Robotics and Autonomous Systems 91 (2017) 0–11

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot

Efficient retrieval of arbitrary objects from long-term robot observations

Nils Bore *, Rares Ambrus, Patric Jensfelt, John Folkesson

Royal Institute of Technology (KTH) Stockholm, Robotics, Perception and Learning Lab, SE-100 44, Sweden

ARTICLE INFO

Article history:
Available online 12 January 2017

Keywords:
Mapping
Mobile robotics
Point cloud
Segmentation
Retrieval

ABSTRACT

We present a novel method for efficient querying and retrieval of arbitrarily shaped objects from large amounts of unstructured 3D point cloud data. Our approach first performs a convex segmentation of the data after which local features are extracted and stored in a feature dictionary. We show that the representation allows efficient and reliable querying of the data. To handle arbitrarily shaped objects, we propose a scheme which allows incremental matching of segments based on similarity to the query object. Further, we adjust the feature metric based on the quality of the query results to improve results in a second round of querying. We perform extensive qualitative and quantitative experiments on two datasets for both segmentation and retrieval, validating the results using ground truth data. Comparison with other state of the art methods further enforces the validity of the proposed method. Finally, we also investigate how the density and distribution of the local features within the point clouds influence the quality of the results.

1. Introduction

Mobile robots are becoming increasingly capable. They are now being commercialized and deployed primarily in indoor environments for continuous operation. Examples include robots serving as vacuum cleaners, logistics systems, or museum tour guides. The next frontier will be robots working to help and assist people in more challenging scenarios. To adapt to more complex tasks, these robots will also require more complex perception capabilities. By now, ubiquitous 3D sensors are used on most indoor robots and the data is fused into large maps. However, the sheer size of some environments and the long run times mean that we will not be able to store the raw 3D data on board the robot. In addition, we want to enable easy inspection of this data, collected over months of autonomous operation. To achieve this, we propose to store a compact representation of the data on the robot, and allow for fetching of specific parts of the raw data or maps from a database on a network server. Overall, the large amount of data characterizes our work. We propose to develop a fast object retrieval system which returns a list of likely matches. More precise data for these matches can then be fetched from the server and validated using more costly techniques. In Fig. 1, we present an overview of the proposed system.

The motivation for such a system comes from our work on mapping and unsupervised object detection. Our robots continuously patrol for periods of several weeks or months. While patrolling, the robots inspect the environment at specified locations at regular intervals. From the observations, we construct 3D maps and look for anything that moved as compared to previous observations [1]. This information can then be used for unsupervised object detection. Given that we have detected objects, our aim in this work is to find other instances of the same objects in other parts of the map that did not change, or from earlier observations. This would then allow us to quickly build precise object models that could be used for example for manipulation or for querying again.

Of course, approaches other than change detection might be used to reason about if a part of a map is of interest. For example, one might use the observation that many objects lie directly on a flat surface [2,3]. Another example is the *objectness measure* of Karpathy et al. [4]. A problem with many of these techniques, including change detection, is that they rely on assumptions that are not always fulfilled. But if an instance is detected as interesting using any technique, we can feed it into the proposed retrieval system and find more examples in places where the assumptions might not be fulfilled. Notably, our approach is not constrained to any particular size or shape. It should be equally suited for searching small objects as for larger furniture. By decoupling the initial detection step, the system also allows, for example, a user to enter

* Corresponding author.
E-mail addresses: nbore@kth.se (N. Bore), raambrus@kth.se (R. Ambrus), patric@kth.se (P. Jensfelt), johnf@kth.se (J. Folkesson).

http://dx.doi.org/10.1016/j.robot.2016.12.013

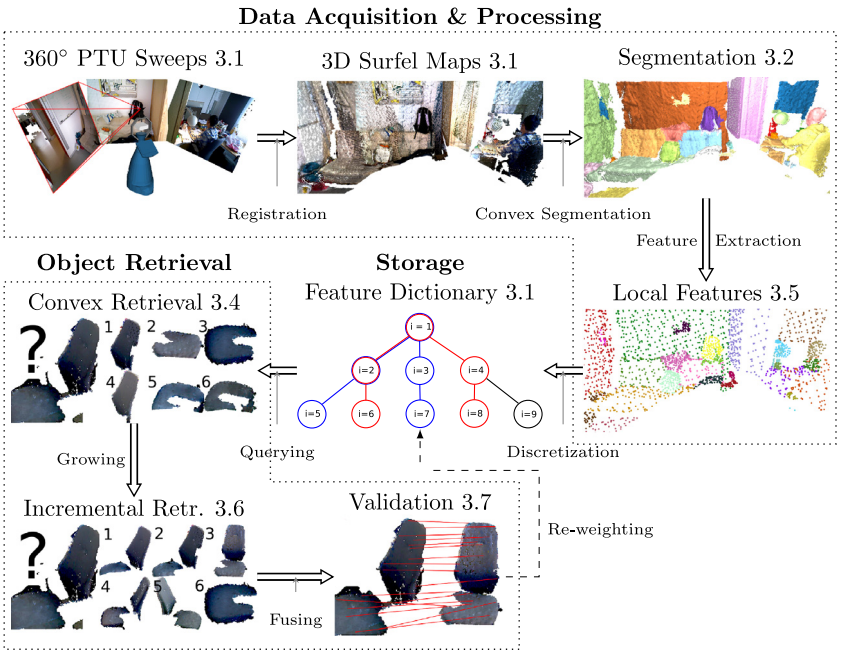


Fig. 1. This paper proposes a pipeline for storage and retrieval of large collections of 3D map data. Above is an overview of the system components with section numbers listed next to the block labels. The data is collected by robots over long periods of time, as shown in block "Data Acquisition". This data is stored along with a compact generic description, labeled as the "Storage". When a point cloud (such as the chair marked with "?") is presented as a query it is fed into the query pipeline, labeled "Object Retrieval".

knowledge of instances that should be extracted and maintained using their own models.

Most of the previous work on recognition and retrieval in 3D maps relies on some sort of segmentation to produce parts or objects that can be analyzed separately [2,3,5]. Because our representation should encompass objects of different size and potentially very complex shape, we cannot rely on any one segmentation to perfectly segment the scene into interesting objects. In general, what constitutes an object largely depends on the application area, be it grasping or navigation. In our case, we look to combine segmentation with retrieval. This allows us to adapt the segmentation to the given query object. Our approach depends on segmenting the scene into parts that should be no larger than any single object, in effect an *oversegmentation*. In [6], Schoeler et al. showed that an *unsupervised* segmentation approach based on convexity can achieve state of the art performance of segmenting objects into semantically meaningful parts. Our approach is inspired by this result and employs a similar segmentation. The goal is then to match the query object that we are looking for to one or several of these segments. In this regard, our proposed model has advantages when compared to a fixed segmentation. The added flexibility comes at the price of not being able to pre-compute global features for every segment. This is solved by using local 3D features assembled into histogram-like representations that can be combined to create features for larger segments.

In the environments where our robot operates, the surfaces of the building such as walls and floor dominate the measurements. To enable retrieval of smaller things while being agnostic to what we can represent, we need to adapt the fidelity of our representation in different areas. A wall does not need to be represented with the same accuracy as a mug since we can still see that it is a wall, even at a coarser resolution. For the mug, this is not true. In our approach this corresponds to how densely we distribute our local feature descriptors. In this work, we present an analysis of how to distribute the features in a way that is suited for the task at hand. One might see this as a continuation of an earlier development in robotic mapping, where the fidelity of metric maps has been adapted to different areas, depending on the task [7,8].

Our matching scheme is inspired by previous work on retrieval in large databases of images [9]. These results show that instance retrieval in databases of millions of images is possible, and that the loss in performance as the data base grows is manageable. In continuous data collection in indoor environments, most of the environment remains static. Therefore, the new observations should add little new information to confuse a query for already observed objects.

For an overview of our system, see Fig. 1. Our robot patrols an environment and collects large amounts of 3D sweeps. The sweeps are registered and fused into 3D surfel maps, see Section 3.1. The maps are segmented using convex cues, see Section 3.2. Local features are extracted with different densities depending on the segment size, see Section 3.5. The feature sets are then assembled into a hierarchical bag-of-words representation for fast retrieval, Section 3.1. By comparing these visual words, the system can group neighboring segments in the maps if they better match the query object, see Section 3.6. This allows us to retrieve arbitrary shapes. The retrieved surface segments can be validated to see if they are identical to the query object. Finally, this information is fed back to improve the overall retrieval results, see Section 3.7.

We are presenting the following contributions beyond our previous work [10]:

1. An extension of our approach from RGB-D frames to work on full 3D surfel maps.
2. A separate evaluation of the incremental segmentation component of the system.
3. A detailed analysis of how to adapt keypoint density across the maps to improve compactness.

2. Related work

Given a query point cloud, our aim in this paper is to identify more examples of identical instances within a large collection of 3D maps in the form of unsegmented point clouds. Given the nature of our data, which is collected in real world environments with noise from movement and sensors, this is a challenging task. Another

requirement is that the retrieval should be fast, performing a query should take at most a few seconds, even as we add more data.

Similar topics have seen a lot of work in the field of 2D image retrieval. While the early systems focused mostly on storing and retrieving images annotated with some semantic information [11], work in the field is now mainly concerned with finding more examples given just a single query image. In addition to digital images, the methods can also be applied to videos, as shown for example by Sivic et al. [12]. Like many of the later methods, they make use of local image features, in this case SIFT [13]. They employ clustering of features, similar to a bag-of-words representation, and weight the importance of words depending on how common they are. In [9], Nistér & Stewénius build on this, by now, relatively standard approach to image retrieval. The authors propose to hierarchically construct a dictionary of visual words, enabling features to be weighted and compared at several different levels of resolution. Similar to [12], they employ K-means clustering, but iteratively to construct a tree. The method demonstrates good performance on a dataset of millions of images. Notably, the method scales well as more images are added, something that is important in the scenario of continuous robot operation. This is also the method used to represent the local features in our work. The well known *pyramid match kernel* (PMK) by [14] Grauman et al. is a similar concept that is instead based on histograms at different resolutions. One of the advantages of this method is that it handles partial overlap better, something which we also address in [10]. It is also worth noting the work of Philbin et al. [15]. They demonstrate improvements over [9] by using a comparatively complex approximate K-means clustering. In later years, aggregating local descriptors into a so called *V-LAD* feature has been a popular technique [16,17]. The main advantage of V-LAD is that these descriptors can be concisely represented by applying optimized dimension reduction, using only a few bytes of memory. Image retrieval techniques have also been applied within robotics. An example of this is [18], where the authors use it in conjunction with other methods to improve Monte Carlo localization.

In regard to our work on result validation, it is worth mentioning some works on feedback based image retrieval. An early example of this is the MindReader system [19]. It allows a user to query for an example and then rate the relevance of the retrieved images. A distance metric is then optimized to force the relevant results closer to the query in feature space. Other approaches instead rely on weighting different local features when performing a second round of retrieval [20,21], so called *feature relevance weighting*. Our approach is more similar to these schemes. Within robotics, Shin et al. use a cheaper initial matching technique, together with precise verification [22].

There are numerous relevant works on 3D recognition, more than can be summarized here. However, we will attempt to discuss the methods that are most similar to ours. In recent years, the need has arisen to also search large collections of 3D models, similar to earlier developments for images. Examples include retrieval of human generated CAD models, often using global shape features [23]. As shown in [10], global shape features can be effectively applied to problems similar to ours. Work on such methods, as pertains to point clouds has mostly dealt with categorization or recognition of objects using several training examples. Many of these features are based on histograms of point and normal direction distributions. The *Viewpoint Feature Histogram* (VFH) [24] is one such method often used as a baseline. In [25], Wohlkinger and Vincze showed that the D2 shape distribution [26] can be used for matching point clouds to a data base of object models. Asari et al. [27] demonstrate that this technique can also be used on 3D sensor data. Notably, [25] applies a hashing algorithm to the features to enable faster matching. In [28], Aldoma et al. demonstrate one of the main differences between our retrieval application and many recognition systems. Because their scenes contain objects from a pre-trained set, the measurements must be explained by one of these objects, something that the authors employ to reason about e.g. occlusions. In relation to the vocabulary tree that we are using, Redondo et al. [29] propose a spatial version of the pyramid match kernel for use in 3D point clouds. However, this version of the PMK is constructed in such a way that it is viewpoint dependent.

Another area of interest is unsupervised object detection inside 3D maps. Particularly, we are interested in the kind of features used here. In [30], Herbst et al. present a system for unsupervised object discovery by change detection in 3D maps. Like our system, they use surfels as the underlying representation. For comparing segments, their system uses Kernel descriptors computed on backprojected RGB and depth patches, together with shape matching using ICP. Their system also performs spectral clustering of detected objects to discover object classes. Another approach for unsupervised discovery is the *supporting planes assumption*, making use of the placement of many objects on tables or floors. In [2], Mason et al. use this for unsupervised discovery. To compare two objects, the authors propose to compute multiple color histograms at various intersections of the objects, and use the minimum histogram value as the distance. They employ this in combination with the size of the object and VFH features [24]. The authors also perform a simple connected components analysis in their descriptor space to cluster instances into classes. An important difference to our system here is the speed requirement. In particular, computing intersections between a query point cloud and all possible matches as employed by [2] would be too slow for our retrieval system, partly because the time complexity grows linearly with the size of the data. The same holds for the ICP matching of [30].

Finman et al. [5] is the work most similar to ours. The aim of their work is to identify instances in large 3D maps. To this end, their main contribution is to learn a segmentation adapted to the current instance to be recognized. In addition, their system finds objects using change detection, something that is outside the scope of this work but which we addressed in [1]. Given a detected object within a map, they propose to vary segmentation parameters and segmentation type to optimally segment the object within its scene. The same segmentation type and parameters can then be applied to other maps and the segments compared to the given instance. The system employs global features based on statistics of points and curvatures along the principal axes and average colors. Both the segmentation parameters and features can be combined over several examples to iteratively refine the models. Schoeler et al. [6] proposed an unsupervised segmentation method based on convexity constraints. From an initial oversegmentation into supervoxels [31], they partition an adjacency graph using greedy cuts. The method demonstrates good segmentation results on several benchmarks, also compared to supervised methods. Our segmentation scheme is inspired by this approach.

Our work differs from similar methods working on 3D maps [30,2,5] in that we aim for *retrieval* of instances within the maps, using scalable methods that take only a *few seconds*. Similar to Finman et al. [5], we also seek to adapt our segmentation, but the time constraint means that we cannot re-segment the maps for every retrieval. Instead we employ a scheme for adaptively aggregating segments based on a query object.

3. Method

Our aim is to retrieve specific object instances from multiple months of data in the form of potentially noisy RGB-D frames, with varying lighting conditions. To enable retrieval, we need to first separate the data by applying a segmentation. We will present a method for matching the query point cloud to multiple neighboring segments using local features. This reduces our reliance on any single segmentation scheme. We proceed to describe our protocol for data collection, and how the individual images are combined into a more manageable representation.

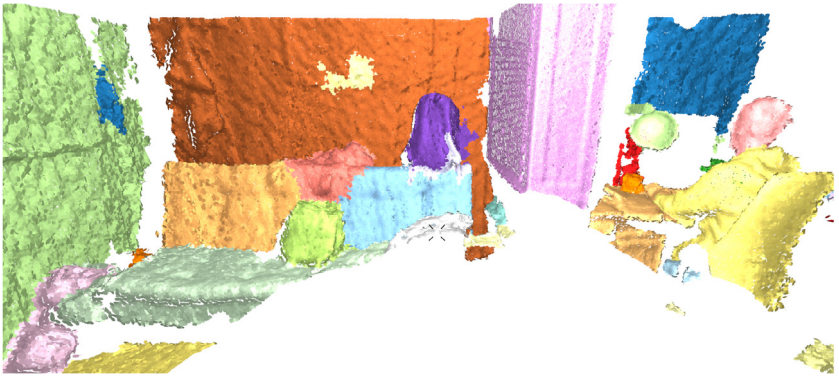


Fig. 2. Example of the convex segmentation of a part of a local map.

3.1. Local maps

Our wheeled robots are continuously patrolling indoor environments. Waypoints are defined at specified locations in these buildings, typically in different rooms. At regular intervals, the robot stops at a waypoint and performs a sweep with a Kinect-like sensor mounted on top of a *Pan-Tilt Unit* (PTU). Using the position of the PTU together with ICP and bundle adjustment, the individual RGB-D frames can be precisely registered to each other (see [32]), creating what we will refer to as *local maps*. Typically, a local map covers the area of a room, ranging between 20 and 30 m^2. To merge the data and build a surfel map we use the data fusion component of the ElasticFusion framework by Whelan et al. [33]. This gives us state-of-the-art data fusion with the future possibility of precisely localizing in and updating the constructed maps. In this paper, we use the surfel map representation to get colored points and normals and also to render the surfaces of maps and retrieved point clouds. In this regard, we expand on our previous work in [10], where the system works directly on RGB-D frames.

3.2. Convex segmentation

For an initial rough segmentation of the maps we rely on local convexity. The segmentation employed is similar to that of Schoeler et al. [6]. Like them, we perform an oversegmentation of the point map into *supervoxels* [31] and cluster the segments by local convexity using iterative graph cuts. Instead of RANSAC, we use the Stoer–Wagner min-cut algorithm [34]. Similar to [6,4,5], we consider the edge between two neighboring supervoxels a, b with point-normal pairs (p_a, n_a), (p_a, n_a) to be convex if $(p_a - p_b) \cdot (n_a - n_b) > 0$. Our algorithm defines the edge weight $\omega(a, b)$ as the average of $\frac{1}{\|p_a - p_b\|}(p_a - p_b) \cdot (n_a - n_b)$ for the respectively closest point-normal pairs at the boundary of the two supervoxels. In this way we deviate from [6,4], which use the centers and mean normal vectors of the supervoxels for the convexity calculation.

Our method performs iterative cuts until the cut cost reaches a certain limit. In addition, we perform an additional cut iteration taking color into account to ensure that the obtained segments are visually homogeneous. To compare the colors of two supervoxels inside a larger segment, we model their *hue* and *saturation* values in HSV space as normal distributions. The *value* component is discarded to make the measure more lighting independent. We define the color similarity between the supervoxels with normal distributions $N_a = N(\mu_a, \Sigma_a)$, $N_b = N(\mu_b, \Sigma_b)$ within the current larger segment $N_S = N(\mu, \Sigma)$ in terms of a normalized *Kullback–Leibler divergence*:

$$KLD_{a,b} = \frac{KLD(N_a, N_b)}{\min(KLD(N_S, N_a), KLD(N_S, N_b))}.$$

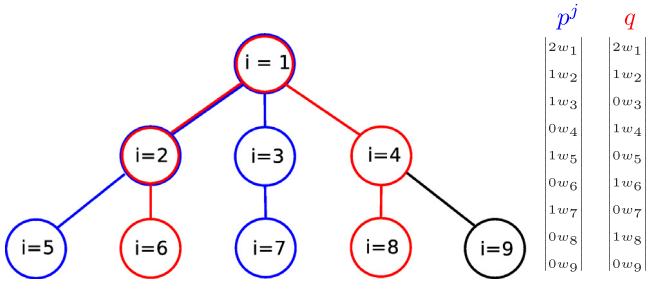


Fig. 3. A simplified query vector q and a data base vocabulary vector p^j. Each node in the vocabulary tree has an index i. The vector entry at index i is the number of times a feature of the instance has a path through node i times the weight w_i. The similarity is determined by some distance metric operating on p^j and q.

The normalization component limits the splitting within consistently heterogeneous segments, e.g. textured surfaces. $KLD_{a,b}$ is subtracted from every edge weight $\omega(a, b)$ times a small tunable constant α_c. The value of α_c needs to be large enough so that objects are not undersegmented; in all experiments we use $\alpha_c = 5 \cdot 10^{-3}$. It is also worth noting that we assign a fixed weight $\omega_{flat} = 0.3$ to all edges that are approximately flat, ensuring a slightly higher cost for cutting across flat surfaces. We refer to this method as a *convex segmentation* although it does take some color into account and the final segments are not guaranteed to be strictly convex. Fig. 2 shows an example output.

3.3. Hierarchical dictionary

Given the segments computed using the convex segmentation method, we extract sets of local features for each segment (see Section 3.5). We first describe the data structure and method used to store and compare the features, as these influence the choice, density and distribution of the local features on the segments.

Our scheme for matching segments is based on comparing sets of local features. This is similar to several systems for image retrieval, many of which use different representations to enable scalable set matching. We are using a method from [9] for hierarchically matching sets of features. The main idea is that typical *bag-of-words* representations discretize at too coarse a resolution. The *vocabulary tree* (VT) instead matches discretized features at different levels of resolution, with different weights at every level. Even if two features do not match at the finest level of resolution (the leaves of the tree), they might match at a coarser level. Closer matches are reflected in the distance metric through the weights w_i of the nodes being higher towards the leaves, see Fig. 3.

The VT representation is based on cluster centers in the feature vector space. At each level, a feature is represented by its closest

cluster center. In the hierarchical structure of the VT, a feature is first assigned a cluster at the coarsest level and then iteratively assigned a new cluster center at the finer level until we reach a leaf. Similarly, hierarchical K-means is employed to first learn the actual clusters given some training data. Once we have trained the representation, new points can be quickly added by hierarchical discretization. This is important, since the initial training can be time consuming. Additionally, in [9] the authors showed that the retrieval system can continue to function well, even when scaling to millions of images. In the proposed system, we employ $K = 8$ different clusters at every level as recommended in [9]. In all the experiments, the height of the VT is 6.

From the convex segments we extract PFHRGB features, a color augmentation of the PFH feature [35]. They showed good performance in an evaluation of local features [36], also as compared to SHOT [37], as we confirmed in [10]. For each set of features $S_j = \{f_1, \ldots, f_{\Theta_j}\}$, representing a convex segment, we have the corresponding *vocabulary tree vector* p_j encoding the paths through the tree (see Fig. 3). The vectors are essentially weighted histograms and they can be compared using some metric (typically L_1) to decide how well the segments match. Initial weights are assigned using a *inverse document frequency* [9].

3.4. Vocabulary metric

In [10], we found that using a different metric for comparison of vocabulary vectors improved results. More importantly, it allows for some partial matching of segments, enabling our incremental matching scheme, see Section 3.6. The distance $d(p^j, q)$ between vocabulary vectors p^j and q is then defined by

$$\rho(p^j, q) = \frac{1}{\|q\|_1} \sum_i \max(q_i - p_i^j, 0), \quad p_i^j, q_i \geq 0,$$

$$d(p^j, q) = \max(\rho(p^j, q), \rho(q, p^j)) = 1 - \frac{\sum_i \min(p_i^j, q_i)}{\max(\|p^j\|_1, \|q\|_1)},$$

with $\|\cdot\|_1$ denoting the L_1 norm. More intuitively, if two segments are overlapping, it can be interpreted as the amount of features that is in the larger segment, but not in the smaller [10].

3.5. Keypoint extraction and descriptors

For keypoint extraction, we use *Intrinsic Shape Signatures* (ISS) [38], following the recommendation of [39,40]. Keypoints are identified by finding points in regions with large variation in all three dimensions. The variation is characterized by the smallest eigenvalue of the point distribution in the neighborhood. The method then chooses keypoints by requiring this value to be larger than a *saliency threshold*. We will investigate how the density of keypoints affects the performance of the vocabulary matching, both in terms of speed and precision. Distributing keypoints more densely will benefit retrieval up to a point. We hypothesize that this point will vary depending on the size of the objects that we want to find. For small objects, it might be beneficial to extract more keypoints than for large objects. Further, we will investigate if extracting more features for the larger objects will make retrieval of smaller ones harder. The keypoint density of ISS is varied by changing its saliency threshold.

To analyze the performance when varying keypoint density, we cluster the objects into 6 different size classes V, each with $\mathbf{N} = (N_v)_{v \in V}$ objects. We will analyze retrieval performance for different choices of $\mathbf{f_d}$, a vector mapping each size to its approximate number of keypoints.

As we will see in Section 5.3, extracting fewer keypoints for the largest object size improves compactness, while having a negligible effect on the method's performance. This analysis leads us to the values we use in the retrieval experiments (see Section 5.2): we extract about 20 keypoints/dm^3 for the smaller sizes, and $\sim$8/dm^3 for the largest. If we approximate the sizes of the objects with sizes of segments in an oversegmentation, the keypoint density will be higher than necessary in certain places. But importantly, it will remain low on the surfaces that dominate indoor environments, i.e. walls and floors, since they can be easily segmented. An alternative would be to base the estimates on characteristics other than size. One example would be the *objectness measure* of Karpathy et al. [4]. In addition to size they also look at e.g. compactness and colors to decide if a surface looks like an object.

3.6. Incremental matching and segmentation

The convex segmentation (Section 3.2) is only the first part of our pipeline. We would now like to group neighboring segments in a map if they together are more similar to the query object than individually. The approach, like the VT matching, also has to be efficient. Fortunately, the underlying representation is essentially a weighted histogram. Therefore, the vocabulary vector of a compound segment is simply the addition of those of its constituting segments, enabling fast evaluation of different combinations. From an initial query on the convex segments, we take the top M results and incrementally grow our segments from them, see Fig. 4(a) for more detail. In [10] we observed that $M \approx 200$ initial segments is sufficient. In all of the experiments presented we let $M = 200$. The approach then incrementally adds new convex segments to the initial ones as long as the distance between the vocabulary vector of the larger segment and the query vector q is less than for the old smaller segment. This approach is detailed in Fig. 4(b).

3.7. Learning a better weighting scheme

In [10], we introduced the concept of validating the first N results of the matching using a more expensive method. The idea is then to feed this information back to a second round of retrieval by updating the VT weights, w_i. To register the retrieved segments with the query object, we use SIFT features from the backprojected 3D segments to find correspondences and estimate the transform using RANSAC (see [10]). We have found that this approach works robustly even when the transformation is large.

In the current system, we measure how well two registered clouds match by their *overlap ratio*. In computer vision, the ratio measures the area of intersection between two regions divided by the area of their union. In point clouds, we can instead look at the volume that the point clouds are occupying by using a voxel grid. In other words the overlap ratio of the point clouds S_1, S_2 is

$$R_{overlap} = \frac{|vol(S_1) \cap vol(S_2)|}{|vol(S1) \cup vol(S_2)|},$$

where $vol(S)$ returns all the voxels of a specified size that S is occupying. In the experiments we use voxels with a side of 1 dm.

If we denote R_j to be the overlap ratio of a query result j with the query cloud, we compute its score among the N query results as

$$R_j^* = \frac{N}{\sum_{i=1}^{N} R_i} R_j.$$

The normalization ensures that the average score is 1. We define the new weight for nodes i in the vocabulary tree with an

procedure MATCH(q)
  $I = \{\text{segment indices}\}$
  $D_c = \{\}$
  for $m \in 1 \dots M$ **do**
    $j = \arg\min_{j \in I} d(q, p_j)$
    $D_c = D_c \cup \text{GROW}(j)$
    $I = I \setminus \{j\}$
  $D_i = \{\}$
  $S_{all} = \{\}$
  for $m \in 1 \dots M$ **do**
    $(S, d) = \arg\min_{(S,d) \in D_c} d$
    if $S_{all} \cap S = \emptyset$ **then**
      $S_{all} = S_{all} \cup S$
      $D_i = D_i \cup (S, d)$
  return SORT(D_i)

(a) Incremental matching.

procedure GROW(j_{min})
  $N(j) := \{\text{neighbors of j in map}\}$
  $S = \{j_{min}\}$
  $u = p_{j_{min}}$
  $d_{min} = d(q, p_{j_{min}})$
  loop
    $$i = \arg \min_{i \in N(j) \,:\, j \in S} d(q, u + p_i)$$
    if $d(q, u + p_i) < d_{min}$ **then**
      $u = u + p_i$
      $d_{min} = d(q, u)$
      $S = S \cup \{i\}$
    else
      return (S, d_{min})

(b) Growing of convex segments.

Fig. 4. The incremental matching and growing algorithms respectively. From the M convex vocabulary vectors p_j that are closest to q, we iteratively grow the segments using neighboring segments $N(j)$. The growing continues until the distance to the compound vocabulary vector u does not decrease any more. Finally, we need to make sure that no grown segments are overlapping, using S_{all}.

Fig. 5. Some of the object instances present in the KTH dataset. Notice how for example the chair on the far right has missing depth values, leading to a partial observation. The object categories are (left to right) chair, backpack, trash bin, boiler, sweater, pillow and chair. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

intersection of a path of the query vector q and one or more result vectors p^j (see Fig. 3 or [10]) as

$$w_i^* = \left(\frac{1}{|I_i|} \sum_{j \in I_i} R_j^* \right) w_i,$$

$$\text{with } I_i = \left\{ j \in 1 \dots N \mid p_i^j \neq 0,\ q_i \neq 0 \right\}. \tag{1}$$

With this update, nodes associated with well scoring results will contribute more to the distance metric after the re-weighting. This should ensure that more relevant matches make it to the top results. Since the same feature set with initial vocabulary vector q_i will be used for the second query, we only need to update the weights for nodes i with $q_i \neq 0$. Only these nodes contribute to the vocabulary vector distance both in the first and second query.

4. Experiments

The aim of this work is to deploy our system on a mobile robot, and allow for fast retrieval of instance observations from months of observations. To this end, we present quantitative and qualitative results on data collected autonomously over a period of several weeks.

4.1. KTH lab experiment

In this dataset, assembled by a robot in our offices at KTH, the robot first autonomously collected data in one room over a period of two weeks. We have then taken some of the objects that the robot observed and moved them around to different rooms and positions manually in order to get more variation. Overall, six different rooms have been recorded from different positions at different times of the day, meaning varying lighting conditions. The data has been recorded over a total of 24 days, generating 149 local maps. Through our convex segmentation method (3.2) we obtain approximately 12 000 segments, from which we further extract 3.4 million features which are stored in the vocabulary tree.

The local maps have been manually annotated with exact masks of a few different kinds of chairs, backpacks, pillows, trash cans etc. The annotations were done directly on the RGB and depth images, using GrabCut [41] in conjunction with manual drawing. Examples of some instances are presented in Fig. 5. Importantly, there are other instances in the same object categories that might confuse the search, for example similar sized trash bins, but with different color.

4.2. UK office longterm experiment

This dataset was collected autonomously by a mobile robot over a period of four weeks, in an office environment in the

United Kingdom. In total there are 103 local maps collected in ten different places. This amounts to approximately 6000 convex segments obtained through the segmentation method described in Section 3.2, from which we extract 2.2 million local features which are stored in the vocabulary tree. However, as we lack ground truth annotations of objects in this dataset, we only show qualitative results for queries of various objects in the results section.

4.3. Keypoint extraction

As outlined in Section 3.5, we investigate the retrieval quality for objects occupying different volume intervals $V = \{v_i\}$ s.t. $0 < v_0 \leq 4.5 < v_1 \leq 10.6 < v_2 \leq 15.0 < v_3 \leq 22.9 < v_4 \leq 40.5 < v_5$ dm^3. The values were chosen by histogram equalization to ensure that there is an approximately equal number of segments $\mathbf{N} = (N_v)_{v\in V}$ in each interval. We would like to evaluate different numbers of keypoints $f_d(v)$ for each size interval $v \in V$. To make this practical we discretize so that f_d maps v to one of 5 density classes for each v: $f_d : V \to D$ with $|D| = 5$. We want to find a function f_d with vector $\mathbf{f_d} = (f_d(v))_{v\in V}$ that minimizes the error given by $p(err|\mathbf{f_d}, \mathbf{N})$, the probability that a returned result does not match the query. The probability can be decomposed into error probabilities for the different queried segment size classes v:

$$p(err|\mathbf{f_d}, \mathbf{N}) = \sum_{v\in\mathbf{V}} p(err|\mathbf{f_d}, v, \mathbf{N})p(v). \quad (2)$$

It is important to note that the error for size v, $p(err|\mathbf{f_d}, v, \mathbf{N})$, depends not only on $f_d(v)$ but also on the assignments $f_d(w)$, $w \in V$, $w \neq v$, as they determine the approximate total number of features in the vocabulary $\mathbf{f_d} \cdot \mathbf{N} = \sum_{v\in V} f_d(v)N_v$, which also influences the error. However, benchmarking with all permutations of densities is still not feasible, as the number of combinations to evaluate is $|D|^{|V|} = 15\,625$ (i.e. for each $v \in V$ we have to evaluate $p(err|\mathbf{f_d}, v, \mathbf{N})$ for all the density combinations defined by $\mathbf{f_d}$). Instead, we assume that $p(err|\mathbf{f_d}, v, \mathbf{N})$ can be closely approximated by evaluating with another $\tilde{\mathbf{f}}_\mathbf{v}$ that returns the same number of keypoints $\tilde{f}_v(v) = f_d(v)$ for the queried size $v \in V$. Instead of evaluating all density combinations for the remaining objects in the other size classes $w \in V$, $w \neq v$, we impose an approximate total vocabulary size $\tilde{\mathbf{f}}_\mathbf{v} \cdot \mathbf{N} \approx \mathbf{f_d} \cdot \mathbf{N}$. Eq. (2) is thus approximated by

$$\sum_{v\in\mathbf{V}} p(err|\tilde{\mathbf{f}}_\mathbf{v}, v, \mathbf{N})p(v), \quad \text{with } \tilde{\mathbf{f}}_\mathbf{v} = \arg\min_{\tilde{\mathbf{f}}_\mathbf{v}} \left|\mathbf{f_d} \cdot \mathbf{N} - \tilde{\mathbf{f}}_\mathbf{v} \cdot \mathbf{N}\right|. \quad (3)$$

This approximation is used in our analysis of different keypoint densities in Section 5.3 and allows us to reuse approximations of $p(err|\mathbf{f_d}, v, \mathbf{N})$ for different $\mathbf{f_d}$ whenever v and $\tilde{\mathbf{f}}_\mathbf{v}$ are the same in Eq. (3). This leads to a total number of $|D| \cdot |D| \cdot |V| = 150$ combinations to be evaluated.

4.4. Comparisons

We are comparing our method to other techniques previously used for retrieval, both in a 3D setting and in images. A popular method used in image retrieval in recent years is V-LAD [16]. It enables a compact representation of sets of local features through optimized dimension reduction. This method is evaluated by matching to the convex segments and we feed the same local descriptors (PFHRGB [35]) into the V-LAD representation as we use for the vocabulary tree.

We also compare with the approach of Finman et al. [5]. In their method, they employ an adaptive segmentation scheme. For every object that is to be searched, the method learns segmentation parameters to optimally segment it in its map. This segmentation is then applied to the different maps where we are searching. Depending on the object, the scheme picks a segmentation based either on shape and convexity or on color. We explore the same parameter ranges as [5] for these two segmentation types. To speed up the approach, we are caching the initial graphs that are constructed, since they can be reused for different segmentation parameters. Further, the segmentation learning takes place only in a smaller part of the map containing the object. The recognition relies on a combination of global shape and color features for every segment. In [5] a combination of the different features results in a probabilistic score that can be used to recognize an object with some threshold. In our retrieval setting, we are instead counting the correct segments among the top matches. Instead of applying a threshold, we therefore return the segments with the highest scores across our data. Since their method takes an order of magnitude more time than the others, we query for only a subset of the annotations, about 400 instances. For each annotation, we search for matches within the entire dataset.

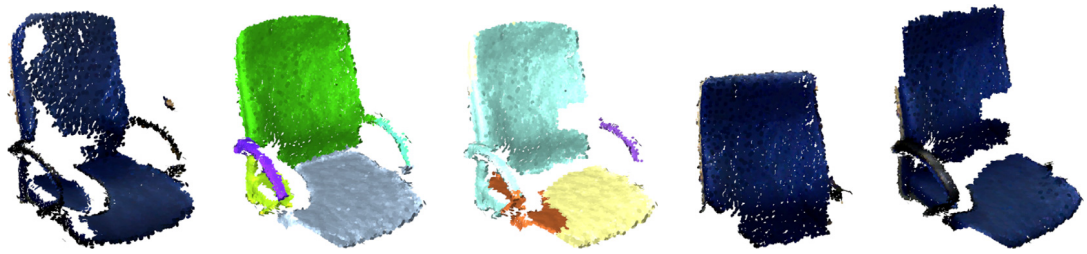
4.5. Segmentation metric

Since both our method and that of [5] rely on a segmentation method, we also evaluate the quality of the segmentations. We do this for every annotation in the initial round of observations in one room, which should enable good opportunity for learning. For every annotated instance within one visit, i.e. one local map, we randomly pick another instance of the same object as a training example. [5] optimizes the segmentation parameters for the provided example within its map. We then apply the resulting segmentation on the original map and find the segment with the highest *overlap ratio* with the annotated instance.

The proposed convex segmentation is also evaluated using this measure, but without any training examples. For evaluating the proposed incremental segmentation, we first query the vocabulary tree and use the top scoring convex segment with any overlap with the annotated instance and grow from there. The score is simply taken to be the overlap ratio between the grown segment and the annotation. It gives us a measure of the quality of the segmentation for the different methods.

4.6. Retrieval metric

We want a system that can retrieve previous observations of a particular object. The basic assumption is that we have enough data so that all instances that we query for have been seen at least a few times. Ideally, our retrieval system should be able to always return at least a few instances among the top results. However, due to the nature of the data, some observations will be hard to definitely identify with the query object since they might be partial or noisy. Instead, we would like to efficiently query for a fixed amount of instances and let a more costly and precise approach handle validation and possible registration of the results. This leads us to our evaluation metric. We define the *retrieval score* to be the ratio of segments with identical instance to that of the query among the top ten results. As stated, given that we have at least ten examples of every instance in the data, a perfect method would be able to retrieve ten identical instances, giving a score of 1. However, as our measurements are characterized by the nature of real world autonomous data collection, this is challenging. Results from the local map containing the query object are left out in order to not let matching that same observation influence the score.



(a) Initial chair used for training segmentations. (b) [5] segm. of ground truth chair in other map. (c) Convex segmentation of chair in other map. (d) [5] top segment overlap with ground truth. (e) Incremental top overlap with ground truth.

Fig. 6. Example result of the different segmentation methods when trained on one chair and applied to another. From left to right: the training observation, the resulting segmentations in another map and the segments with the highest overlap to the segment in the new map. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 1
Mean overlap ratios for the segments with the highest overlap with an instance. All three methods have been evaluated on 816 annotated instances within 88 local maps. The mean is computed as the mean overlap ratio among all evaluated object examples.

Instance	Backpack	Trashbin	Lamp	Chair	Desktop	Pillow	Sweater	Boiler	All
Mean size (dm^3)	20	10	10	50	10	10	20	10	20
	Segment overlap ratio								mean
Convex segments	0.50	0.42	0.68	0.47	0.74	0.65	0.33	0.40	**0.51**
Incremental convex	0.47	0.40	0.68	0.51	0.68	0.57	0.32	0.39	**0.51**
Finman et al. [5]	0.54	0.37	0.69	0.49	0.73	0.62	0.44	0.31	**0.52**

5. Results

5.1. Segmentation evaluation

In the segmentation comparison, we observe that the convex segmentation performs comparably to the method of [5], with the mean overlap ratio of the latter being slightly better, see Table 1. Particularly for the sweater objects, the difference is more significant. This is to be expected, since the only instance in this category is mostly hanging flat against a wall. It is therefore hard to segment based on shape, and most likely the method of [5] optimized the parameters to rely on color only, yielding better results. The methods perform comparably on the chair category. Since the convex segmentation will segment any general chair into a seat and a back rest, it suggests that [5] will most often fail to segment a chair as one object if it is uniformly colored. Fig. 6 illustrates an example of this. Both our convex segmentation and [5] divides the object into several pieces, the largest being the back rest. The incremental approach manages to combine the convex segments into a more complete chair.

When comparing the convex segmentation to the incremental approach (again, see Table 1), we observe that mean overlap ratio stays the same. In many of the categories the ratios of the incremental segmentation are slightly lower. Importantly however, in the chair category it performs better by a significant margin.

5.2. Retrieval evaluation

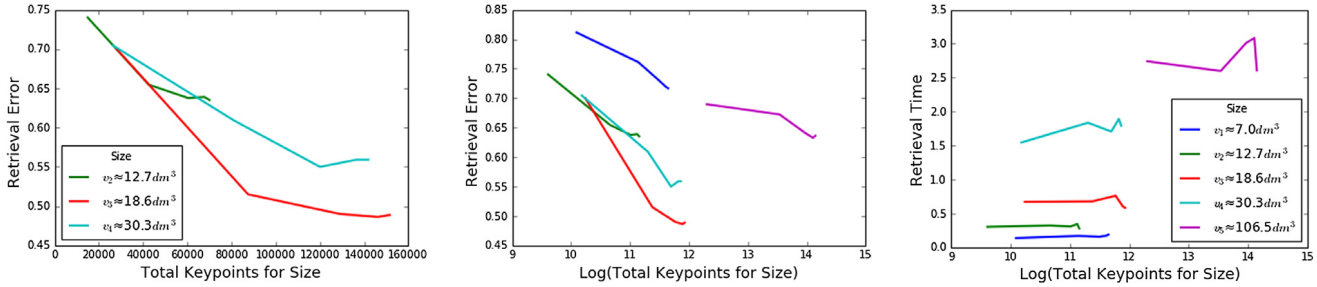
To benchmark the retrieval performance, we ran queries on all the annotated instances in the KTH dataset. We noticed that the method in [5] performs considerably slower than the other methods considered, which prompted us to perform a second test on a subset of the annotated instances in the KTH dataset. As mentioned earlier, the KTH dataset consists of 149 local maps, out of which approximately 67% are collected in one room, while the remainder are collected in five other rooms. We ran the second test mostly on annotated instances from the other rooms captured in the dataset, which entails greater light and viewpoint variation, thus making this second experiment a more challenging one. In the following, we refer to *convex* as our method when querying strictly on the convex segments, *incremental* when also applying the growing scheme and *re-weighted* when re-weighting the incremental method for a final result. A match is returned if the largest *overlap ratio* (see Section 3.7) of an annotation with a query result exceeds 0.25 in one of the frames of the local map. The results are summarized in Table 2, with the figures enclosed in brackets denoting results obtained during the second test (i.e. on a subset of the data).

In our evaluation, we found that the relevance of the retrieved results depends to a large extent on the quality of the input query. This holds for all of the methods, however some are more robust than others. The results of our convex method is largely similar to that of our incremental approach, see Table 2. The main difference lies in that the incremental approach performs significantly better on the chair category. The V-LAD method as applied to the convex segments with PFHRGB features performs consistently worse than the corresponding VT methods.

Regarding the comparison with [5], it seems the method is suffering from noise in the data. The reliance on global features also means that the limited view coverage in the local maps is problematic. Although the segmentation was better than our convex segmentation, the recognition consistently performs worse on this data. We have seen that it performs well on smaller objects with distinct shapes that are observed close to the camera. In those cases the results are often very good. However, when there is noise or the shape or color are less distinct, the performance drops.

We note that the re-weighted method improves results slightly for most of the objects, and the mean retrieval score. However, we do not see the large gains that we observed in [10]. The largest improvement are seen in the categories which typically have more distinctive features, such as the jacket and the backpack. The largest decline is seen in the desktop category. The one instance in this category is a black computer, making registration hard.

We see a significant drop in the performance of all the methods between the two tests performed. We argue that this is the case due to the nature of the data—in the first test, the annotated instances from the first part of the dataset (i.e. observations from one



(a) Error for the smaller size groups as a function of total number of keypoints.

(b) Error as a function of the logarithm of the total number of keypoints.

(c) Query time as a function of the logarithm of the total number of keypoints.

Fig. 7. Each line represents a size $v \in V$. v_0 is left out as there were not sufficiently many query objects of that size for our analysis. These plots show the errors (one minus retrieval score) and the query times for objects in each size interval when we change the keypoint density for that class. The density is measured by the total number of keypoints belonging to all segments of that size in the vocabulary tree. Note the logarithmic scale on the last two plots. Time is entirely dependent on keypoint extraction, which takes more time for larger segments. The errors level out at about 10–20 keypoints/dm^3, corresponding to about 60k, 130k and 120k total features in the vocabulary tree for the sizes v_2, v_3 and v_4 respectively.

Table 2
The mean retrieval score for the different methods. The results on the subset of 400 objects are within brackets. The score is defined as the ratio of correct instances among the top ten results. The mean is computed as the mean retrieval score among all queries. Note that the reported time includes time required to visually render the top results.

Category	Backpack	Trashbin	Lamp	Chair	Desktop	Pillow	Jacket	Boiler	Total	
Queries	189(61)	82(35)	77(11)	426(209)	85(17)	96(30)	75(22)	22(22)	1052	
Instances	3	1	1	3	1	2	3	1	14	
	Retrieval score								mean	time
Convex	0.54 (0.52)	0.56 (0.43)	0.94 (0.93)	0.39 (0.30)	0.59 (0.51)	0.69 (0.46)	0.54 (0.50)	0.31 (0.31)	**0.54** (**0.39**)	6.2 s 6.2 s
Incremental	0.61 (0.50)	0.55 (0.41)	0.94 (0.93)	0.47 (0.36)	0.56 (0.49)	0.68 (0.43)	0.51 (0.46)	0.31 (0.31)	**0.56** (**0.41**)	10 s 10 s
Re-weighted	0.64 (0.54)	0.59 (0.46)	0.94 (0.93)	0.46 (0.34)	0.52 (0.52)	0.64 (0.44)	0.59 (0.56)	0.35 (0.35)	**0.57** (**0.42**)	21 s 21 s
V-LAD	0.56 (0.47)	0.53 (0.38)	0.92 (0.95)	0.28 (0.24)	0.38 (0.28)	0.64 (0.49)	0.54 (0.47)	0.30 (0.30)	**0.46** (**0.34**)	5.2 s 5.2 s
[5][a]	(0.21)	(0.23)	(0.32)	(0.14)	(0.28)	(0.03)	(0.09)	0.14	(**0.16**)	120 s

[a] Measured only on the subset.

room) dominate. These instances are subject to natural variations in position and orientation as are typical of most office environments, for example chairs move in well-defined spaces in front of desks. The same holds for e.g. lamps and desktops on tables. The second test mostly contains instances which have been subjected to an artificial perturbation, as they have been deliberately placed around and moved each time between observations. This variation in position, and hence illumination, makes the problem much harder and leads to lower retrieval results overall.

5.3. Keypoint density adaptation

We studied different combinations of functions $\mathbf{f_d}$ for mapping segment sizes to numbers of keypoints using Eq. (3). The equation returns the error as the inverse of the retrieval score, i.e. one minus the score. A subset of those results is presented in Fig. 7. Most importantly, when querying for objects of a particular size $v \in V$ we found that there is not a clear connection between the query duration and the number of features of the other objects $w \neq v, w \in V$ stored in the VT representation. Instead, the extraction of features from the query object dominates, see Fig. 7(c). Note that the major factor is the number of points in the query objects rather than the number of features, explaining why the time stays relatively constant within one size class. Also, at least for this limited dataset, the retrieval score of one size is not impacted negatively by adding more features to the other sizes. However, as can be seen in Fig. 7(b), we can add several orders of magnitude more features for the largest size than for the other ones without significantly improving the error for that size. For the smaller sizes, adding keypoints improves rates up to about 10–20 keypoints/dm^3 when the errors level out, see Fig. 7(a).

In the segmentation experiments we noticed that the keypoint distribution of the larger segments might have an effect on the segmentation. If the keypoint density is too low ($\sim$1/dm^3), the resulting vocabulary vector will have a very small contribution when added to another vector. It might therefore happen that the incremental scheme adds one such segment as it has negligible effect.

5.4. Compression & execution time

The vocabulary generated from the KTH dataset occupies 298 MB on disk. The pre-computed vocabulary vectors occupy about 110 MB. This should be compared to the 11.8 GB of the original point clouds making up the PTU sweeps. Since the majority of the vocabulary representation consists of the cluster center vectors of the tree nodes, adding new maps will mostly grow the size of the cached vectors. Since we have a total of 149 local maps, adding a new one adds about $\sim$0.6 MB. This shows that the size of the vocabulary is not an issue and that the system is able to integrate several months of observations.

In Table 2, we also present the mean query time of the different methods. As we can see, all the proposed methods return a query result within a matter of seconds.

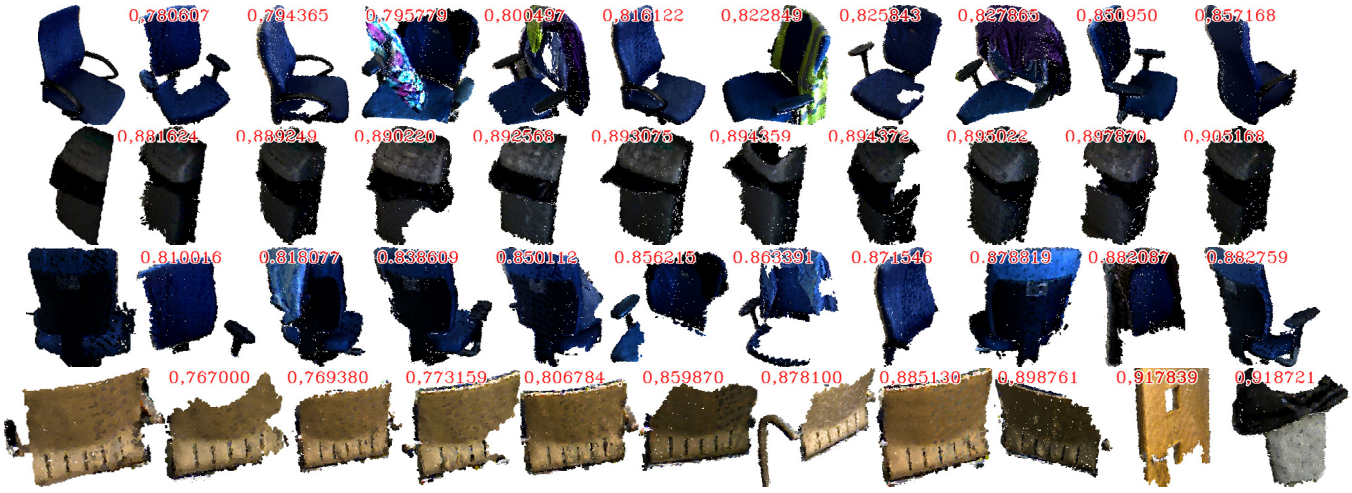


Fig. 8. Queries on the UK office dataset. One result of the queries is an image of the top matches in the form of shaded surfel clouds. These four different queries (one for each row) show the query objects on the far left together with the top ten results (left to right). On top of the results are the vocabulary distances to the query vector (red). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

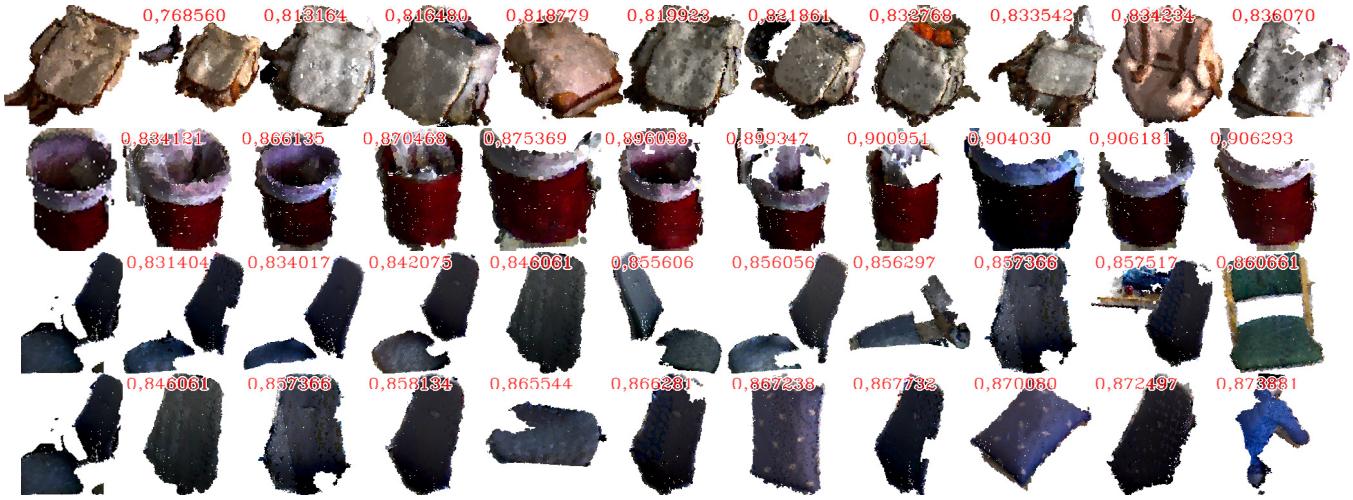


Fig. 9. Queries on the KTH dataset: One backpack, one trash bin and two queries for the same chair. The first chair query is using the incremental segmentation while the second one is matching only based on the convex segments. One additional chair of the same type as the query is present in the top matches when using the incremental segmentation. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

5.5. Qualitative results

In Figs. 8 and 9 we can see some examples of queries on the UK Office data and KTH data respectively. In the UK office data we noticed that performance varied between different categories. The results for the trash bins for instance are very good, since they are often perfectly segmented and with distinctive features. Some of the chairs posed problems since they are often only partially observed. The same thing can be said for the KTH objects. Parts that are observed up close in good lighting consistently provide better queries.

6. Discussion

As stated, noise and other artifacts in the data make our stated problem a challenge. However, as we have seen, we can rely on our retrieval system having a performance of around 50% in natural environments. This enables us to narrow down the search in our data and use costly techniques such as GO-ICP [42] to verify the matches. We conclude that the performance of our convex segmentation rivals the adaptive approach of [5], enabling us to match segments to the query object. However, a convex segmentation will always have problems with certain types of objects. The incremental segmentation aims to solve this problem. As we saw in the results, this approach significantly improved both segmentation and retrieval performance for the non-convex object category found in our dataset; the chairs. It is important to note that retrieval performance for the other categories does not drop when using the incremental segmentation, which might very well have been the case. Instead, we observe that the incremental segmentation mostly decides to grow a segment when it is not covering the object already. An example of how the incremental approach might be beneficial for chairs can be seen in Fig. 9. By having both the back rest and the seat available for comparison, the performance can be improved. The re-weighting scheme showed some improvement in the categories that typically have more distinctive features. The system therefore seems to be reliant on good features to register the objects well. When this is the case, as seen in the jacket category and in [10], re-weighting can significantly boost performance.

In the analysis of the density of keypoints, we concluded that higher densities do not affect the retrieval time significantly. This is in fact a merit of the representation, something which will allow timely results even when having months of observations in the

vocabulary tree. We further saw that for the largest surfaces, there is little reason for having the same keypoint density as for small ones. As can be seen in Fig. 7(b), an order of magnitude more features would be added if we keep keypoint density the same as for other sizes. In order for the system to be scalable and keep its compactness, limiting the number of keypoints in these areas is recommendable.

7. Conclusion

We have presented a system for retrieving object instances from 3D maps. Given a single query object, the algorithm looks for matches using hierarchical matching of local 3D features. The system performs segmentation of the maps simultaneously with querying, enabling retrieval of potentially complex shapes. Experiments comparing the segmentation and retrieval to similar systems show good results on the kind of data one can expect from an autonomous system.

A potential extension of the proposed framework would be to allow temporal indexing of the data, which we hope would enable more complex use cases. One would for example be able to check if a certain computer was still present after work hours. By chaining other predicates such as location or presence of other objects, this would allow intuitive inspection of the data. This is something that we think is lacking in present non-semantic map representations. Our system allows the data to be directly and easily inspected without introducing category labels or training on different object categories.

Once we have built a surfel map using the ElasticFusion framework [33], we can improve on the accuracy in parts of the map by moving closer and continuously registering and updating the map. An opportunity to use this further in our system would be to reinspect an object if we fail to find good results when first querying. A more precise model would likely yield better results. In addition to intelligently updating the maps, we also look to combine the query results to build high quality 3D models. This might allow for iterative querying with increasingly good models, enabling better results.

Code for the complete pipeline is available at https://github.com/strands-project/strands_3d_mapping/tree/hydro-devel/dynamic_object_retrieval. The data is part of the open *KTH Long Term Dataset*[1] and the *G4S Y2 Dataset*, which is available upon request.

Acknowledgments

The authors would like to thank Ross Finmann for letting us use his code in our experiments. The work presented in this paper has been funded by the European Union Seventh Framework Programme (FP7/2007–2013) under grant agreement No 600623 (“STRANDS”).

References

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

[2] J. Mason, B. Marthi, R. Parr, Unsupervised discovery of object classes with a mobile robot, in: 2014 IEEE International Conference on Robotics and Automation, (ICRA), IEEE, 2014, pp. 3074–3081.

[3] L.-C. Caron, D. Filliat, A. Gepperth, Neural network fusion of color, depth and location for object instance recognition on a mobile robot, in: 2014 European Conference on Computer Vision Workshops, (ECCV Workshops), Springer, 2014, pp. 791–805.

[4] A. Karpathy, S. Miller, L. Fei-Fei, Object discovery in 3D scenes via shape analysis, in: 2013 IEEE International Conference on Robotics and Automation, (ICRA), IEEE, 2013, pp. 2088–2095.

[5] R. Finman, T. Whelan, M. Kaess, J.J. Leonard, Toward lifelong object segmentation from change detection in dense RGB-D maps, in: 2013 European Conference on Mobile Robots, (ECMR), IEEE, 2013, pp. 178–185.

[6] M. Schoeler, J. Papon, F. Wörgötter, Constrained planar cuts-object partitioning for point clouds, in: 2015 IEEE Conference on Computer Vision and Pattern Recognition, CVPR, 2015, pp. 5207–5215.

[7] K.M. Wurm, D. Hennes, D. Holz, R.B. Rusu, C. Stachniss, K. Konolige, W. Burgard, Hierarchies of octrees for efficient 3D mapping, in: 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, (IROS), IEEE, 2011, pp. 4249–4255.

[8] T. Whelan, L. Ma, E. Bondarev, P. de With, J. McDonald, Incremental and batch planar simplification of dense point cloud maps, Robot. Auton. Syst. 69 (2015) 3–14.

[9] D. Nister, H. Stewenius, Scalable recognition with a vocabulary tree, in: 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Vol. 2, (CVPR), IEEE, 2006, pp. 2161–2168.

[10] N. Bore, P. Jensfelt, J. Folkesson, Retrieval of arbitrary 3D objects from robot observations, in: 2015 European Conference on Mobile Robots, (ECMR), IEEE, 2015, pp. 1–8.

[11] B.E. Prasad, A. Gupta, H.-M.D. Toong, S.E. Madnick, A microcomputer-based image database management system, IEEE Trans. Ind. Electron. 1 (1987) 83–88.

[12] J. Sivic, A. Zisserman, Video google: A text retrieval approach to object matching in videos, in: 2003 IEEE International Conference on Computer Vision, (ICCV), IEEE, 2003, pp. 1470–1477.

[13] D.G. Lowe, Object recognition from local scale-invariant features, in: 1999 IEEE International Conference on Computer vision, Vol. 2, (ICCV), IEEE, 1999, pp. 1150–1157.

[14] K. Grauman, T. Darrell, The pyramid match Kernel: Efficient learning with sets of features, J. Mach. Learn. Res. 8 (2007) 725–760.

[15] J. Philbin, O. Chum, M. Isard, J. Sivic, A. Zisserman, Object retrieval with large vocabularies and fast spatial matching, in: 2007 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, (CVPR), IEEE, 2007, pp. 1–8.

[16] H. Jégou, M. Douze, C. Schmid, P. Pérez, Aggregating local descriptors into a compact image representation, in: 2010 IEEE Conference on Computer Vision and Pattern Recognition, (CVPR), IEEE, 2010, pp. 3304–3311.

[17] R. Arandjelovic, A. Zisserman, All about vlad, in: 2013 IEEE Conference on Computer Vision and Pattern Recognition, (CVPR), IEEE, 2013, pp. 1578–1585.

[18] J. Wolf, W. Burgard, H. Burkhardt, Robust vision-based localization by combining an image-retrieval system with monte carlo localization, IEEE Trans. Robot. 21 (2) (2005) 208–216.

[19] Y. Ishikawa, R. Subramanya, C. Faloutsos, Mindreader: Querying databases through multiple examples, in: Proc. of the 24:th VLDB Conference, 1998.

[20] M.L. Kherfi, D. Ziou, Relevance feedback for cbir: a new approach based on probabilistic feature weighting with positive and negative examples, IEEE Trans. Image Process. 15 (4) (2006) 1017–1030.

[21] Y. Wu, A. Zhang, A feature re-weighting approach for relevance feedback in image retrieval, in: 2002 International Conference on Image Processing, Vol. 2, (ICIP), IEEE, 2002, pp. II-581–II-584.

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

[23] J.W. Tangelder, R.C. Veltkamp, A survey of content based 3D shape retrieval methods, Multimedia Tools Appl. 39 (3) (2008) 441–471.

[24] R.B. Rusu, G. Bradski, R. Thibaux, J. Hsu, Fast 3D recognition and pose using the viewpoint feature histogram, in: Intelligent Robots and Systems, (IROS), IEEE, 2010, pp. 2155–2162.

[25] W. Wohlkinger, M. Vincze, Shape-based depth image to 3D model matching and classification with inter-view similarity, in: 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, (IROS), IEEE, 2011, pp. 4865–4870.

[26] R. Osada, T. Funkhouser, B. Chazelle, D. Dobkin, Matching 3D models with shape distributions, in: 2001 International Conference on Shape Modeling and Applications, (SMI), IEEE, 2001, pp. 154–166.

[27] M.A. Asari, E. Supriyanto, U.U. Sheikh, The evaluation of shape distribution for object recognition based on Kinect-like depth image, in: Computational Intelligence, Communication Systems and Networks, (CICSyN), IEEE, 2012, pp. 313–318.

[28] A. Aldoma, F. Tombari, L. Di Stefano, M. Vincze, A global hypotheses verification method for 3D object recognition, in: 2012 International Conference on Computer Vision, (ICCV), Springer, 2012, pp. 511–524.

[29] C. Redondo-Cabrera, R.J. Lopez-Sastre, J. Acevedo-Rodríguez, S. Maldonado-Bascón, Surfing the point clouds: Selective 3D spatial pyramids for category-level object recognition, in: Computer Vision and Pattern Recognition, (CVPR), IEEE, 2012, pp. 3458–3465.

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

[31] J. Papon, A. Abramov, M. Schoeler, F. Worgotter, Voxel cloud connectivity segmentation-supervoxels for point clouds, in: Computer Vision and Pattern Recognition, (CVPR), IEEE, 2013, pp. 2027–2034.

[1] https://strands.pdc.kth.se/public/KTH_longterm_dataset_labels/readme.html, https://strands.pdc.kth.se/public/KTH_labelled_moving_objects/readme.html

[32] R. Ambrus, J. Ekekrantz, J. Folkesson, P. Jensfelt, Unsupervised learning of spatial-temporal models of objects in a long-term autonomy scenario, in: 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems, (IROS), IEEE, 2015, pp. 5678–5685.
[33] T. Whelan, S. Leutenegger, R.F. Salas-Moreno, B. Glocker, A.J. Davison, ElasticFusion: Dense SLAM without a pose graph, in: Robotics: Science and Systems (RSS), Rome, Italy, 2015.
[34] M. Stoer, F. Wagner, A simple min-cut algorithm, J. ACM 44 (4) (1997) 585–591.
[35] R.B. Rusu, Z.C. Marton, N. Blodow, M. Beetz, Learning informative point classes for the acquisition of object model maps, in: 10th International Conference on Control, Automation, Robotics and Vision, (ICARCV), IEEE, 2008, pp. 643–650.
[36] L.A. Alexandre, 3D descriptors for object and category recognition: a comparative evaluation, in: 2012 IEEE/RSJ International Conference on Workshop on Color-Depth Camera Fusion in Robotics at Intelligent Robots and Systems, IROS, Citeseer, 2012.
[37] F. Tombari, S. Salti, L. Di Stefano, A combined texture-shape descriptor for enhanced 3D feature matching, in: 2011 International Conference on Image Processing, (ICIP), IEEE, 2011, pp. 809–812.
[38] Y. Zhong, Intrinsic shape signatures: A shape descriptor for 3D object recognition, in: 2009 IEEE 12th International Conference on Computer Vision Workshops, (ICCV Workshops), IEEE, 2009, pp. 689–696.
[39] F. Tombari, S. Salti, L. Di Stefano, Performance evaluation of 3D keypoint detectors, Int. J. Comput. Vis. 102 (1–3) (2013) 198–220.
[40] S. Filipe, L.A. Alexandre, A comparative evaluation of 3D keypoint detectors, in: 9th Conference on Telecommunications, Conftele, 2013, pp. 145–148.
[41] C. Rother, V. Kolmogorov, A. Blake, Grabcut: Interactive foreground extraction using iterated graph cuts, in: ACM Transactions on Graphics, Vol. 23, (TOG), ACM, 2004, pp. 309–314.
[42] J. Yang, H. Li, Y. Jia, Go-icp: Solving 3D registration efficiently and globally optimally, in: 2013 IEEE International Conference on Computer Vision, (ICCV), IEEE, 2013, pp. 1457–1464.

Nils Bore received the degree of M.Sc. in Mathematical Engineering from Lund University, Faculty of Engineering in 2012. He is currently a Ph.D. candidate at the Centre for Autonomous Systems and the Computational Vision Active Perception lab at KTH Royal Institute of Technology. His research interests include mapping, unsupervised learning, 3D perception and robot navigation.

Rares Ambrus received the M.Sc. degree from Jacobs University in 2008. Since 2013 he is a Ph.D. student at KTH Roy Institute of Technology within the Computational Vision Active Perception lab. His research interest include localization, mapping, object segmentation and semantic scene understanding from a spatial and temporal point of view.

Patric Jensfelt received the degree of M.Sc. in Engineering Physics from KTH Royal Institute of Technology in 1996 and a Ph.D. in Automatic Control in 2001. Since 2012 he is a professor of Computer Science at the Centre for Autonomous Systems and the Computational Vision Active Perception lab. His main research interests include navigation, localization, mapping, spatial understanding and system integration for robotic systems.

John Folkesson received the B.A. in Physics from Queens College CUNY in 1983, M.S. in Computer Science, 2001 and Ph.D. in Robotics, 2006 from Royal Institute of Technology—KTH. Currently he is Associate Professor of Robotics at KTH in the department of Computational Vision Active Perception and the Center for Autonomous Systems. His research is in navigation, mapping, perception and situation awareness for autonomous driving, indoor robots, and underwater robots.