Toward Object-based Place Recognition in Dense RGB-D Maps

Ross Finman[1], Liam Paull[1], and John J. Leonard[1]

***Abstract*— Longterm localization and mapping requires the ability to detect when places are being revisited to "close loops" and mitigate odometry drift. The appearance-based approaches solve this problem by using visual descriptors to associate camera imagery. This method has proven remarkably successful, yet performance will always degrade with drastic changes in viewpoint or illumination. In this paper, we propose to leverage the recent results in dense RGB-D mapping to perform place recognition in the space of *objects*. We detect objects from the dense 3-D data using a novel feature descriptor generated using primitive kernels. These objects are then connected in a sparse graph which can be quickly searched for place matches. The developed algorithm allows for multi-floor or multi-session building-scale dense mapping and is invariant to viewpoint and illumination. We validate the approach on a number of real datasets collected with a handheld RGB-D camera.**

I. Introduction

As robots become more capable of exploring their environments, they collect more data over longer stretches about their surroundings. With this new capability comes the need to determine if the robot has returned to the same place, which can be challenging due to the accumulation of odometry error with time. *Place recognition* is used in simultaneous localization and mapping (SLAM) systems to identify loop closures to correct for map drift. Much previous work has been done with matching lidar scans [1] or camera data [2], [3]. However, recently there have been many algorithmic advances in dense RGB-D SLAM [4]–[7] that combine raw data into rich 3-D maps. We argue here that this data domain has the higher potential for robustly detecting loop closures particularly in the context of variable viewpoints and lighting.

Other attempts such as [8] to use objects as features for SLAM on the space of objects have only considered matching of individual 2.5D RGB-D frames. These methods have shown good results but trajectory and lighting invariance are still a challenge since the RGB data is used. Our approach is to extract objects from the full 3-D dense maps generated in real-time using Kintinuous [7]. However, directly matching places on this massive amount of data is computationally infeasible. Therefore we propose a pipeline

The authors thank Thomas Whelan of Imperial College London for his implementation of Kintinuous.

This work was partially supported by the Office of Naval Research under grants N00014-10-1-0936, N00014-11-1-0688 and N00014-13-1-0588 and by the National Science Foundation under grant IIS-1318392, which we gratefully acknowledge.

[1]R. Finman, L. Paull, and J. J. Leonard are with the Computer Science and Artificial Intelligence Laboratory (CSAIL), Massachusetts Institute of Technology (MIT), Cambridge, MA 02139, USA. {rfinman, lpaull, jleonard}@mit.edu

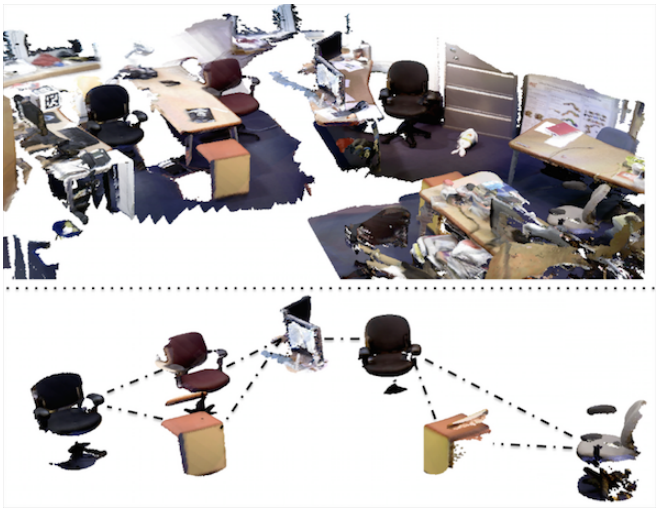

Fig. 1. An object graph being built from a map. **Top:** a full dense RGB-D map of a cluttered office scene. **Bottom:** The object graph extracted from the office scene above. The objects detected are chairs, small file cabinets, and computer monitors. As new objects are detected, they are used to compare new places against the existing object graph.

of object detection, followed by object graph creation and matching to solve the place recognition problem in full 3-D.

Many previous works on object detection from 3-D data, such as our own work [9], [10] and others [11] have operated on point cloud segmentations. However, after extensive experimentation, our conclusion is that these segmentation-based methods are not sufficiently invariant to sensor coverage for viewpoint and lighting independent place recognition. Instead, we propose to detect objects using a novel high-level feature descriptor for objects and use those objects for place recognition (Fig. 2). These feature descriptors are generated by convolving the point cloud with a number of pre-generated primitive kernel patches. These kernels are flat planes, corners, edges and other basic shapes. Object feature descriptors for a few objects are calculated offline, comprising of distributions over the kernel. The result of this convolution with the feature descriptor is a heat map over the point cloud, which is then thresholded to detect if any new instances of the known objects are found.

Once objects are found, they are connected into a graph, shown in Figure 1. As clusters of objects are detected, they are compared against the full object graph using a statistical graphical matching approach. If matches are found, a loop closure is initiated which can then be followed by a mesh deformation to correct the map [12].

One interesting point of consideration in this context

is the definition of a ”place”. Standard appearance-based place recognition work such as [3] defines a place as the location where a camera frame was taken, but the output data from dense mapping methods (such as Kintinuous) is asynchronous, variably sized sets of points and therefore the definition should be reconsidered in this context. We define a place as a location where a set of objects could be observed. This differs from the 2-D frame definition in that objects could be occluded from one another from any single viewpoint, not seen within the field of view of the camera along the trajectory, or seen along vastly different trajectories.

The contributions of this paper are twofold. First, we present a non frame-based object descriptor that operates on dense 3-D models using kernel parts for object-based place recognition. The technique is invariant to changes in lighting and easily incrementalized. Second, we present a sparse metrical object graph that can be efficiently used for performing place recognition. We evaluate our method qualitatively on several real-world data sets, demonstrating the flexibility of this framework.

II. Related Work

The work here is multidisciplinary in scope. We will try to summarize some of the more closely related and important works in the areas of object recognition, 3-D SLAM, and place recognition.

A. Object Detection

Object recognition, both in robotics and computer vision is a problem that spans decades [13], [14]. We narrow our scope of review to 3-D object recognition methods. An early 3-D descriptor, spin-images, was developed by Johnson and Hebert [15] which is still used today. More recently, Rusu *et al.* [16] efficiently builds a fast point feature histogram of two-point descriptors for points in the neighborhood of a point. Tombari *et al.* [17] developed a variation on the popular SHOT descriptor to incorporate texture. Bo *et al.* [18] designed kernel descriptors which give a general design pattern for local feature responses which have led to promising results. While these approaches have proved useful, for place recognition these low-level descriptors lack the richness of full objects. So while these features can be used to find objects, we are looking for objects to find places. Other more recent related work by Herbst *et al.* developed a method for online object discovery in 3-D maps, thus introducing a way for robots to automatically learn about the objects in their environment [19].

To match 3-D objects, an obvious approach is to match the 3-D structure. One widely used approach for matching 3-D geometry is the Iterative Closet Point (ICP) algorithm. ICP iteratively finds the transformation between two point clouds that minimizes the error between the points given an initial starting transformation. This works well in practice if an object is extracted cleanly in a scene and all objects are observed with the same coverage as the model being referenced. However, since the algorithm matches all the points, it is not robust to partial views. Furthermore, running ICP for every possible subset of points in a map is computationally intractable. We propose a method to score an object based on how well a pre-defined set of kernels match across all points in an object.

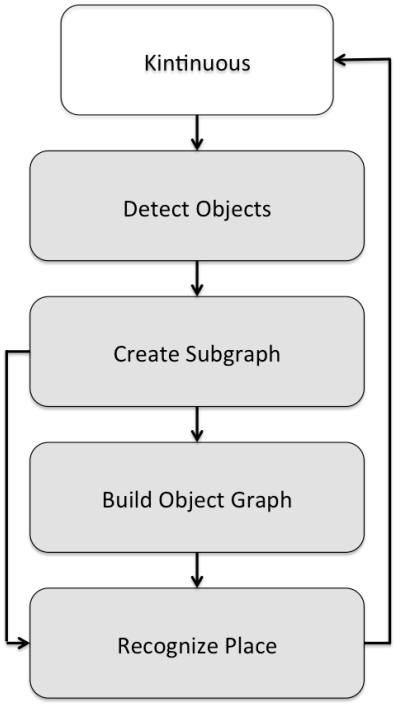

Fig. 2. System architecture diagram. Raw RGB-D point cloud data is fed in from Kintinuous [12] into an object detection window that runs detectors on recently added data. Any objects discovered are joined together into subgraph *places*, and those places are compared against previous places in the object graph. Any detections are then fed back into Kintinuous. This system outputs both a loop closure constraint between two parts of a map, and a map of objects.

B. 3-D SLAM

3-D SLAM has seen a recent explosion in robotics. For example, Newcombe *et al.*'s [5] KinectFusion has opened the door for dense mapping with RGB-D sensors by exploiting a truncated signed distance function representation of free space and massively parallel processing. Whelan *et al.* [12] further advance the method by allowing the mapping cube to move and therefore enable the mapping of much larger areas and show how to deform the mesh upon detecting loop closures. These works have laid the foundation for 3-D SLAM. For example, Choudhary *et al.* [20] worked on object discovery in SLAM maps for the purpose of closing loops. Perhaps the most similar formulation of our method is SLAM++ by Salas-Moreno *et al.* [8], which describes a traditional SLAM framework except using objects as landmarks instead of low-level point features. They use a frame-based object detector (based on [21]) to create objects, which are used as landmarks for SLAM. Our method differs in several ways. We are not directly solving the SLAM problem, only determining place recognition. We create a metric, object-based representation of the world only if objects are densely

connected, and otherwise keep a sparse graph of objects. Lastly, we take a different approach by working directly and only with the map and not the RGB-D frames.

C. Place Recognition

The seminal work by Cummins *et al.* [22] on fast appearance based mapping (FABMAP) was one of the first to pose SLAM in a topological context where loop closures are detected through place recognition and is still among the state of the art for these types of methods. In effect, they train a dictionary of images that can be efficiently queried. Many works have attempted to either robustify this approach to weather and/or lighting [23], reduce the size of the required dictionary [24], and learn the dictionary online [25]. These approaches, while working well with RGB cameras, do not have 3-D information and have a lower-level representation, and will necessarily be somewhat dependent on illumination due to the sensing modality. One interesting approach is Paul *et al.* [26], which is a 3-D version of FABMAP for place recognition, but which still uses point features in frames. Our work detects objects within a map using the object's geometry (not visual appearance) and does place recognition using a rich object graph.

III. Object Detection

This section discusses the object detection framework used to find objects for subsequent graph matching in Section IV. We propose a method to score an object based on how well a pre-defined set of kernels match across all points in an object. The general approach is to treat objects as uniquely composed of primitive shapes such as flat sections, round sections, corners etc. If this unique object representation can be pre-loaded with one instantiation of an object then subsequent objects of the same type can be detected from the incoming RGB-D.

To begin, an object $\mathcal{O}$ is defined as

$$\mathcal{O} \triangleq \{r_{\mathcal{O}}, d_{\mathcal{O}}, \mathcal{H}_{\mathcal{O}}\} \tag{1}$$

with a bounding sphere of radius $r_{\mathcal{O}}$ around the center, points $d_{\mathcal{O}}$ within the bounding sphere, and kernel feature histogram $\mathcal{H}_{\mathcal{O}}$ (discussed below). Similarly to the ubiquitous bounding box used in computer vision object recognition, we use a bounding sphere to represent our object. Spheres have the convenient property of being rotationally invariant and efficiently searchable with a K-D tree representation since the 3-D points are unstructured (i.e. not a depth frame).

The kernel feature histogram, $\mathcal{H}$ is an $\alpha \times |K|$ matrix, where K is the set of all kernels and α is the resolution of the histogram. A kernel k is defined as $k \triangleq \{r_k, d_k\}$ where all points d_k are within a bounding sphere with radius r_k. Examples of primitive kernels include flat planes, corners, edges, curves, and random noise, although in general we can choose any shape. We choose $r_k \ll r_{\mathcal{O}}$, so $d_k \ll d_{\mathcal{O}}$ since we use the same resolution. We convolve each kernel over all data points as detailed in Algorithm 1. The result is a set of scores for every point with a lower score signifying a closer match. For simplicity, we assume $S_k \sim \mathcal{N}(\mu_k, \sigma_k^2)$.

Algorithm 1 Kernel Scoring

Input: D: Set of data points
Input: k_i: Kernel i
Output: S: Set of point scores

$S \leftarrow \emptyset$
for $d \in D$ **do**
  $n \leftarrow \{d^{'} \in D \mid \|d^{'} - d\| < r_{k_i}\}$ // Neighbors of d
  ICP(d_{k_i}, n)
  $S \leftarrow S \cup \{\text{Residual error of ICP}\}$
end for

Algorithm 2 Histogram

Input: S_{k_i}: Set of scores for a kernel k_i
Input: min: Min histogram bound
Input: max: Max histogram bound
Output: $\mathcal{H}^{k_i}$: Histogram of scores for k_i

$\mathcal{H}^{k_i} \leftarrow \mathbf{0}^{\alpha}$ // Initialize histogram with α bins
for $s_i \in S_{k_i}$ **do**
  $$idx \leftarrow \begin{cases} \alpha - 1 & \text{if } s_i > (max) \\ 0 & \text{if } s_i < (min) \\ \left\lfloor \frac{(s_i - (min)) * (\alpha - 1)}{(max - min)} \right\rfloor & \text{otherwise} \end{cases}$$
  $\mathcal{H}^{k_i}[idx] \leftarrow \mathcal{H}^{k_i}[idx] + 1$
end for

Finally, the values of the $\mathcal{H}$ matrix are populated with a histogram of the scores for each kernel. There are α bins over the range $\mu_{s_i} \pm 2\sigma_{s_i}$ of the scores within the object bounding sphere. So $\mathcal{H}$ is a set of binned distributions over kernels. A visualization of the feature descriptor for a chair is shown in Fig. 4 and a detailed description of the object creation is given in Algorithms 1, 2 and 3. Using kernels instead of parts of the object helps with scalability. Our algorithm scales linearly with the number of kernels. The kernels representing all objects are the same. The subject of future work is to learn kernels from the data that are the most discriminative for object sets.

A. Detection

From Kintinuous, there is a continual stream of point cloud data. We want to do object detection on this streaming data, but storing the entire map is unnecessary. We keep a window of points behind the mapping cube for object detection. The size of the window is $2r_{\mathcal{O}_i}^{max}, \forall \mathcal{O}_i \in \mathcal{O}$ - twice the size of the largest object we are detecting. Then, for every point in the window we call the kernel scoring function (Algorithm 1) for every kernel.

Using the kernel heatmaps, we then make an object heatmap. Almost identical to Algorithm 3, we score every point $d \in D$. Using the kernel scores in the surrounding $r_{\mathcal{O}_i}$ radius to build an object model for that point. We build a histogram around a point with the radius $r_{\mathcal{O}_i}$ using the kernel scores (using the object's kernel bounds, and not the local $\mu \pm 2\sigma$). We then find the ℓ_1 distance between the two histograms, with lower values being closer matches. Such values are then thresholded to determine if an object

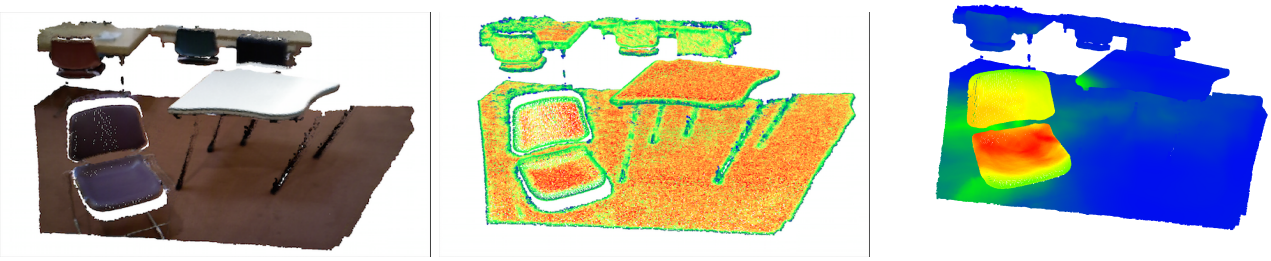

Fig. 3. **Left:** a dense RGB-D map of a sitting area with a table and chair. **Middle:** a heatmap showing the values of the kernel scores from Algorithm 1 with a flat disk kernel. Red corresponds to a high match and blue corresponds to a low match. The flat disk kernel finds local planes in the map. **Right:** An object heatmap that matches based on three additional kernels (edge, corner, and noise) Object matching is done on the heatmap to find the objects based on their scores. The color of each point is determined by the distribution of kernel scores within a bounding sphere centered on the point.

Algorithm 3 Object model creation

Input: D: Set of data points
Input: K Set of kernels
Input: c: Center point
Input: $r_{\mathcal{O}}$: Object radius
Output: $\mathcal{O}$: Object model

1: $S \leftarrow \mathbf{0}^{|K|\times|D|}$
2: $d_{\mathcal{O}} \leftarrow \{d_i \in D |\ \{\|d_i - c\| < \ r\}$
3: **for** $k_i \in K$ **do**
4: $\quad S_{k_i} \leftarrow \{S \cup \textsc{Kernel\_Scoring}(D, k_i)\}$
5: **end for**
6: **for** $k_i \in K$ **do**
7: $\quad s' \leftarrow$ All scores $s_i \in S_{k_i}$ corresponding to $d_{\mathcal{O}}$ points
8: $\quad \mu =\leftarrow$ Mean of s'
9: $\quad \sigma^2 \leftarrow$ variance of s'
10: $\quad \mathcal{H}_{\mathcal{O}}^{k_i} \leftarrow \textsc{Histogram}(S_{k_i}, \mu - 2\sigma, \mu + 2\sigma)$
11: **end for**

is detected or not. The object detection is done on each individual point. Note that since the histogram values add up to the number of points for every kernel, there is an implicit size bias to the feature descriptor as shown mathematically below.

$$\sum_{j=1}^{\alpha} \mathcal{H}_{\mathcal{O}}^{k_{ij}} = |d_{\mathcal{O}}| \ , \forall k_i \in K \tag{2}$$

Intuitively, this is stating that the number of points is encoded within each histogram. Visually, this is summing over a single non-normalized row in Fig. 4. Because we have full 3-D data, normalizing the histogram is unnecessary. In practice, the noise around the edge of a map tends to be detected as objects if the histograms are normalized.

For efficiency in computation, the kernel and object scoring are each parallelized. Furthermore, since the new data being added is small relative to the window, only the points and their neighbors in the window are calculated or recalculated. Neighbors are defined as within r_k^{max} of the new data for kernels and $r_{\mathcal{O}}^{max}$ for objects. We found that subsampling the points used for object detection by 8 still produced satisfactory results. However, due to the small radius of the kernels, they were computed on the fully dense point cloud. Subsampling the kernels produced noise in the

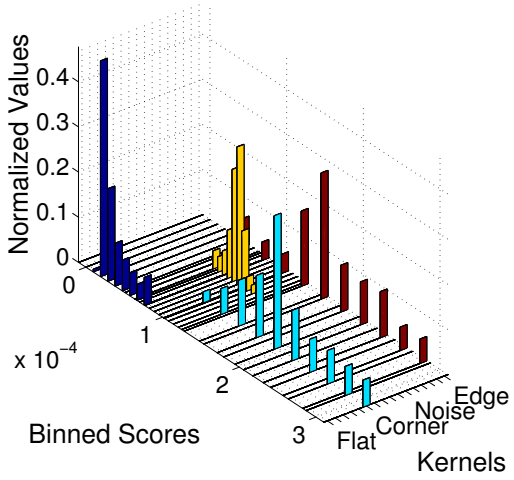

Fig. 4. Histogram for a chair feature descriptor. The x-axis lists the kernels used for the points, with each name describing the general shape of the kernel. The y-axis lists the bins used. The z-axis shows the normalized values of the bins.

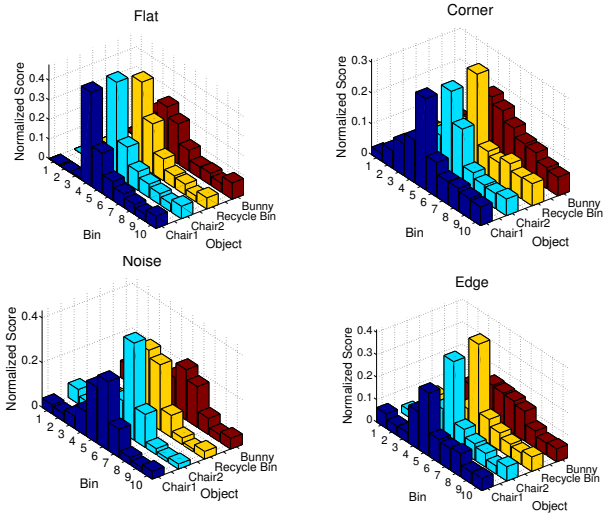

Fig. 5. Sample kernel histograms for four different objects for comparison. Note that some kernels have similar distributions to others depending on the object. The objects used are directly the objects shown in Fig. 9. The distributions are normalized on the y-axis into bins so a direct comparison can be shown, however the distributions vary as shown in the Chair1 histogram in Fig. 4.

detection output.

IV. Graph Formation

Using the objects from Section III, we want to construct an object graph to use for place recognition. Once we have a newly detected object $\mathcal{O}_i$ we create and add object vertex v_i (corresponding to $\mathcal{O}_i$) to an object graph G. G is composed of object vertices V and metric edges E between the vertices. Each vertex is defined as

$$v \triangleq \{c_v, l_v\} \tag{3}$$

where $c_v \in \mathcal{R}^3$ represents the center of the detected bounding sphere and l_v is the label associated with the object (e.g. chair). Note that an object vertex is different from an object since only the center and label are needed.

Using object vertices and their labels as nodes in the graph, we need to decide on which objects to connect. One solution is to use the Kintinuous SLAM result and have metric edges between all objects or all objects within some Euclidean distance. However, this simple solution can lead to problems in the case that the SLAM solution drifts over time, thus leading to more erroneous edge measures between objects that are farther apart. Conversely, having no edges would produce false positives due to aliasing in the case that there can be multiple instantiations of the same object. So we desire some connectivity, in between these two extremes.

We set a threshold e^{max} for the maximum edge length between objects. The value of e^{max} should depend on the sensor used, an Asus Xtreme Pro with a maximum usable depth range of 4m, and the size of the mapping volume (also 4m). As such, we set e^{max} to 4m. The intuition is that if we connect objects mapped together within the same cube, the error due to drift between the observations of the two objects will be negligible. Almost all uncertainty in the edge distance will be induced by the uncertainty in the locations of the objects themselves as described below. We only keep the last e^{max} of object data to connect objects together; hence, if two objects were mapped at different times, but were close to each other in metric space due to differences in coverage or a loop in the sensor trajectory, they would not be joined together. Note that this threshold results in a graph that is not guaranteed to be connected.

For a newly created vertex v_i, we find the set of recently added objects in G within e^{max} distance away and create an edge e between them. Every e_{ij} connecting vertices v_i and v_j is defined as

$$e_{ij} \sim \mathcal{N}(\|c_{v_i} - c_{v_j}\|, \Sigma_v) \tag{4}$$

We model each edge as a distribution centered on the distance between the two object vertices (with both objects adding variance). Determining Σ_v is a difficult problem as described in Fig. 6. Since we already assume the variance in edge length is negligible for $e_{ij} < e^{max}$, all the variance comes from the vertex positions v_c. Variance in the node positions primarily results from the object detection method's determination of the center, as well as a lack of coverage of the object due to the trajectory of the camera. We characterize Σ_v through collecting data across multiple observations and using previously acquired training data.

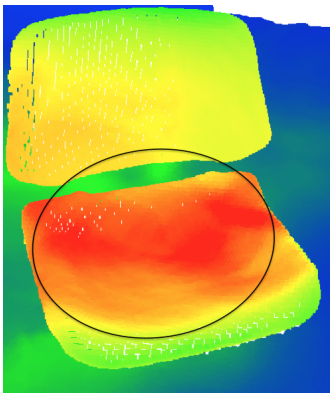

Fig. 6. A visual interpretation of the Σ_v in Eq. 4 for a chair. The object heatmap method we use usually has a wide area where there are matches due to the similarity of the object bounding sphere around each point. We characterize this variance with test data.

V. Graph Matching

After each new vertex, v_i and corresponding edges e_{ij} are added the graph, a search is performed to see if the newly created subgraph, G_1 matches any parts of the full graph G. We build the subgraph G_1 using the following 3 criteria:

- $G_1 \subseteq G$
- $v_i \in V^{G_1}$
- $\exists d \in \mathbf{R}^3, \forall v_j \in V^{G_1}, \|c_{v_j} - d\| < \beta$

Intuitively, G_1 is a graph that contains the new vertex and all object vertices within 2β distance away. Setting β small makes the expected edge variance smaller, but potentially limits the number of objects used to match, which decreases confidence in a match. We set β to be $\frac{e^{max}}{2}$, thus limiting a place to be within the maximum edge distance.

The problem to solve then becomes finding a match for any new place in the entire object graph. Following the formulation from [27], we consider two labeled graphs G_1 and G_2 made up of vertices V_1, V_2 and edge sets E_1, E_2. We say $G_2 = G \setminus G_1$ so that the place cannot be associated with itself. Without loss of generality, we assume $|G_2| \geq |G_1|$, i.e. the graphs can be differently sized.

Let $v_i^{G_1}$ be the i^{th} object vertex of graph G_1. The problem of matching objects from G_1 to objects in G_2 is the equivalent to looking for an $|V_1| \times |V_2|$ *assignment matrix* X such that X_{i_1,i_2} is the indicator variable:

$$X_{i_1,i_2} = \begin{cases} 1 & \text{when } v_{i_1}^{G_1} \text{ is matched to } v_{i_2}^{G_2} \\ 0 & otherwise \end{cases} \tag{5}$$

We are looking for a one-to-many mapping of nodes in G_1 and G_2. So any object in a potential place in G_2 will map to a single object in the place in G_1, thus limiting the search space to reasonable matches (three chairs in a new place won't map to one chair in the object graph, but many chairs in the object graph may map to a single box in the

new place). In other words, we assume that the sums of each column in X are equal to 1:

$$\{X \in \{0,1\}^{N_1 \times N_2} | \sum_{i_1} X_{i_1,i_2} = 1\} \tag{6}$$

Changing the formulation from [27], [28] to include object labels, the graph matching problem can now be formulated as $X^* = \text{argmax}_X \text{score}(X)$ where:

$$\begin{aligned} \text{score}(X) &= \sum_{i_1,i_2,j_1,j_2} H^*_{i_1,i_2,j_1,j_2} X_{i_1,i_2} X_{j_1,j_2} \\ H^*_{i_1,i_2,j_1,j_2} &= H^l_{i_1,i_2,j_1,j_2} * H^r_{i_1,i_2,j_1,j_2} \end{aligned} \tag{7}$$

where H^* is the element-wise multiplication of H^r and H^l. We define the pair potential H^r and the binary label potential H^l as

$$\begin{aligned} H^r_{i_1,i_2,j_1,j_2} &= e^{max} - \|e_{i_1 j_1} - e_{i_2 j_2}\| \\ H^l_{i_1,i_2,j_1,j_2} &= \begin{cases} 1 \text{ if } l_{i_1} = l_{i_2} \ \& \ l_{j_1} = l_{j_2} \\ 0 \ otherwise \end{cases} \end{aligned} \tag{8}$$

Higher values of H^r corresponds to more similar pairs.

However, the graph-matching problem has no known efficient solution in the general case. For the results in this work, we brute force searched through all possibilities and thresholded graphs with $X* > 0.5$. This is done for all subgraphs in the full map (which may be several connected graphs). Brute-force search works for small examples since the size of each new place $G_1 \ll G_2$ as the graph grows larger, but for scaling, approximate methods should be used. We suggest using the spectral method described in [28], with a clearer overview in [27]. Our formulation of the problem including object labels in this section follows the formulation and notation of the latter.

VI. Discussion & Results

For this work, we show qualitative results highlighting our system performing in a variety of maps. At the time of this writing, there are no open-source methods or object-based dense place-recognition datasets available to the authors' knowledge, so quantitative comparisons are difficult. Following [8], our results are best visualized so we refer the reader to our video `http://youtu.be/jqh__XJADwE`.

In Figure 7, we show a single map with two loops. The drift on even a small indoor scene is enough to distort the global map. Our method is able to recognize the objects and further recognize the place over three loops. With four objects, our system can recognize multiple places in the same scene. When the third object is detected, the subgraph is found within the full object graph, and the same place can be re-recognized when the fourth object is found.

In addition to the video results, we highlight some of the benefits of our current method. We discuss the lighting invariance of our method and how that compares to the popular appearance-based place recognition systems. Further, we detail the simple way that our method handles multi-session mapping. Lastly, we show preliminary results for an object discovery method that can allow robots to automatically learn objects in their surroundings directly from our place-recognition method. Our system can handle some moving objects as long as most of the objects used for place-recognition are static.

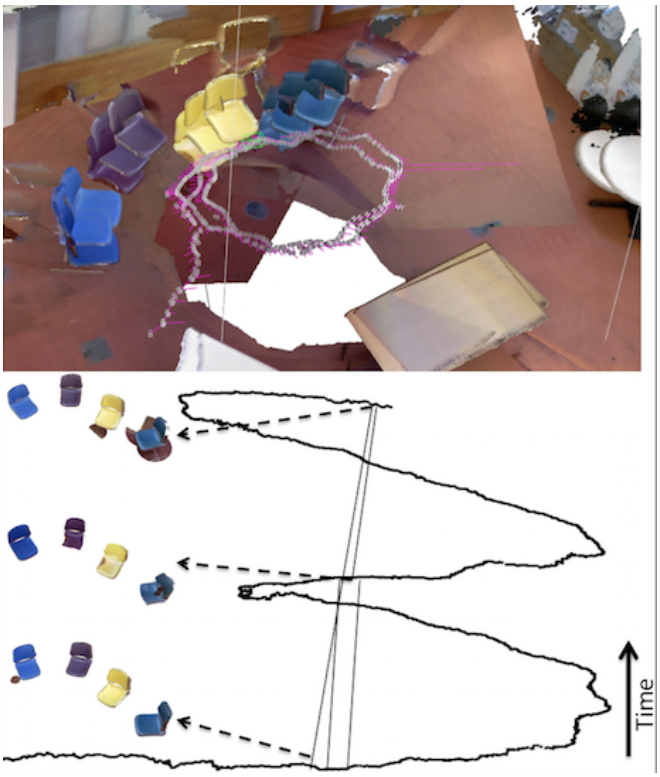

Fig. 7. **Top:** a map of a small scene with two loops. The trajectory of the camera is shown in purple. As shown, the map drifts with time, as can be seen with the offset objects. **Bottom:** the trajectory of the camera, with time being the vertical axis. On left, all places matched in the object graph are shown. These matches are shown in the trajectory with lines connecting the trajectory between times.

A. Lighting Invariance

One advantage of object-based place recognition over purely vision-based methods is that we recognize objects based on geometry, thus negating the lighting or shadowing issues. Variations in lighting are almost inevitable for any long-term deployment of a robot, so methods should be invariant to lighting conditions. Fig. 9 shows an example.

B. Multi-Session

Our place recognition framework extends naturally to multiple sessions of mapping or even sharing maps between cooperative robots. Since the object graph may have disconnected sub-graphs, all that is needed for multi-session mapping is to load in the previous object graph matrix, and the systems runs identically. This is shown in Fig. 8 where two separate maps are built, and the second map finds the same object place in the first graph. The two graphs can then be fused.

C. Object Discovery

The use of objects for place recognition raises an obvious concern: since movement is an inherent quality of objects

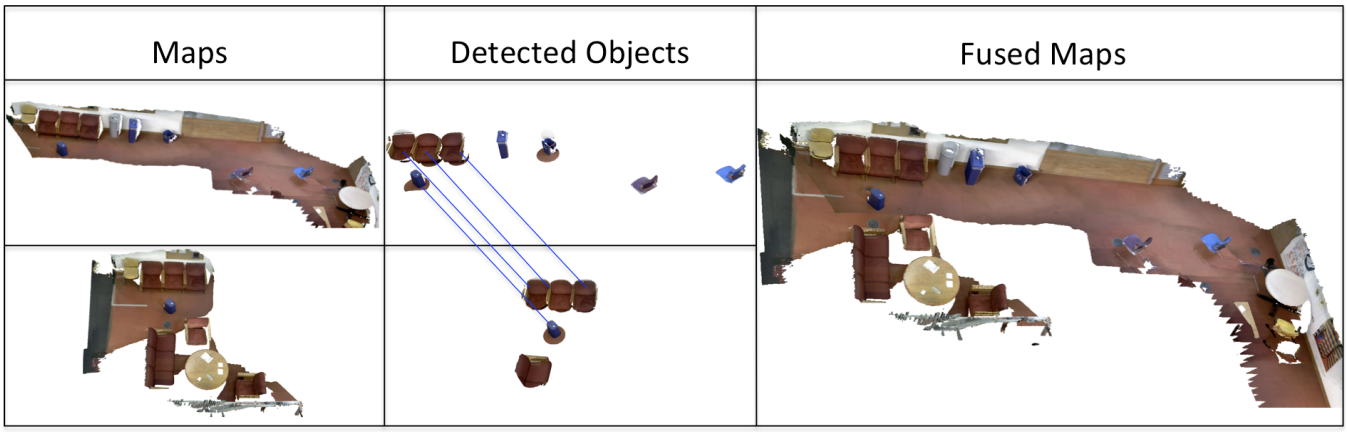

Fig. 8. **Left:** two maps collected at different times with different trajectories with some overlap between them. **Middle:** the detected objects using four object models (large and small chair, large and small recycling bin) to account for the variations within the category. For ease of viewing, the connectivity of the graph is not shown. The blue lines signify a correspondence between the two object graphs. So the detected place is the three red chairs and the small recycling bin. **Right:** the two maps transformed into the same coordinate system using the object locations.

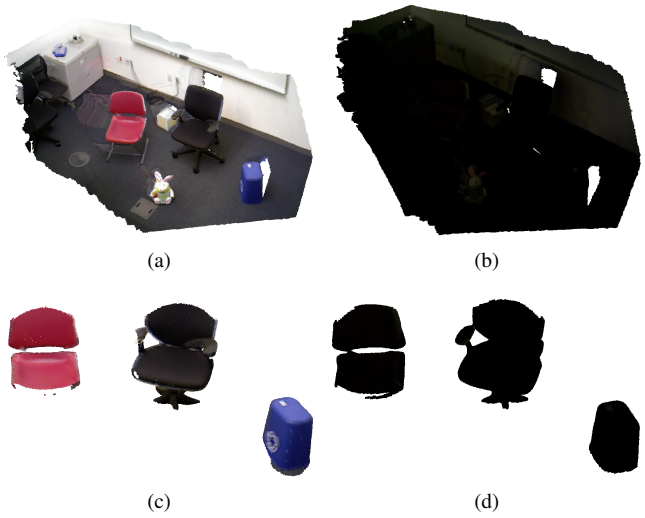

Fig. 9. Examples showing how our method is lighting invariant. (a) (b), two maps of the same scene with roughly similar trajectories. The second map (b) is purposefully difficult to see since the lights are turned off, thus showing the shortcomings of purely vision-based approaches. (a) (b) show the objects found within the two maps. Note that both chairs and the recycling bin can be detected.

over time [9], are objects a good representation for place recognition? The objects used in the evaluation of this work cannot be expected to remain static (chairs for example). We analyze this issue by observing that objects within an environment have a rate of change. Chairs move several times throughout a workday, but computer monitors generally remain constant for months. A scene should have some consistent structure between successive robot runs. Future work will evaluate this on larger and longer datasets.

Furthermore, changing environments can actually lead to improvements in place recognition since place recognition can be used for more than just aligning maps. Learning from changes in maps is an unsupervised way of performing object discovery [9], [19]. Object discovery has a limited window size that can be searched through since drift leads to changes in maps. However, with our method, there already is a place localized in a region where drift is negligible (by construction of our graph), so local change detection can occur even within a large map. By doing a simple 3-D difference in overlapping maps with minor filtering (see [9]), objects can be found. Fig. 10 shows an example of an object being found. With this recursive idea, a robot can start with a base set of objects (and as such, limited place recognition capability), and bootstrap its object dictionary. Using objects to find more objects.

VII. Conclusion

In this paper, we present a place-recognition system that operates over an object graph, and a method for incrementally building such a graph from a dense RGB-D map. Our method builds an object heatmap over a map by convolving and scoring multiple kernel patches, and detecting objects from the heatmap. The objects are then connected into a sparse object graph structure. We provide a definition for a *place* in the context of an object graph and use this definition to recognize places within the global object graph. Our flexible framework allows for efficient place comparison as well as naturally extending to multi-session mapping. We evaluate our method on multiple datasets.

We believe that object-based representations of the world are a prerequisite for many interesting avenues of future work. Specifically for our method presented, we will pursue learning optimal set kernel patches that discriminate the objects from the background rather than manually choosing them. We also will pursue how our method handles much larger object graphs (building-sized) in dynamic environments. Another promising direction is building on Section VI-C and having a robot bootstrap its object dictionary.

References

[1] E. Olson, "Real-time correlative scan matching," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, (Kobe, Japan), pp. 4387–4393, June 2009.

(a)

(b)

(c)

Fig. 10. (a) (b), Two maps with objects detected (two chairs and a file cabinet), similar to Fig. 8. Using the suggested alignment, changes in the map can be discovered. These changes are often objects, which can then be added into our dictionary of objects. (c), A discovered change from the two scenes.

[2] B. Williams, M. Cummins, J. Neira, P. Newman, I. Reid, and J. Tardós, "A comparison of loop closing techniques in monocular SLAM," *Robotics and Autonomous Systems*, vol. 57, no. 12, pp. 1188–1197, 2009.

[3] M. Cummins and R. Newman, "Appearance-only SLAM at large scale with FAB-MAP 2.0," *Intl. J. of Robotics Research*, 2010.

[4] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox, "RGB-D mapping: Using depth cameras for dense 3D modeling of indoor environments.," in *Intl. Sym. on Experimental Robotics (ISER)*, vol. 20, pp. 22–25, 2010.

[5] R. A. Newcombe, A. J. Davison, S. Izadi, P. Kohli, O. Hilliges, J. Shotton, D. Molyneaux, S. Hodges, D. Kim, and A. Fitzgibbon, "KinectFusion: Real-time dense surface mapping and tracking," in *IEEE and ACM Intl. Sym. on Mixed and Augmented Reality (ISMAR)*, (Basel, Switzerland), pp. 127–136, Oct. 2011.

[6] H. Johannsson, M. Kaess, M. Fallon, and J. Leonard, "Temporally scalable visual SLAM using a reduced pose graph," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, (Karlsruhe, Germany), May 2013.

[7] T. Whelan, M. Kaess, M. Fallon, H. Johannsson, J. Leonard, and J. McDonald, "Kintinuous: Spatially extended KinectFusion," in *RSS Workshop on RGB-D: Advanced Reasoning with Depth Cameras*, (Sydney, Australia), July 2012. Available as MIT CSAIL Technical Report MIT-CSAIL-TR-2012-020.

[8] R. Salas-Moreno, R. Newcombe, H. Strasdat, P. H. J. Kelly, and A. J. Davison, "SLAM++: Simultaneous localisation and mapping at the level of objects," in *Proc. IEEE Int. Conf. Computer Vision and Pattern Recognition*, (Portland, Oregon), June 2013.

[9] R. Finman, T. Whelan, M. Kaess, and J. Leonard, "Toward lifelong object segmentation from change detection in dense RGB-D maps," in *European Conference on Mobile Robotics*, (Barcelona, Spain), Sept. 2013.

[10] R. Finman, T. Whelan, L. Paull, and J. J. Leonard, "Physical words for place recognition in dense RGB-D maps," in *ICRA workshop on visual place recognition in changing environments*, June 2014.

[11] E. Herbst, X. Ren, and D. Fox, "RGB-D object discovery via multi-scene analysis," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pp. 4850–4856, IEEE, 2011.

[12] T. Whelan, M. Kaess, J. Leonard, and J. McDonald, "Deformation-based loop closure for large scale dense RGB-D SLAM," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, (Tokyo, Japan), Nov. 2013.

[13] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, pp. 91–110, Nov. 2004.

[14] J. Sivic and A. Zisserman, "Video Google: A text retrieval approach to object matching in videos," in *Intl. Conf. on Computer Vision (ICCV)*, vol. 2, (Los Alamitos, CA, USA), p. 1470, IEEE Computer Society, 2003.

[15] A. E. Johnson and M. Hebert, "Using spin images for efficient object recognition in cluttered 3D scenes," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 21, no. 5, pp. 433–449, 1999.

[16] R. B. Rusu, N. Blodow, and M. Beetz, "Fast point feature histograms (FPFH) for 3D registration," in *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pp. 3212–3217, IEEE, 2009.

[17] F. Tombari, S. Salti, and L. Di Stefano, "A combined texture-shape descriptor for enhanced 3d feature matching," in *Image Processing (ICIP), 2011 18th IEEE International Conference on*, pp. 809–812, IEEE, 2011.

[18] L. Bo, X. Ren, and D. Fox, "Depth kernel descriptors for object recognition," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pp. 821–826, IEEE, 2011.

[19] E. Herbst, P. Henry, and D. Fox, "Toward online 3-d object segmentation and mapping," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2014.

[20] S. Choudhary, A. J. Trevor, H. I. Christensen, and F. Dellaert, "Slam with object discovery, modeling and mapping," in *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pp. 1018–1025, IEEE, 2014.

[21] B. Drost, M. Ulrich, N. Navab, and S. Ilic, "Model globally, match locally: Efficient and robust 3D object recognition," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pp. 998–1005, IEEE, 2010.

[22] M. Cummins and P. Newman, "FAB-MAP: Probabilistic localization and mapping in the space of appearance," *Intl. J. of Robotics Research*, vol. 27, pp. 647–665, June 2008.

[23] M. J. Milford and G. F. Wyeth, "Seqslam: Visual route-based navigation for sunny summer days and stormy winter nights," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pp. 1643–1649, IEEE, 2012.

[24] M. Milford, "Vision-based place recognition: how low can you go?," *The International Journal of Robotics Research*, vol. 32, no. 7, pp. 766–789, 2013.

[25] Y. Latif, G. Huang, J. Leonard, and J. Neira, "An online sparsity-cognizant loop-closure algorithm for visual navigation," in *Robotics: Science and Systems (RSS)*, (Berkeley, CA), July 2014.

[26] R. Paul and P. Newman, "FAB-MAP 3D: Topological mapping with spatial and visual appearance," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pp. 2649–2656, IEEE, 2010.

[27] O. Duchenne, F. Bach, I.-S. Kweon, and J. Ponce, "A tensor-based algorithm for high-order graph matching," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 33, no. 12, pp. 2383–2395, 2011.

[28] M. Leordeanu and M. Hebert, "A spectral technique for correspondence problems using pairwise constraints," in *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, vol. 2, pp. 1482–1489, IEEE, 2005.