View-Invariant Loop Closure with Oriented Semantic Landmarks

Jimmy Li[1], Karim Koreitem[1], David Meger[1] and Gregory Dudek[1]

***Abstract*— Recent work on semantic simultaneous localization and mapping (SLAM) have shown the utility of natural objects as landmarks for improving localization accuracy and robustness. In this paper we present a monocular semantic SLAM system that uses object identity and inter-object geometry for view-invariant loop detection and drift correction. Our system's ability to recognize an area of the scene even under large changes in viewing direction allows it to surpass the mapping accuracy of ORB-SLAM, which uses only local appearance-based features that are not robust to large viewpoint changes. Experiments on real indoor scenes show that our method achieves mean drift reduction of 70% when compared directly to ORB-SLAM. Additionally, we propose a method for object orientation estimation, where we leverage the tracked pose of a moving camera under the SLAM setting to overcome ambiguities caused by object symmetry. This allows our SLAM system to produce geometrically detailed semantic maps with object orientation, translation, and scale.**

I. Introduction

In this paper we present an approach to mapping and localization that uses categorical objects as the key mapping primitive, allowing scene correspondence despite large changes in viewpoint, appearance or even scene configuration. We focus primarily on loop closure, a key aspect of simultaneous localization and mapping (SLAM) that leverages repeated observations of landmarks to correct for drift, which is the accumulation of errors in the map over time. To recognize previously-seen landmarks, traditional visual SLAM methods use dense low-level features for accurate alignment. These low-level features are not, however, invariant to large viewpoint changes ($> 30°$) [1], and cannot be used to reliably detect a loop when seeing the same area again from a very different viewpoint, as we will show in this paper.

To achieve greater viewpoint robustness, we use objects as high-level semantic landmarks, which can be detected with modern deepnet-based object detectors regardless of viewpoint [2]. Using only RGB images taken by a monocular camera, our SLAM system builds a semantically meaningful map containing the pose of objects, and uses their identity and geometry to detect and correct large loops. Fig. 1 shows an example where ORB-SLAM [3], a state-of-the-art SLAM method based on local appearance-based features, fails to loop close due to changes in viewpoint, whereas our method loop closes using a cell phone, a cup, and a bowl, thereby reducing drift by 80.4%.

*This work was supported by the Natural Sciences and Engineering Research Council (NSERC) of Canada, and the NSERC Canadian Robotics Network (NCRN)

[1]The authors are affiliated with the Mobile Robotics Lab, the Center for Intelligent Machines (CIM), and the School of Computer Science at McGill University in Montréal, Canada. jimmyli, karimkor, dmeger, dudek@cim.mcgill.ca

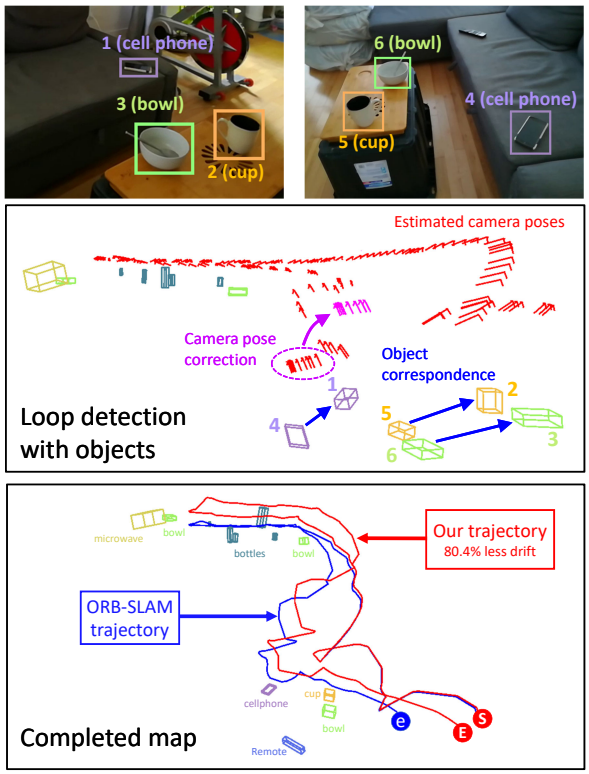


Fig. 1: View-invariant loop closure using objects. Top: the same objects observed from two disparate viewpoints along the camera trajectory. Middle: duplicate objects arising from drift are matched (blue arrow), informing camera pose correction (purple arrow). Bottom: camera starts and ends at the same position (S). ORB-SLAM estimates end position e, whereas our method estimates E, which has 80.4% less drift.

We build on our prior work [1], in which we present our semantic mapping algorithm, and demonstrate the utility of object landmarks for view-invariant relocalization. We have shown that given two separate maps of the same scene built from two completely different viewpoints (up to 125° apart), our system can use co-visible objects to reliably fuse the maps and align all camera poses in a single reference frame. This work expands on our prior work to leverage object landmarks for loop closure. Whereas the relocalization task assumes we are given two distinct maps containing possibly overlapping landmarks, in loop closure we continuously monitor for duplicate landmarks that arise due to drift, which requires a different inference engine.

Additionally, we introduce an approach for object orientation estimation that leverages multiple views captured by a moving camera. At the category level, many objects exhibit symmetry, which makes orientation estimation difficult. Our

approach exploits the estimated camera motion to overcome symmetric ambiguities. Compared to our prior work, which aligns object landmarks with the current scene layout, the ability to infer individual object orientation allows our system to map larger environments, and to better integrate with robotics tasks like manipulation that require fine-grained geometry.

II. Related Work

While traditional visual SLAM methods based on matching local appearance-based features and image intensities offer accurate mapping and localization [3]–[6], the map representation that results lacks semantic information, and localization is not robust to changes in illumination and viewpoint [1], [7]. This has inspired SLAM methods that use high-level objects as landmarks to address these issues. The notion of using high-level features for mapping can be traced back to Binford [8], and is used in mapping for mobile robots by Galindo [9]. Several object representations have been proposed in this context. Point [10] and quadric [11], [12] representations are convenient for optimization but are not well-suited to capture the physical extent of common cuboidal objects (i.e. monitors, keyboards). Inspired by the work of Bao *et al.* [13] and Meger *et al.* [14], we represent objects as bounding cuboids that tightly fit target objects. An effective modern realization of object-based SLAM is the work of Bowman *et al.* who capitalize on probabilistic optimization over visual and IMU data [10], [15]. Our work, in contrast, focuses on the recognition of scenes from drastically varying viewpoints using semantic cues. While RGB-D sensing could enable more detailed geometric reconstruction [16], [17], we opt to rely only on the ubiquitous RGB camera, which makes our system applicable to a wide range of hardware platforms.

Estimating object orientation from images is a well-studied problem [18]–[20], but has commonly been treated as a separate problem from SLAM. Inspired by Pillai *et al.* [21], who have used SLAM to improve object recognition, we leverage SLAM for multi-view orientation inference. This allows us to disambiguate object symmetry and simplifies the learning problem compared to prior work [18]. Unlike existing instance-level methods [22], we are interested in orientation estimation at the category level, which allows our system to operate in previously unseen environments.

III. SLAM System

A. Overview

Our SLAM system is based on the idea that local appearance-based features (i.e. ORB [23]) are valuable for accurately tracking the camera locally, but are not always reliable over long trajectories since they are not robust to large changes in viewing angle. Natural objects such as common household items, on the other hand, are reliably detectable across viewpoints, but in the absence of other cues, are difficult to use since computing the precise location of objects in the image plane is challenging. Modern object detectors typically output coarse bounding boxes that roughly enclose the object. Thus, we adopt a hybrid approach, where we use local appearance-based features to track the camera locally, and we leverage the known camera poses to simplify the inference of object poses in 3D space. Then, when the system is later required to match across large baselines for closing large loops or for relocalization, we use object landmarks for view-invariant matching.

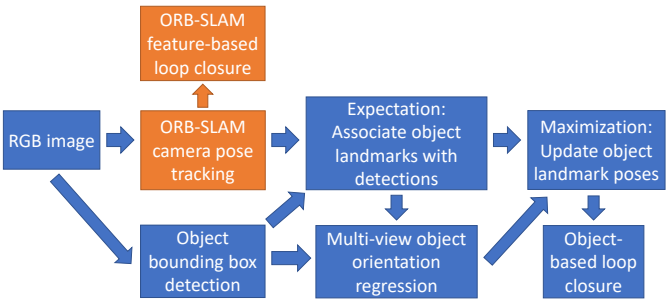


Fig. 2: Flowchart of our semantic SLAM pipeline.

Having shown the utility of this approach for the relocalization task in our prior work [1], in this paper we focus on applying this strategy to loop closure. Loop identification is achieved by matching the identity and geometric layout of duplicate objects that arise due to camera pose drift. By aligning duplicate landmarks, we also recover the pose correction for the camera frames that observe these landmarks. Graph-based non-linear optimization is then used to distribute this correction across all the camera poses in the trajectory. Fig. 2 contains a flowchart of the key components of our SLAM pipeline, which we will discuss next.

B. Semantic Mapping

We provide a brief summary of the semantic mapping framework presented in our prior work [1], which we use for inferring the pose of object landmarks. Our mapping system takes as input a sequence of RGB images, and outputs the estimated camera trajectory, along with object landmarks represented as bounding cuboids with 9 degrees of freedom – translation (x, y, z), rotation (roll, pitch, yaw), and scale (length, width, height). We build on the foundation of ORB-SLAM [3], which tracks the pose of a moving camera by matching ORB features [23] across image frames. The estimated camera poses allow us to then optimize bounding cuboid parameters to minimize reprojection errors with respect to bounding box detections produced by Faster-RCNN [2]. In this paper we improve our mapping system by introducing a new method that allows for instance-level object orientation regression. We discuss this in Section IV.

Since optimizing over all camera poses is expensive, ORB-SLAM prunes away redundant image frames that see similar feature points and operates over a sparse set of keyframes, which we leverage for efficient object inference. Upon keyframe insertion, we update the object landmarks using expectation maximization (EM): 1) landmarks are projected onto each keyframe and are matched with object detections; 2) the matched detections are combined with keyframe camera poses to triangulate and update landmark

poses. Detected bounding boxes that are not matched with any existing landmark are used to initialize new landmarks.

C. Loop Detection

Executing our semantic mapping pipeline over long trajectories typically leads to drift in the estimated camera poses, which results in duplicate object landmarks being added to our map. Since our method is based on ORB-SLAM, we inherit its feature-based loop closure mechanism, which allows our system to correct for drift if the same surfaces are viewed again from a similar viewpoint. This leaves much to be desired, however, since ORB-SLAM cannot cope with the case where the same area is viewed again from a drastically different viewpoint, which causes valuable loop closure opportunities to be missed. So, we add a second loop closure mechanism to our system that relies on the layout of mapped objects for loop detection.

After each iteration of EM in the mapping process, we identify the most recently mapped landmark as well as a set of close by landmarks, and try to match them to earlier landmarks. To measure the closeness of landmarks l and m, which are seen in keyframe sets K_l and K_m respectively, we define the *keyframe separation* between l and m as

$$\delta(l, m) = \min_{u \in K_l, w \in K_m} |u - w| \tag{1}$$

Here, $u - w$ denotes the subtraction between keyframe indices, which reflect the order in which keyframes are added to the map. l and m are considered *close* if $\delta(l, m) < \delta_K$, where δ_K is chosen to be a small positive integer (~ 5).

Let L be the set of landmarks containing the newest landmark l and the landmarks close to l. We aim to identify a subset of landmarks in L such that their spatial layout is similar to that of another set of landmarks seen earlier in the trajectory. If we manage to do so, then there is evidence that a loop is present. By only considering the set of recently mapped landmarks in L, we avoid having to consider all landmarks in our map.

It has been shown that given 3 corresponding non-colinear 3D points measured in two different reference frames, it is possible to recover the relative transformation between the two systems [24]. Thus, we attempt to match 3 objects in L with 3 objects taken from the rest of the map. Once the match is established, we can use the relative transformation to inform our loop correction. Since objects are sparse, we exhaustively iterate over all possible matches and consider each loop candidate in turn. Each candidate must pass the following checks before it is accepted as a valid match.

D. Geometric Loop Verification

Let (l_1, l_2, l_3) and (m_1, m_2, m_3) be the two sets of objects comprising the loop candidate we are currently considering, where l_u is matched with m_u for $u \in \{1, 2, 3\}$. We reject the candidate if it does not pass the following checks.

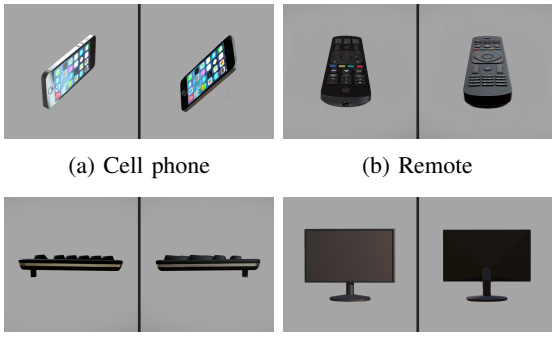
1) Matching proximity: Earlier we have defined keyframe separation in Equation 1, and mentioned that two objects are considered *close* if their keyframe separation is less than a threshold δ_K. We require that there must not be a match $l_u \leftrightarrow m_u$ such that l_u and m_u are close. This weeds out erroneous matches between objects that are next to each other, which could arise in cluttered scenes.

2) Object pose confidence: To prevent false positive matches, we need to ensure that the matched objects are well-localized, and are not instantiated based on spurious object detections. This is crucial for the subsequent checks that utilize object geometry. We require that each matched object must have been observed by at least two keyframes with viewing angles spanning at least θ degrees. Setting $\theta = 15^\circ$ typically works well.

3) Object layout: Comparing the geometric layout of the matched objects requires us to express all objects in a common reference frame. Representing each object as a point indicated by its translation, we first establish local coordinate systems A and B based on (l_1, l_2, l_3) and (m_1, m_2, m_3) respectively, and then compute a similarity transformation consisting of a rotation R, translation t, and scaling s such that any point p_l in A can be mapped to the corresponding position p_m in B with $p_m = sR \cdot p_l + t$. This can be accomplished using the method of Horn [24]. We map l_1, l_2, and l_3 from A to B, while also scaling their width, length, and height using s to account for any scale drift. We then proceed to the following checks using object poses in B.

- **Scale consistency**: False object correspondences often lead to an unreasonable scale factor s. Thus, we ensure that once l_1, l_2, and l_3 have been scaled by s, their length, width, and height are consistent with that of m_1, m_2, and m_3 respectively. Let $wid(l)$, $len(l)$, and $hei(l)$ denote the width, length, and height of l, and let $g(l) = max\big(wid(l), len(l), hei(l)\big)$. For each matching pair $l_u \leftrightarrow m_u$. We require that $|g(l_u) - g(m_u)|/min\left(g(l_u), g(m_u)\right) < \tau_s$.
- **Translational consistency**: Let $t(l)$ denote the translation of object l. The distance between matched objects l_u and m_u must not exceed a threshold τ_t. Since monocular tracking does not provide absolute scale, we use a scale-normalized distance and require $||t(l_u) - t(m_u)||_2/min\big(wid(l_u), len(l_u), hei(l_u)\big) < \tau_t$.
- **Rotational consistency**: A failure case we have observed in cluttered scenes is where translational consistency is achieved, but the alignment requires objects to be flipped upside down. Thus, we require the difference in orientation between l_u and m_u must not exceed a threshold τ_r, where the difference is measured according to the metric we will present in Section IV-F.

4) Supporting Inliers: Having 3 well aligned object correspondences is not always sufficient for recognizing a loop, especially when dealing with repetitive objects such as bottles that tend to form dense clusters. To identify additional matching pairs, we make two copies of all object landmarks, with one copy in coordinate system A and the second in B. As before, we map both sets of landmarks into a common coordinate system using the similarity transformation, and then accumulate pairs of landmarks that pass all of the aforementioned checks. We refer to these pairs as inliers.



(a) Cell phone (b) Remote

(c) Keyboard left, right (d) Monitor front, back

Fig. 3: Many objects have similar appearance when viewed from different orientations, causing our orientation regressor to diverge during training.

While we could directly threshold the number of inliers, this does not take into account the frequency of each object category. Instead, we compute a weighted count of inliers, which we formulate as

$$C = \sum_{i \in inliers} w_{lab(i)} \tag{2}$$

where $lab(i)$ denotes the category label of the objects in the inlier pair, and w_α is the weight of category α. A lower weight is used for higher frequency categories. The weighted inlier count C of an acceptable loop candidate must surpass a threshold τ_i. The tuning of these parameters depends on the types of objects used and their frequencies in the environment.

5) Inlier Separation: In situations where multiple sets of objects in the environment have similar geometric layout, the detected loop may become ambiguous. Suppose we are presented with multiple loop candidates sorted in descending order by their weighted inlier count. If the inlier count of the first candidate exceeds that of the second candidate by τ_g, then we accept the first candidate and use it to perform a loop correction. Otherwise, we interrupt the loop closure operation and wait for a less ambiguous candidate.

E. Loop Correction

ORB-SLAM's loop closure mechanism relies on distributing the loop error along the *essential graph*, which is a spanning tree of the co-visibility graph, in which keyframe vertices are connected based on co-visibility of ORB features. This approach is more efficient than operating over all images. Given an accepted loop candidate, we apply the same similarity transformation used for aligning the object matches to the keyframe camera poses that observe the landmarks l_1, l_2, and l_3. We then run ORB-SLAM's essential graph optimizer to distribute this correction throughout all the keyframes via non-linear least squares optimization.

Having corrected the keyframe poses, we proceed to update our semantic map. For each object landmark, we take one of the keyframes that observe it, and compute the relative pose of the keyframe before and after the update. We apply the same relative transformation to update the landmark. We then run an EM step to fine tune all the landmark poses. Note that the object pair in each loop candidate inlier are now duplicates, and so we drop one landmark from each pair. Additionally, there may be objects that did not contribute to loop closure due to incorrect pose, but whose poses are improved as mapping continues after loop correction. Thus, we perform non-maximum suppression to remove redundant objects as they arise, by applying the identity, scale, translational and rotational consistency checks.

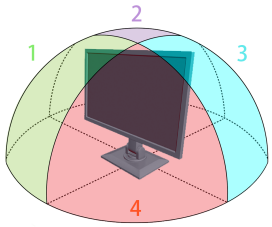


Fig. 4: Separate orientation regressor trained for each partition of the top viewing sphere to eliminate ambiguity.

IV. Orientation Regression

A. Overview

This section describes our approach for estimating the orientation of object landmarks, which we use in our SLAM system. A key challenge that arises when training an orientation regressor for typical household objects is symmetry. Fig. 3 shows examples where objects exhibit similar appearances when observed from different viewpoints. This usually causes the learner to diverge during training, and motivates our proposed multi-view inference. We focus primarily on addressing reflectional symmetry shown in Fig. 3 and briefly discuss the simpler case of cylindrical symmetry.

B. Multi-view Orientation Inference

To cope with reflectional symmetry in cuboidal objects, we start by dividing the viewing sphere into partitions, such that no two viewing directions in each partition captures a similar image. Fig. 4 shows one possible partitioning into four quadrants. The bottom half of the sphere is omitted since we assume the camera will be held at roughly eye-level. We train a separate orientation regressor for each partition. Details regarding training are provided in Section IV-D.

Suppose there are n orientation regressors that correspond to n partitions. Each time the object is seen by the moving camera, we use all n regressors to estimate an orientation, giving us $Q_i^1, ...Q_i^n$ for each camera pose i. Assume that camera pose estimates are given by our SLAM system, and let i and j be any two camera poses that observe the object. We can identify the correct object orientations Q_i^* and Q_j^* seen from i and j respectively with

$$Q_i^*, Q_j^* = \underset{\substack{Q_i^* \in \{Q_i^1 ... Q_i^n\}, \\ Q_j^* \in \{Q_j^1 ... Q_j^n\}}}{\operatorname{argmin}} \Delta(T_{j \to i}(Q_j^*), Q_i^*) \tag{3}$$

where $T_{j \to i}$ transforms Q_j^* to be in camera i's reference frame and Δ computes the relative angle between two orientations. The resulting Q_i^*, Q_j^* are consistent with the estimated camera motion. Typically, each object is seen from

Fig. 5: Examples of synthetically rendered training data for our orientation regressor.

more than two camera poses, in which case we use the pair of camera poses that differ the most in their global viewing direction. Our experiments show that this approach works well in practice, since using the regressor trained on one partition to process an image taken in a different partition tends to lead to inconsistencies with the camera motion.

When integrated with our SLAM pipeline, orientation regressors operate on image patches enclosed by object bounding box detections. Thus, estimated orientations $Q_i^1, ...Q_i^n$ are only accurate with respect to the local patch. We follow the approach of Mousavian *et al.* and use the angle of the ray through the bounding box center to correct the orientations so they are in the camera's frame of reference [25].

C. Reducing Regressors

While the formulation we have presented is valid for the general case, we can reduce the number of orientation regressors if we only aim to compute a tightly fitting bounding cuboid. For the example in Fig. 4, we could consolidate partitions 2 and 4, as well as partitions 1 and 3. We could then train our orientation regressor to ignore any features that distinguish the front and back of the monitor. Furthermore, because the monitor exhibits reflectional symmetry, any image taken in partition 2/4 is similar to a flipped image taken in partition 1/3. Thus, we could use a single regressor to output two orientations based on the original image and a flipped image. We use these simplifications when dealing with cuboidal objects, including monitor, keyboard, cell phone, remote, and microwave, which allows to estimate orientation up to a 180° yaw.

D. Training

The convolutional network used for orientation regression is described in our prior work [26], where it is used to estimate the orientation of a mobile robot. The network outputs a unit quaternion $q = (w, x, y, z)$ given an image. Following the methodology of domain randomization [27], we generate synthetic training data by rendering CAD models of objects taken from ShapeNet [28] using a variety of lighting conditions. We use a random patch taken from the Pascal dataset [29] as the background for each rendering. Examples of training data are shown in Fig. 5.

E. Cylindrical Objects

For objects with cylindrical symmetry, we simply train our orientation regressor to output a 3-dimensional unit vector indicating the up direction of the object. Since our convolutional network architecture is not specific to quaternions, it can be readily adapted to output a 3-dimensional vector. When determining the orientation of bounding cuboids for a cylindrical object we need only to ensure that its up direction is properly aligned. The bounding cuboid thus loses one degree of rotational freedom.

F. Comparing Orientations

Our SLAM pipeline compares the orientation between objects of the same category as part of its loop detection mechanism. For cuboidal objects, we determine the orientation up to a 180° yaw, so given the rotations Q_i and Q_j of two cuboidal objects, we output $min\big(\Delta(Q_i, Q_j), \Delta(Q_i, R_\pi Q_j)\big)$, where R_π applies a 180° yaw to Q_j. For two cylindrical objects, we output the angle between their up vectors.

V. Experiments

A. Loop Closure

Existing visual SLAM datasets for evaluating loop closure typically allow the camera to re-image the same surfaces without large changes in viewpoint. The TUM dataset [30], for instance, has this property, and so our method does not significantly enhance mapping accuracy. To better gauge the benefit of our system, we collect our own RGB video sequences, where the camera is allowed to re-image the same areas of the scene, but not from similar viewpoints.

We present our result on all 7 sequences we collect – one is shown in Fig. 1 and the rest in Fig. 6. All data are collected indoors, except for the bottom right sequence in Fig. 6, in which the camera starts in a kitchen, goes outside the building, circles around the block, re-enters through the front door, and revisits the kitchen. Our system uses 5 cuboidal objects (monitor, keyboard, cell phone, remote, microwave) and 4 cylindrical objects (cup, bottle, bowl, potted plant) for semantic mapping and loop closure.

We compare our semantic SLAM system with ORB-SLAM. Each trajectory is collected such that the camera starts and ends at the same position, which allows us to gauge drift via the discrepancy in the estimated start and end camera poses. A short video segment with translational camera motion precedes each sequence to allow ORB-SLAM to initialize. We save the state of the system just before performing an object-based loop correction, so that we can later resume the run with object-based loop correction disabled to measure the performance of ORB-SLAM. This approach ensures both methods initialize with the same scale. We report our method's percentage reduction in drift relative to ORB-SLAM's drift, defined as

$$\frac{||E - S||_2 - ||e - S||_2}{||e - S||_2} \cdot 100 \tag{4}$$

where S is the common starting camera translation, E is our method's estimated end translation, and e is ORB-SLAM's estimated end translation. Since ORB-SLAM only maintains the pose of a sparse network of keyframes, for both methods we enable ORB-SLAM's relocalization mode after mapping is completed and input the start and end images to obtain their corresponding camera poses.

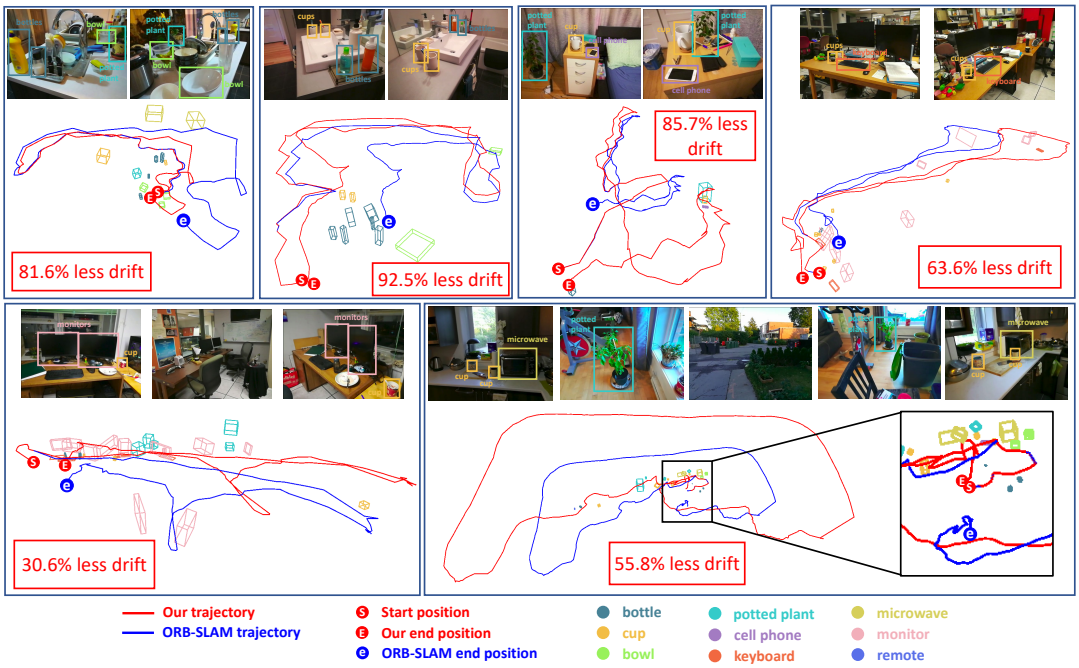
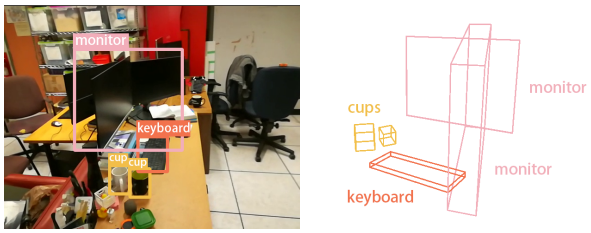


Fig. 6: We show six maps built by our semantic SLAM system, which uses object landmarks for view-invariant loop closure. Images taken during the trajectory are shown with the objects involved in the loop correction labeled with bounding boxes. Our estimated trajectory (red) is compared to that of ORB-SLAM (blue). The camera starts and ends at the same position (S). ORB-SLAM estimates end position e, whereas our method estimates E. We achieve lower drift in all cases.



(a) Bad monitor detection (b) Erroneous monitor pose

Fig. 7: Spurious detections lead to poor object pose estimates.

Fig. 1 and 6 show that our method successfully loop closes using objects and achieves a drift reduction compared to ORB-SLAM in all cases, ranging from 30.6% to as high as 92.5%. Table I shows the drift reduction for all seven sequences, along with the mean (70.0%). ORB-SLAM is unable to detect loops, causing drift to remain uncorrected.

B. False Detections

Our method is sensitive to the quality of object detection. Fig. 7 shows an example where incorrect detections of the monitor adversely affects the estimated object pose. These false detections are intermittent, and in this case affects both monitors shown. While our method manages to correctly loop close for the trajectory from which this example is taken, it does so using other better-localized objects in the scene and is unable to leverage the monitors.

C. Orientation Estimation

We evaluate object orientation estimates produced by our SLAM pipeline against human-labeled ground truth. For each object in our trajectories, we record its orientation relative to the first keyframe that detects it, and ask a human annotator to label its orientation relative to the same keyframe. We use the tool of Xiang *et al.* [31], which allows the annotator to align a CAD model to the object in the image. The mean orientation error for cuboidal and cylindrical objects are presented separately in Table II. Errors are computed according to the metric described in Section IV-F.

TABLE I: Drift reduction vs ORB-SLAM

Sequence	Drift reduction (%)
living room	80.4
kitchen 1	81.6
bathroom	92.5
bedroom	85.7
computer lab 1	63.6
computer lab 2	30.6
kitchen 2	55.8
Mean	70.0

TABLE II: Orientation estimation error

	Mean (degrees)	Stdev (degrees)	Count
Cuboidal objects	28.57	21.22	31
Cylindrical objects	15.78	8.58	65

VI. Conclusion

We have presented a SLAM system that builds semantically meaningful maps containing object landmarks, which it exploits to robustly close loops under large viewpoint variations. For future work we intend to experiment on longer trajectories that span both indoor and outdoor environments, and use a wider range of object categories. We also wish to leverage more detailed observation models for objects, such as object keypoint detection.

REFERENCES

[1] J. Li, D. Meger, and G. Dudek, "Semantic Mapping for View-Invariant Relocalization," in *International Conference on Robotics and Automation*, 2019.

[2] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems*, 2015.

[3] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós, "ORB-SLAM: a versatile and accurate monocular SLAM system," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[4] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "Monoslam: Real-time single camera slam," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, no. 6, pp. 1052–1067, 2007.

[5] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *International Symposium on Mixed and Augmented Reality*, 2007.

[6] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-scale direct monocular SLAM," in *European Conference on Computer Vision*, 2014.

[7] C. Valgren and A. J. Lilienthal, "Sift, surf and seasons: Long-term outdoor localization using local features," in *European Conference on Mobile Robots*, 2007.

[8] D. J. Kriegman and T. O. Binford, "Generic models for robot navigation," in *International Conference on Robotics and Automation*, 1988.

[9] C. Galindo, A. Saffiotti, S. Coradeschi, P. Buschka, J. A. Fernandez-Madrigal, and J. Gonzalez, "Multi-hierarchical semantic maps for mobile robotics," in *International Conference on Intelligent Robots and Systems*, 2005.

[10] S. Bowman, N. Atanasov, K. Daniilidis, and G. J. Pappas, "Probabilistic data association for semantic slam," in *International Conference on Robotics and Automation*, 2017.

[11] L. Nicholson, M. Milford, and N. Sünderhauf, "Quadricslam: Dual quadrics from object detections as landmarks in object-oriented slam," *Robotics and Automation Letters*, vol. 4, no. 1, pp. 1–8, 2018.

[12] M. Hosseinzadeh, K. Li, Y. Latif, and I. Reid, "Real-time monocular object-model aware sparse slam," in *International Conference on Robotics and Automation*, 2019.

[13] S. Y. Bao, M. Bagra, Y.-W. Chao, and S. Savarese, "Semantic structure from motion with points, regions, and objects," in *Conference on Computer Vision and Pattern Recognition*, 2012.

[14] D. Meger, C. Wojek, J. J. Little, and B. Schiele, "Explicit occlusion reasoning for 3d object detection." in *British Machine Vision Conference*, 2011.

[15] N. Atanasov, S. L. Bowman, K. Daniilidis, and G. J. Pappas, "A unifying view of geometry, semantics, and data association in slam," in *Proceedings of the International Joint Conference on Artificial Intelligence*, 2018.

[16] B. Mu, J. L. Shih-Yuan Liu, Liam Paull, and J. P. How, "Slam with objects using a nonparametric pose graph," in *International Conference on Intelligent Robots and Systems*, 2016.

[17] R. F. Salas-Moreno, R. A. Newcombe, H. Strasdat, P. H. J. Kelly, and A. J. Davison, "Slam++: Simultaneous localisation and mapping at the level of objects," in *Conference on Computer Vision and Pattern Recognition*, 2013.

[18] A. Saxena, J. Driemeyer, and A. Y. Ng, "Learning 3-d object orientation from images," in *International Conference on Robotics and Automation*, 2009.

[19] S. Suwajanakorn, N. Snavely, J. J. Tompson, and M. Norouzi, "Discovery of latent 3d keypoints via end-to-end geometric reasoning," in *Advances in Neural Information Processing Systems*, 2018.

[20] J. Ku, A. D. Pon, and S. L. Waslander, "Monocular 3d object detection leveraging accurate proposals and shape reconstruction," in *Conference on Computer Vision and Pattern Recognition*, 2019.

[21] S. Pillai and J. Leonard, "Monocular slam supported object recognition," in *Robotics: Science and Systems*, 2015.

[22] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, "Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes," in *Robotics: Science and Systems*, 2018.

[23] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," in *International Conference on Computer Vision*, 2011.

[24] B. K. Horn, "Closed-form solution of absolute orientation using unit quaternions," *Josa a*, vol. 4, no. 4, pp. 629–642, 1987.

[25] A. Mousavian, D. Anguelov, J. Flynn, and J. Kosecka, "3d bounding box estimation using deep learning and geometry," in *Conference on Computer Vision and Pattern Recognition*, 2017.

[26] K. Koreitem, J. Li, I. Karp, T. Manderson, F. Shkurti, and G. Dudek, "Synthetically trained 3d visual tracker of underwater vehicles," in *OCEANS*, 2018.

[27] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *International Conference on Intelligent Robots and Systems*, 2017.

[28] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, *et al.*, "Shapenet: An information-rich 3d model repository," *arXiv preprint arXiv:1512.03012*, 2015.

[29] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, June 2010.

[30] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of rgb-d slam systems," in *International Conference on Intelligent Robots and Systems*, 2012.

[31] Y. Xiang, W. Kim, W. Chen, J. Ji, C. Choy, H. Su, R. Mottaghi, L. Guibas, and S. Savarese, "Objectnet3d: A large scale database for 3d object recognition," in *European Conference on Computer Vision*, 2016.