamic for some time until they are absorbed back into the static structure, as defined in section IV-B. During that time, the precision and recall figures will be affected.

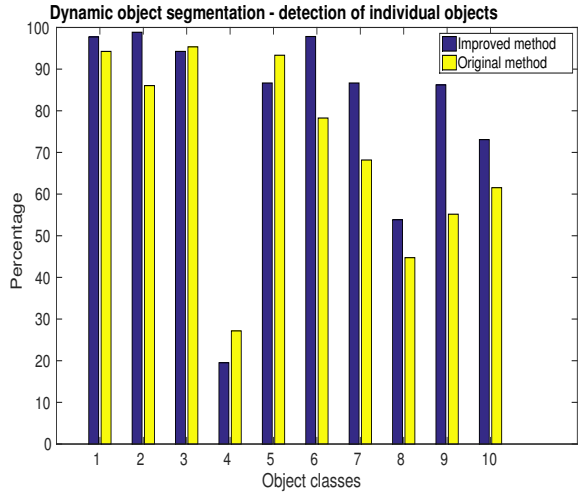

Fig. 7: Dynamic element segmentation - detection rate

We have also checked the performance of the segmentation by looking, for each element in each observation in the labelled data, whether it was detected and segmented out as a

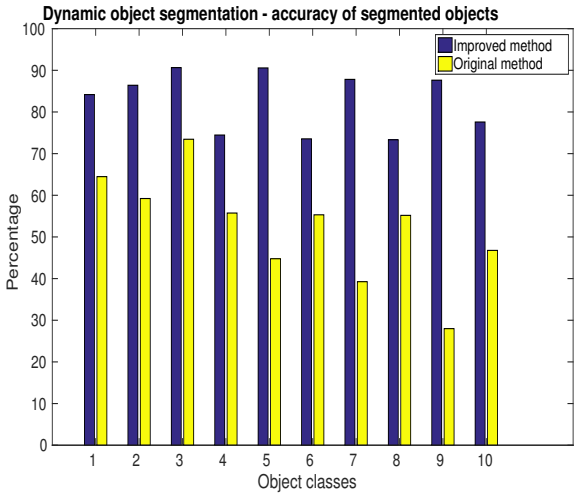

Fig. 8: Dynamic element segmentation - accuracy, averaged over all detections for each object

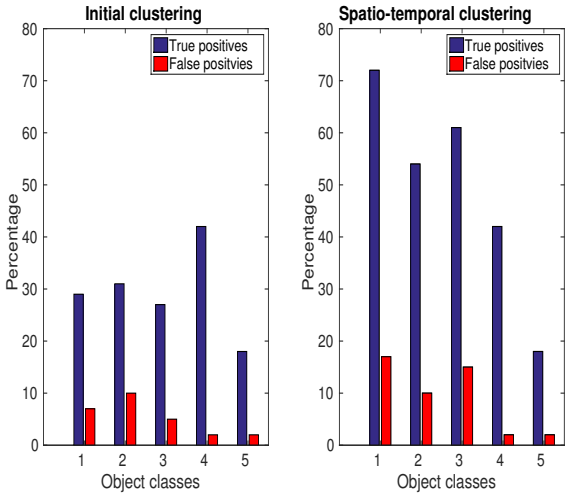

Fig. 9: Dynamic element clustering - true and false positive rate for the initial and spatial-temporal clustering methods

dynamic element (shown in Fig. 7), and if segmented, how accurate was the segmentation, in terms of spatial overlap between the labelled element and the segmented element (shown in Fig. 8, averaged across all the detections for each particular object). The figures show the results for the 10 elements with the highest number of occurrences in the labelled set.

Again we see a significant increase in the performance of the improved method, particularly in the accuracy of the elements segmented. The average detection rate and accuracy are summarized in Table II.

TABLE II: Average detection rate and accuracy - original method and improved method.

	Detection rate (%)	Accuracy (%)
Original method	52.2	79.4
Improved method	70.4	82.6

B. Dynamic element clustering

To quantify the results of the clustering algorithm, we have compared the resulting clusters with the objects in the labelled dataset, where each object in the labelled dataset (e.g. a chair or a backpack) is present a certain number of times across the observations making up the dataset. Thus subsets of a particular object's instances would be covered by various clusters we obtain through our algorithm, which we count as true positives for those particular clusters. Conversely, those clusters might contain other instances of other objects due to clustering errors; those instance we count as false positives with respect to the object we are currently matching to. In the case when multiple clusters contain instances of a particular object (which is indeed the case most of the time), we display the one with the highest number of true positives in the subplots of Fig. 9.

In Fig. 9 we show the true positive and false positive percentages for five of the objects in the labelled dataset, on the left displaying the result of the initial clustering and on the right the result of the spatial-temporal clustering. There is a clear and visible increase in the percentage of true positives with a very little increase in the percentage of false positives after the spatial-temporal clustering step. The object classes represent the same physical objects in both the subplots.

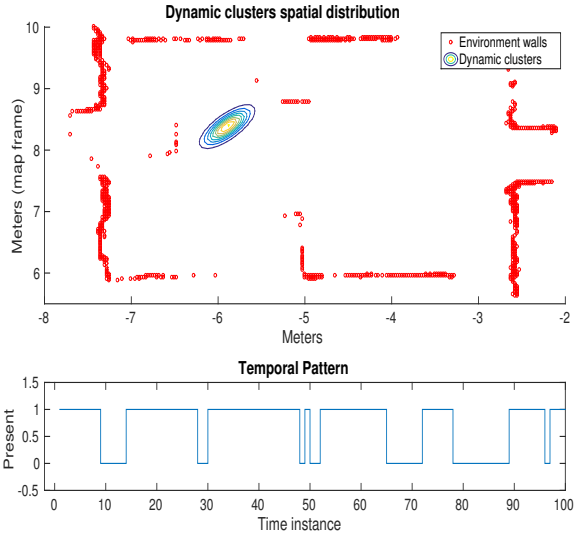

Fig. 10: Dynamic element spatial-temporal distribution

Fig 10 shows the resulting spatial-temporal distribution of one of the dynamic element clusters. The spatial distribution is shown in the map frame of reference here modelled as a Gaussian, and for clarity we also display the outline of the environment at the waypoint where the observations were collected. The bottom row of Fig. 10 shows the actual observed temporal pattern of the cluster across all the observations collected by the robot.

VIII. Conclusion and future work

This paper presents a novel method of object clustering based on spatial distributions and temporal patterns. We show that the spatial-temporal clustering algorithm allows clustering object instances that vary significantly in appearance, and that this leads to an increase in overall classification accuracy. The input to the clustering algorithm is a set of dynamic elements segmented through the Meta-Room method [1], and we investigate the performance of this segmentation method on a large dataset collected autonomously by a mobile robot over a long time period. We propose a number of improvements to the original segmentation method and we evaluate their performance in terms of precision and recall of the dynamic elements segmented as compared to a labelled dataset. Finally, we share the collected dataset with other researchers along with the tools required for easy access as well as for labelling.

For future work we will focus on developing a probabilistic way of assigning segmented objects to clusters, which allows for correcting the assigned labels once more information becomes available. We will also investigate how the spatial-temporal models of the dynamic element clusters can be used to predict where the objects are most likely to be at different points in time, by exploiting things like periodicities in their behaviour. In addition, we will look into whether the spatial-temporal models can be used to cluster objects in different rooms or different environments altogether. And finally, we would like to see how localization can be improved in an environment where the robot knows which objects are dynamic and that they are likely to move within some spatial distribution.

IX. Acknowledgments

The work presented in this papers has been funded by the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement No 600623 (”STRANDS”), the Swedish Foundation for Strategic Research (SSF) through its Centre for Autonomous Systems and the Swedish Research Council (VR) under grant C0475401.

References

[1] 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 IEEE/RSJ International Conference on*. IEEE, 2014, pp. 1854–1861.

[2] J. Saarinen, H. Andreasson, and A. J. Lilienthal, “Independent markov chain occupancy grid maps for representation of dynamic environment,” in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, 2012, pp. 3489–3495.

[3] T. Kucner, J. Saarinen, M. Magnusson, and A. J. Lilienthal, “Conditional transition maps: Learning motion patterns in dynamic environments,” in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE, 2013, pp. 1196–1201.

[4] P. Biber and T. Duckett, “Experimental analysis of sample-based maps for long-term slam,” *The International Journal of Robotics Research*, vol. 28, no. 1, pp. 20–33, 2009.

[5] A. Walcott-Bryant, M. Kaess, H. Johannsson, and J. J. Leonard, “Dynamic pose graph slam: Long-term mapping in low dynamic environments,” in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, 2012, pp. 1871–1878.

[6] T. Krajnik, J. P. Fentanes, O. M. Mozos, T. Duckett, J. Ekekrantz, and M. Hanheide, “Long-term topological localisation for service robots in dynamic environments using spectral maps,” in *Intelligent Robots and Systems (IROS), 2014 IEEE/RSJ International Conference on*. IEEE, 2014, pp. 4537–4542.

[7] R. F. Salas-Moreno, R. A. Newcombe, H. Strasdat, P. H. Kelly, and A. J. Davison, “Slam++: Simultaneous localisation and mapping at the level of objects,” in *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*. IEEE, 2013, pp. 1352–1359.

[8] Y. Li, A. Dai, L. Guibas, and M. Nießner, “Database-assisted object retrieval for real-time 3d reconstruction,” in *Computer Graphics Forum*, vol. 34, no. 2. Wiley Online Library, 2015.

[9] M. Gunther, T. Wiemann, S. Albrecht, and J. Hertzberg, “Building semantic object maps from sparse and noisy 3d data,” in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE, 2013, pp. 2228–2233.

[10] J. Mason and B. Marthi, “An object-based semantic world model for long-term change detection and semantic querying,” in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, 2012, pp. 3851–3858.

[11] 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*. IEEE, 2011, pp. 2623–2629.

[12] E. Herbst, X. Ren, and D. Fox, “Rgb-d object discovery via multi-scene analysis,” in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*. IEEE, 2011, pp. 4850–4856.

[13] E. Herbst, P. Henry, and D. Fox, “Toward online 3-d object segmentation and mapping,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

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

[15] A. Aldoma, F. Tombari, L. Di Stefano, and M. Vincze, “A global hypotheses verification method for 3d object recognition,” in *Computer Vision–ECCV 2012*. Springer, 2012, pp. 511–524.

[16] F. Tombari, S. Salti, and L. Di Stefano, “Unique signatures of histograms for local surface description,” in *Computer Vision–ECCV 2010*. Springer, 2010, pp. 356–369.

[17] F. Endres, C. Plagemann, C. Stachniss, and W. Burgard, “Unsupervised discovery of object classes from range data using latent dirichlet allocation.” in *Robotics: Science and Systems*, vol. 2. Seattle, Washington;, 2009, p. 113120.

[18] J. Shin, R. Triebel, and R. Siegwart, “Unsupervised discovery of repetitive objects,” in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE, 2010, pp. 5041–5046.

[19] O. Mattausch, D. Panozzo, C. Mura, O. Sorkine-Hornung, and R. Pajarola, “Object detection and classification from large-scale cluttered indoor scans,” in *Computer Graphics Forum*, vol. 33, no. 2. Wiley Online Library, 2014, pp. 11–21.

[20] D. Chetverikov, D. Svirko, D. Stepanov, and P. Krsek, “The trimmed iterative closest point algorithm,” in *Pattern Recognition, 2002. Proceedings. 16th International Conference on*, vol. 3. IEEE, 2002, pp. 545–548.

[21] T. Stoyanov, M. Magnusson, and A. J. Lilienthal, “Point set registration through minimization of the l 2 distance between 3d-ndt models,” in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 5196–5201.

[22] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, “Orb: an efficient alternative to sift or surf,” in *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, 2011, pp. 2564–2571.

[23] M. A. Fischler and R. C. Bolles, “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography,” *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[24] S. Agarwal, K. Mierle, and Others, “Ceres solver,” http://ceres-solver.org.

[25] R. B. Rusu, G. Bradski, R. Thibaux, and J. Hsu, “Fast 3d recognition and pose using the viewpoint feature histogram,” in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*. IEEE, 2010, pp. 2155–2162.

[26] C. Rother, V. Kolmogorov, and A. Blake, “Grabcut: Interactive foreground extraction using iterated graph cuts,” *ACM Transactions on Graphics (TOG)*, vol. 23, no. 3, pp. 309–314, 2004.