ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/274729790

Real-time monocular object SLAM

Article *in* Robotics and Autonomous Systems · April 2015

DOI: 10.1016/j.robot.2015.08.009 · Source: arXiv

CITATIONS
7

READS
106

4 authors, including:

Marta Salas
University of Zaragoza

Juan Domingo Tardos
University of Zaragoza

J. M. M. Montiel
University of Zaragoza

Some of the authors of this publication are also working on these related projects:

Project 3D-Surg View project

Project Ready-to-Transfer Visual SLAM View project

All content following this page was uploaded by J. M. M. Montiel on 16 June 2015.

The user has requested enhancement of the downloaded file. All in-text references underlined in blue are added to the original document and are linked to publications on ResearchGate, letting you access and read them immediately.

arXiv:1504.02398v1 [cs.RO] 9 Apr 2015

Real-time Monocular Object SLAM

Dorian Gálvez-López[a,*], Marta Salas[a,*], Juan D. Tardós[a], J. M. M. Montiel[a]

[a] *Instituto de Investigación en Ingeniería de Aragón (I3A), Universidad de Zaragoza, María de Luna 3, Zaragoza 50018, Spain.*

Abstract

We present a real-time object-based SLAM system that leverages the largest object database to date. Our approach comprises two main components: 1) a monocular SLAM algorithm that exploits object rigidity constraints to improve the map and find its real scale, and 2) a novel object recognition algorithm based on bags of binary words, which provides live detections with a database of 500 3D objects. The two components work together and benefit each other: the SLAM algorithm accumulates information from the observations of the objects, anchors object features to especial map landmarks and sets constrains on the optimization. At the same time, objects partially or fully located within the map are used as a prior to guide the recognition algorithm, achieving higher recall. We evaluate our proposal on five real environments showing improvements on the accuracy of the map and efficiency with respect to other state-of-the-art techniques.

Keywords: object slam, object recognition

---

1. Introduction

A robot that moves and operates in an environment needs to acquire live information about it in real time. This information can be obtained from Visual SLAM (*simultaneous localization and mapping*), a key component of many systems that allows mobile robots to create maps of their surroundings as they explore them, and to keep track of the location of themselves. Computed maps provide rich geometrical information useful for reliable camera location, but it is poor for describing the observed scene. Recently, these maps have been augmented with objects to allow the robots to interact with the scene [1, 2, 3].

To include objects in SLAM maps, these must be recognized in the images acquired by the robot by computing a rigid-body 3D transformation. A vast research line has provided solutions to this problem [4, 5, 6, 7], but it has been aside from visual SLAM.

Our aim in this paper is to approach object recognition and monocular object SLAM together, with a novel solution based on accumulating information over time to compute more robust poses of objects and to keep them constantly located in the scene. To achieve this, we propose a novel object recognition algorithm that provides detections of objects as a keyframe-based visual SLAM algorithm builds a map.

Once an object is observed several times from different camera positions, those object features with several observations are triangulated within the map as anchor points. Anchor points provide the location of the object within the map and set additional geometrical constraints in the bundle adjustment (BA) optimization. Since object models are at real scale, anchor points provide observations of the map scale.

Standard BA optimizes camera poses and map point locations and it is well known that it can only recover maps up to scale. In contrast, our algorithm optimizes the camera poses, the points as well as the anchor points, the objects and the scale, and as a result we have maps at real scale composed by objects.

Our system relies on an object recognition algorithm that works on a single-image basis but takes advantage of the video sequence. It exploits the information collected by SLAM to treat previous observations as cues for the location of the objects in the current image. This allows to obtain faster and more repeatable detections that, in turn, provide more geometrical constraints to SLAM.

The novel object recognition algorithm we propose, based on bags of binary words [8], uses a static visual vocabulary that is independent of the number of objects, and models the entire appearance of the objects with ORB (*oriented FAST and rotated BRIEF*) features [9]. Poses of objects are found from 2D-to-3D correspondences that are refined by guided matching during a RANSAC-like step [10]. Our system performs a fast and reliable recognition of 3D objects with databases comprising up to 500 objects, while keeping the real-time constraints of a SLAM system.

Our work makes the following contributions:

1. We present a complete visual SLAM system that is able to insert real objects in the map and to refine their 3D pose over time by re-observation, with a single monocular camera.
2. We show the feasibility of storing hundreds of comprehensive 3D models in a single object database, composed of bags of binary words with direct and inverted indices. We also propose a novel technique to sample putative correspondences in the verification stage.

---

[*] These authors have contributed to the present work in equal parts.

Email addresses: `dorian3d@gmail.com` (Dorian Gálvez-López), `msalasg@unizar.es` (Marta Salas), `tardos@unizar.es` (Juan D. Tardós), `josemari@unizar.es` (J. M. M. Montiel)

3. We propose a new SLAM back-end that includes the geometrical information provided by the objects into the map optimization to improve the accuracy of the map, the objects and their relative scale at each step.
4. We present results in real and independent datasets and comparisons with other systems. Our results proof that, by including objects, our monocular system can retrieve the real scale of the scene, and obtains more accurate results that PTAM [11] and RGB-D SLAM [12], while keeping realtime performance (tracking takes 7.6 ms, and recognition, around 200 ms per image). Our results also demonstrate that the system is extremely robust against occasional wrong detections, avoiding map corruption.

The paper is distributed as follows: Section 2 presents the related work of object SLAM and object recognition. Section 3 gives an overview of our complete system. Section 4 details the visual SLAM approach and the object insertion, and Section 5, the object recognition algorithm. Section 6 shows the experimental evaluation of our system, and Section 7 concludes the paper.

2. Related work

Object-augmented mapping has been previously approached by SLAM methods based on the extended Kalman filter [2, 13]. However, nowadays state-of-the-art monocular SLAM methods are based on keyframes, which create maps just with some selected video frames. As Strasdat *et al.* [14] proved, these systems are able to produce better results than filter based methods because they handle a great deal of points and produce larger accurate sparse point maps in real time at frame rate.

The work by Castle *et al.* [3] was one of the first ones that merged object recognition and monocular keyframe-based SLAM. After detecting an object in two frames, they compute its pose in the map. These objects are shown as augmented reality but, unlike our approach, they do not add the objects to the optimization. They built a database of 37 planar pictures described by SIFT features. Contrary to this system, restricted to planar objects, we can deal with objects with arbitrary 3D shapes.

Bao *et al.* [15, 16] were the first to present Semantic Structure from Montion (SSfM), which is a framework to jointly optimize cameras, points and objects. SLAM methods deal with the fact that the information proceeds from a video stream, thus the graph of points and keyframes is incremental, while SSfM processes all the frames at once. Moreover, recognition and reconstruction steps are separated and independent in [15]. However, on our algorithm, recognition and reconstruction take place at the same time since SLAM and object detection are fully integrated. The recognition method in [15] retrieves a bounding box of the object while our object detector retrieves a 6DoF pose.

Along the same line, Fioraio *et al.* [17] presented a SLAM system that adds 3D objects in the maps when they are recognized with enough confidence, optimizing their pose together with the map by bundle adjustment. They build a database of 7 objects that are described by 3D features that are acquired at several scales with a RGB-D camera, creating an independent index for each scale. The recognition is performed by finding 3D-to-3D putative correspondences that are filtered by a RANSAC-based algorithm. Although they are able to build room-sized maps with a few objects, their system does not run in real time. In comparison, our system improves scalability and execution time by using binary 2D features and a single index structure that can deal with all the keypoint scales at the same time.

Salas-Moreno *et al.* [1] presented one of the most recent visual SLAM systems that combines RGB-D map building and object recognition. They represent the map with a graph in which nodes store position of cameras or objects, and enhance the pose of all of them when the overall graph is optimized. A database of objects is built beforehand with KinectFusion [18], describing their geometry with point pair features [19]. These are indexed by a hash table, and the recognition is performed by computing a large number of candidate rigid-body transformations that emit votes in a Hough space. Hough voting is a popular technique for object detection with RGB-D data [20, 21, 22], but its scalability to hundreds of objects is not clear. In fact, Salas-Moreno *et al.* [1] achieve real-time execution by exploiting GPU computation, but they show results with just 4 objects. In our work, we show results at high frequency with up to 500 objects, computed on a CPU from a monocular camera.

Regarding object recognition, our proposal follows the line of research consisting in finding matches of local features between an image and an object model. Sivic & Zisserman [23] presented a visual vocabulary to match 2D images in large collections. They proposed to cluster the descriptor space of image features with k-means to quantize features and represent images with numerical vectors, denoted as bags of words, enabling quick comparisons. On the other hand, Lowe [24] popularized an approach based on directly matching SIFT features between query and model 2D images. Matching features requires to compute the descriptor distance between large sets of features, which can be very time-consuming. To speed up this process, he proposed the *best-bin-first* technique to find approximate neighbors with a k-d tree. Both visual vocabularies and k-d trees were later generalized for matching large sets of images in real time. Nister & Stewenius [25] presented a hierarchical visual vocabulary tree built on MSER (*maximally stable extremal regions*) [26] and SIFT features with which yielded fast detections with a dataset of 40000 images. Muja & Lowe [27] presented a method to automatically configure a set of k-d trees to best fit the feature data to match.

To fully recover the pose of an object from a single image, it is necessary to incorporate 3D information to the models. Gordon & Lowe [28] started to create 3D point cloud models, recovering the object structure by applying structure from motion techniques. The pose could be then retrieved by solving the perspective-n-problem [10] from 2D-to-3D correspondences. This has been the basis of a lot of recent object recognition approaches [5, 29, 7, 6, 30]. For example, Collet *et al.* [5] build 3D models for 79 objects and use the training images of the objects to build a set of k-d trees to index their SIFT features and

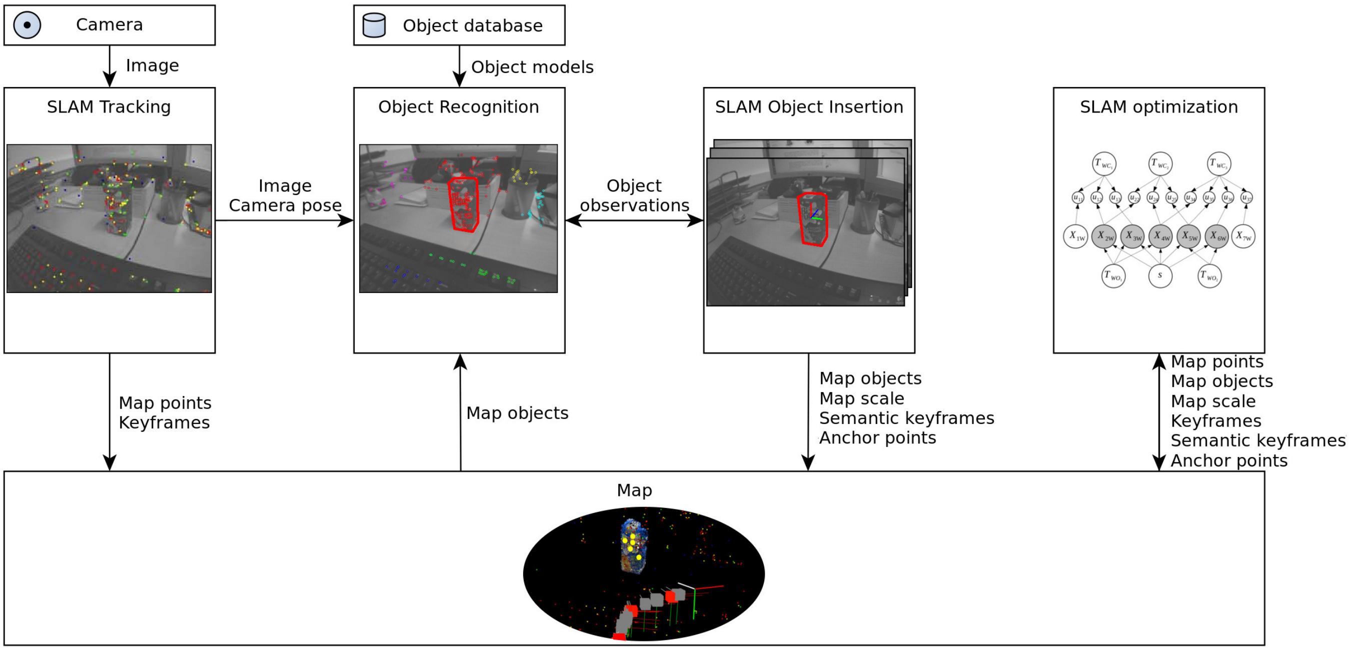


Figure 1: System overview: Every video frame is processed by the SLAM tracking thread to locate the camera, and to determine if a new keyframe is added to the map. Object recognition is applied to as many frames as possible, exploiting the information of the location of objects previously seen. If the recognition is successful, the observation of the object is stored until there is enough geometrical information about it. In that moment, the object instance is triangulated and inserted in the map, together with new map points anchored to object points and a subset of frames that observed them, coined semantic keyframes. This operation allows to find the map scale and to include object geometrical constraints to the map optimization.

do direct matching. To enhance the detection of small objects and avoid the background, they run the recognition on small sets of SIFT features that are close in the query image, merging the detections later if the object poses overlap. Although we also divide the query features into regions, we merge the 2D-to-3D correspondences before computing any pose. This prevents from missing detections in the cases of oversegmented regions with few correspondences. In a similar way, Pangercic *et al.* [7] create a database of 50 3D objects represented with a SIFT vocabulary tree, trained with the same object images. They rely on a RGB-D camera to segment out the background.

The diverse discretization levels of trees allow to compute feature correspondences in several manners. For example, Hsiao *et al.* [30] discretize the SIFT descriptor space in a hierarchical manner to create a 3-level tree. They show the benefits of computing feature matches at all the levels and not only the finest one, obtaining more putative correspondences that increase the object recognition rate. However, an excess of correspondences may overburden the pose recovery stage, leading to a large execution time. In contrast, Sattler's *et al.* [29] approach retrieves correspondences only from those features that lie in the same visual word, but this may miss correct pairs of points that do not share the visual word due to discretization error. In our work, we use a direct index [8] to compute correspondences between features that lie in the same tree node at a coarse discretization level. This provides a balanced trade-off between amount of corresponding points and execution time.

All these works use SIFT or SURF features, which are described with vectors of 64 or 128 float values, and train matching trees with the same images with which the objects are modeled, which forces them to recreate the trees when new objects are added to the database. Rublee *et al.* [9] presented ORB features, which are binary and compact (256 bit length descriptor), and provide a distinctiveness similar to that of SIFT and SURF [31]. Furthermore, visual vocabularies of binary words created from independent data and that do not need reconstruction are suitable to index large collections of images [8]. We show in this work the viability of a single independent vocabulary of ORB features to recognize 3D objects with large databases (up to 500 objects) in real time (around 200 ms/image).

3. System overview

Our system builds a 3D map composed by camera poses, points *and objects*, as illustrated by Figure 1. We make use of the front-end of the *Parallel Tracking and Mapping* (PTAM) algorithm [11] to track the camera motion, and add two new parallel processes to perform object recognition and object insertion in the map. Our system also includes a completely redesigned back-end based on g2o [32] that performs a *joint* SLAM optimization of keyframe poses, map points, objects and map scale.

The *SLAM tracking* processes all the video frames to compute the pose of the camera at every time step with an unknown map scale. When a frame provides distinctive geometrical information, it is inserted in the map as a *keyframe* together with new map points.

Simultaneously, object recognition is performed on as many frames as possible to search for known objects stored in an *object model database*. If there is available information of the location of objects, given by both the SLAM map and previous recognitions, this is exploited to guide the detection in the current image. A successful detection provides an *observation* of an object instance.

Regardless of the recognition algorithm used, a detection obtained from a single image may be spurious or inaccurate. To avoid these problems, instead of placing an object in the map after a first recognition, we insert it in the SLAM map after *accumulating* consistent observations over time. The information given by all the observations is used to triangulate the object points, and hence the pose of the object inside the SLAM map. The resulting points are inserted in the 3D map as *anchor points*, and the cameras that observed them, as *semantic keyframes*. These keyframes are not selected because of a geometrical criteria, but because they contain relevant semantic information. The frames of the observations that do not provide parallax or distinctive geometrical information are discarded. Each triangulation provides us with an estimate of the map scale which we use to globally optimize it.

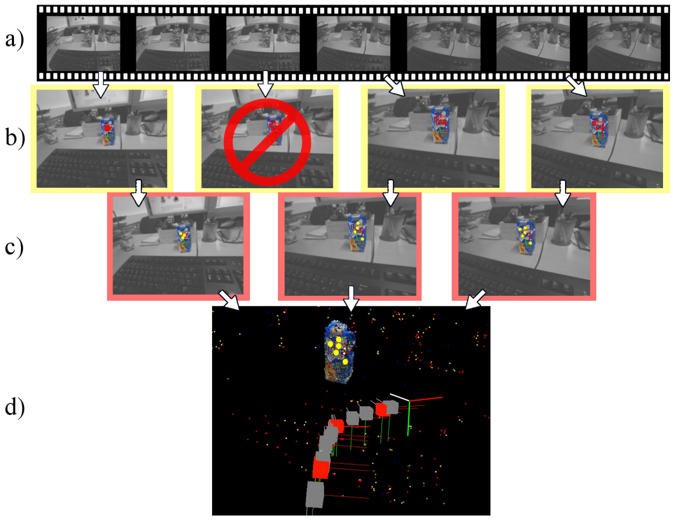


Figure 2: Object insertion with a monocular camera. a) Object detection is performed as fast as possible on the frames of the video stream. b) The bottle is detected in some frames and its observed 2D points (red points) are accumulated. c) When several points are observed with enough parallax (yellow points), their frames are selected as semantic keyframes. Detection frames that offer no parallax are discarded. d) Observations from semantic keyframes are used to triangulate the object and its 3D points. The semantic keyframes (red cameras), the object and its points are inserted in the map, updating its scale.

4. Object-aware SLAM

4.1. Object insertion within the map

The recognitions of objects in single images yielded by our algorithm are used to insert those objects in the SLAM map. To robustly place them, instead of relying on a single detection, we accumulate several of them until we have enough geometrical information to compute a robust 3D pose. This process is depicted by Figure 2 and explained next.

The object detector described in Section 5 searches for objects in as many frames as possible, whereas SLAM uses them to track the camera, so that the pose $T_{WC_i} = [R_{WC_i} \,|\, s\,\mathbf{t}_{WC_i}]$ of each camera i is known, with a map scale s that is initially unknown. A successful recognition of an object model O returns a transformation T_{C_iO} from the camera to the object frame. Since multiple physical instances of the same object model may exist, we check to which instance this detection belongs. We do so by computing a hypothesis of the global pose $T^*_{WO} = T_{WC_i} T_{C_iO}$ of the detected object in the world, and checking for overlap with the rest of the objects of the same model that had been previously observed or are already in the map. Note that this operation is valid only if we already have an estimate of the map scale s. Otherwise, we assume that consecutive detections of the same model come from the same real object. After this, we determine the detection of object O is an observation of O_k, the k-th instance of model O. If there is no overlap with any object observed before, we just create a new instance.

An observation $B^i_{O_k} = \langle T_{WC_i}, T_{C_iO_k}, \mathcal{X}_O, \mathcal{U}_i \rangle$ yields a set of correspondences between some 3D points of model O, $\mathcal{X}_O$, and 2D points of the image taken by camera C_i, $\mathcal{U}_i$. For each correspondence $\langle \mathbf{x}_O, \mathbf{u}_i \rangle \in \langle \mathcal{X}_O, \mathcal{U}_i \rangle$, if the parallax with respect to the rest of observations of $\mathbf{x}_O$ of the same object instance is not significant enough, the corresponding pair is discarded. If an object observation does not offer parallax or new points, it is completely disregarded.

The observations of an object instance are accumulated until the following conditions hold: 1) at least 5 different object points $\mathbf{x}_O$ are observed from two different positions, 2) with at least 3 degrees of parallax between the cameras, and 3) showing no alignment and a good geometrical conditioning. The points are triangulated in the world frame ($\mathbf{x}_W$) and the pairs $\langle \mathbf{x}_O, \mathbf{x}_W \rangle$ are inserted in the map as anchor points. The frames that offered parallax are also inserted as semantic keyframes.

Anchor points play a decisive role on the object SLAM, because they provide the location of the object within the map and set additional geometrical constraints in the BA enabling the map scale estimation. For this reason, anchor points have a different treatment than map points: they are not discarded by the maintenance algorithm of PTAM, are updated using new object observations only and are propagated among the keyframes of the map by using matching cross correlation in a 3×3 pixel region defined around the projected anchor point in the target keyframe. The patches for the correlation are extracted from the semantic keyframes and warped in order to compensate scale, rotation and foreshortening by means of a homography.

By triangulating object points from several observations, we provide a more robust 3D pose than relying on a single detection. Furthermore, this operation is necessary to find the map scale s if our only source of data is a monocular camera. Since we aim for $T^{-1}_{WC_i}\mathbf{x}_W = T_{C_iO_k}\mathbf{x}_O$ for each point, an estimate of

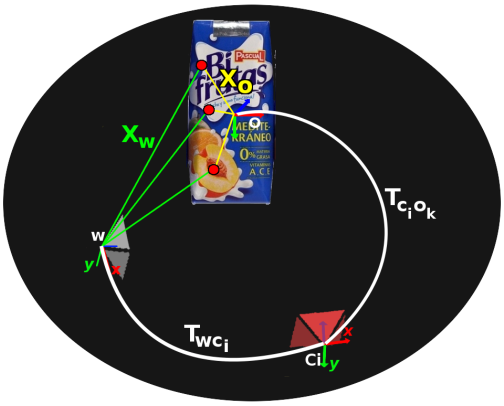


Figure 3: Object pose prior estimation (T_{WO_k}). Red landmarks show the pairs $\langle \mathbf{x}_O, \mathbf{x}_W \rangle$ of object points and anchor points. T_{WC_i} is the pose of the camera, with map scale s already estimated, and $T_{C_iO_k}$ is the pose of the object with respect to the camera for the last observation $B^i_{O_k}$.

the map scale is given by each triangulated object instance:

$$s_{O_k} = \arg\min_{s} \sum_{i} \sum_{\langle \mathbf{x}_O, \mathbf{x}_W \rangle} \left\| R^t_{WC_i} \mathbf{x}_W - R_{C_iO_k} \mathbf{x}_O - \mathbf{t}_{C_iO_k} - s\, R^t_{WC_i} \mathbf{t}_{WC_i} \right\|^2 . \quad (1)$$

To insert the object in the map, we must compute its pose T_{WO_k} in the map world frame. After the first triangulation of an object, we compute an initial pose which is used as a prior in the subsequent SLAM optimization. The object pose prior is computed by composing the information provided by the SLAM and the object detector by means of equation (2). This composition is shown in Figure 3.

$$T_{WO_k} = \left[R_{WC_i} R_{C_iO_k} \,|\, R_{WC_i} \mathbf{t}_{C_iO_k} + \hat{s}\, \mathbf{t}_{WC_i} \right] . \quad (2)$$

The pose of the semantic keyframe T_{WC_i} and the pose of the object with respect to the camera $T_{C_iO_k}$ corresponds to the information of the last observation $B^i_{O_k}$. The scale $\hat{s}$ used is either the scale estimate s_{O_k} computed above, or the map scale s if we already had a previous estimate that had been refined in the optimization stage.

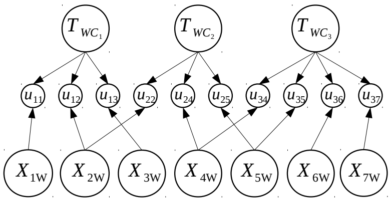
The anchor points, the semantic keyframes and the object pose priors produced by each triangulation are then included in the optimization stage of the SLAM mapping algorithm to obtain more accurate values during the SLAM execution.

4.2. Object SLAM optimization

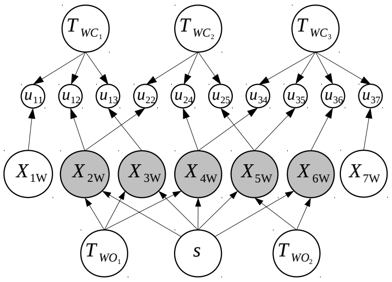
In standard keyframe-based SLAM, a sparse map of points $\mathcal{X}_W$ and the camera location of selected keyframes T_{WC_i} are estimated by means of a joint bundle adjustment (BA). Figure 4(a) shows a Bayesian network representing the estimation problem structure.

The BA minimizes *the map reprojection error*, $\mathbf{e}_{ji}$, between the j-th map point observed by the i-th keyframe and the corresponding measurement $\mathbf{u}_{ji} = \left(u_{ji}, v_{ji}\right)^t$:

$$\mathbf{e}_{ji} = \begin{pmatrix} u_{ji} \\ v_{ji} \end{pmatrix} - \mathrm{CamProj}\left(T^{-1}_{WC_i}\, \mathbf{x}_{jW}\right) . \quad (3)$$



(a) Standard SLAM



(b) Object SLAM

Figure 4: SLAM estimation problem. a) Bayesian network of standard SLAM. T_{WC_i} are the cameras, $\mathbf{x}_{jW}$ the map points and $\mathbf{u}_{ij}$ is the measurement on the image. b) Bayesian network of object SLAM. Some objects are added to the BA, where the object location is represented as T_{WO_k} and the scale s becomes observable. Highlighted map points are those which belong to the objects.

The point in the camera frame $\mathbf{x}_{jC_i} = T^{-1}_{WC_i} \mathbf{x}_{jW}$ is projected onto the image plane through the projection function *CamProj* : $\mathbb{R}^3 \mapsto \mathbb{R}^2$ defined by Devernay & Faugeras [33].

Our goal is to include in the estimation the constraints given by the triangulation of a set of K objects, and then obtain optimized estimates of the map cameras, points, objects and scale. A triangulated object instance $R_{O_k} = \langle T_{WO_k}, \mathcal{X}_O, \mathcal{X}_W \rangle$ comprises a set of 3D points $\mathcal{X}_O$ in the object frame and its corresponding anchored landmarks in the map, with coordinates $\mathcal{X}_W$ in the SLAM world frame. It is well known that from monocular sequences, the scene scale s is unobservable. However, if some map points correspond to a known size scene object (anchor points) the resulting geometrical constraints allow to estimate the SLAM scale. Figure 4(b) shows the new estimation problem structure after including anchor points. New nodes are added to the Bayesian network because new parameters have to be estimated: object locations T_{WO_k} and map scale s, which is observable now, being a single value for all the inserted objects.

Each anchor point $\langle \mathbf{x}_O, \mathbf{x}_W \rangle \in \langle \mathcal{X}_O, \mathcal{X}_W \rangle$ sets a new constraint, the *object alignment error*, $\mathbf{a}_{jk}$, defined as the difference between the positions of the point when both measures are translated into the same object frame of reference:

$$\mathbf{a}_{jk} = \mathbf{x}_{jO_k} - s\, R^t_{WO_k} \mathbf{x}_{jW} + R^t_{WO_k} \mathbf{t}_{WO_k} . \quad (4)$$

We propose a BA to iteratively estimate the scale, the map points, the cameras and the objects, by minimizing a robust objective function combining the reprojection error (3) and the object alignment error (4):

$$\hat{\mathbf{p}} = \arg\min_{\mathbf{p}} \sum_{i=1}^{N} \sum_{j \in \mathcal{S}_i} H\left(\mathbf{e}_{ji}^{t} \Omega^{C} \mathbf{e}_{ji}\right) + \sum_{k=1}^{K} \sum_{j \in \mathcal{S}_k} H\left(\mathbf{a}_{jk}^{t} \Omega^{O} \mathbf{a}_{jk}\right), \quad (5)$$

where N is the total number of keyframes, $\mathcal{S}_i$ is the subset of map points seen by the i-th camera, $\mathcal{S}_k$ is the subset of anchor points of the k-th object instance, Ω^C and Ω^O are the information matrices of the reprojection error and the alignment error respectively. Errors are supposed uncorrelated and follow a Gaussian distribution, thus the covariance matrices are diagonal. Regarding Ω^C, it is a 2x2 matrix and the measurement error is $\sigma^2 = 2^{2l}$, where l is the level of the pyramid in which the feature was extracted. Similarly, Ω^O is a 3x3 matrix with measurement error $\sigma^2 = 0.01^2$. $H(\cdot)$ is the Huber robust influence function [34]:

$$H(x) = \begin{cases} x & \text{if } |x| < \delta^2, \\ 2\delta x^{\frac{1}{2}} - \delta^2 & \text{otherwise.} \end{cases} \quad (6)$$

Here, δ^2 corresponds to the $\chi^2_{0.05}(r)$ distribution, being r the number of degrees of freedom. The δ^2 value for the reprojection error (3) is $\chi^2_{0.05}(2) = 5.991$ and for the alignment error (4) is $\chi^2_{0.05}(3) = 7.815$.

The optimization vector is

$$\mathbf{p} = \left(s,\ \boldsymbol{\nu}_{WC_2}, \ldots, \boldsymbol{\nu}_{WC_N},\ \boldsymbol{\nu}_{WO_1}, \ldots, \boldsymbol{\nu}_{WO_K},\ \mathbf{x}_{1W}, \ldots, \mathbf{x}_{MW}\right), \quad (7)$$

where s is the map scale and $\boldsymbol{\nu}$ represents a transformation parametrized as a 6 component vector (rotation and translation) of the SE(3) Lie group.

While the camera is exploring the scene, new keyframes are inserted in the map. To compute a prior pose for the new keyframes a sliding window is applied to the keyframes of the map and only four neighbor keyframes of the new keyframe and all the visible points are included in a BA, minimizing the reprojection error eq (3). The object pose priors are computed as previously explained in this section, following eq (2). Global BA (5), including all the cameras, points and objects, is performed every time a new keyframe or a new object is inserted within the map.

5. 3D Object recognition with large databases

The object recognition requires a visual vocabulary built from an independent set of images, and a database of models that is created offline. Then, the recognition process is executed online in real time, performing two main steps on a query image taken at position T_{WC_i}: *detection* of several model candidates that fit the image features, and *verification* of the candidates by computing a rigid body transformation between the camera and the objects. The candidates are obtained either by querying all the models in a database based on bag of words, or by taking advantage of previous known locations of objects. The verification step makes use of 2D-to-3D correspondences between image and object model points to find the object pose in the image T_{C_iO}. The result are observations $B^i_O = \langle T_{WC_i}, T_{C_iO}, \mathcal{X}_O, \mathcal{U}_i \rangle$ of the object models O recognized. The SLAM algorithm associates then these results to their corresponding object instances O_k taking the pose of the current camera into account (Section 4.1).

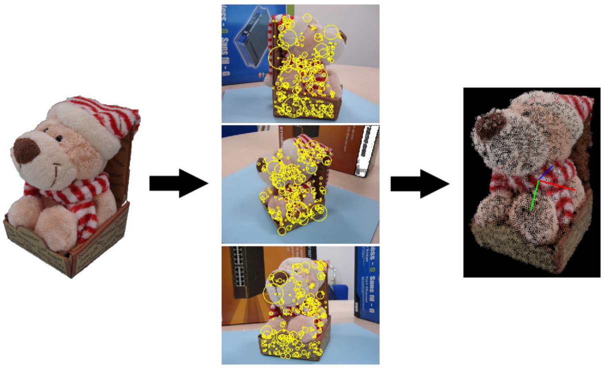


Figure 5: Objects are modeled with a point cloud obtained from multiple view geometry.

5.1. Object models

Our object models are composed of a set 3D points associated to ORB descriptors and an appearance bag-of-words representation for the complete object. ORB features are computational efficient because they describe image patches with strings of 256 bits.

Each object model O is created offline from a set of training images taken from different points of view of the object. We use the Bundler and PMVS2 software [35, 36] to run bundle adjustment on these images and to obtain a dense 3D point cloud of the object $\mathcal{P}_O$, as shown by Figure 5. We keep only those points that consistently appear in at least 3 images. Since objects can appear at any scale and point of view during recognition, we associate each 3D point to several ORB descriptors extracted at different scale levels (up to 2 octaves) and from several training images.

If the point of view of the training images hardly differs, we may obtain 3D points with very similar descriptors that add little distinctiveness. To avoid over-representation, we convert features into visual words and keep the average descriptor per 3D point and visual word [29]. Finally, an appearance-based representation of the object is obtained by converting the surviving binary features of all its views into a bag-of-words vector with a visual vocabulary. This model provides information of all the object surface, so that a single comparison yields a similarity measurement independently of the viewpoint and the scale of the object in the query image.

5.2. Object model database

The object models are indexed in a database composed of a visual vocabulary, an inverted index and a direct index [8]. The visual vocabulary consists in a tree with binary nodes that is created by hierarchical clustering of training ORB descriptors. The leaves of the tree compose the words of the visual vocabulary. We used 12M descriptors obtained from 30607 independent images from Caltech-256 [37] to build a vocabulary with $k = 32$ branches and $L = 3$ depth levels, which yields 33K words. When an ORB feature is given, its descriptor vector traverses the tree from the root to the leaves, selecting at each level the node which minimizes the Hamming distance, and obtaining the final leaf as word. By concatenating the equivalent words of a set of ORB features, we obtain a bag-of-words vector, whose entries are weighted with the term frequency – inverse document frequency (*tf-idf*) value, and normalized with the L_1-norm. This weight is higher for words with fewer occurrences in the training images, since they are expected to be more discriminative.

The inverted index stores for each word in the vocabulary the objects where it is present, along with its weight in that object. When a query image is given, this structure provides fast access to the common words between the query bag-of-words vector and the model one. The direct index stores for each object model the tree nodes it contains and the associated ORB features. This is used to discriminate those features that are likely to match when 2D-to-3D correspondences are required in the verification stage. We can increase the amount of correspondences if we use the direct index to store nodes at other tree levels (coarser discretization levels), with little impact on the execution time [8]. In this work, we store nodes at the first discretization level of the vocabulary tree.

5.3. *Prior knowledge to obtain object candidates*

The first method to obtain detection candidates arises from those objects that have been previously observed or inserted in the map. Detecting objects that are already in the map is useful because we can find new points that were not anchored yet to landmarks. Inserting them help optimize the pose of the object. The process is described in Algorithm 1.

We have two sources of information of observed objects: triangulated objects inserted in the SLAM map, with optimized poses, and non-triangulated accumulated observations. From these, we can estimate the expected pose $T^*_{C_iO_k}$ of each object instance in the current image if the map scale s has been estimated. If it has not, we assume that $T^*_{C_iO_k}$ is the same than the last computed transformation $T_{C_jO_k}$ if it was obtained recently (up to 2 seconds ago). The transformation $T^*_{C_iO_k}$ is computed as:

$$T^*_{C_iO_k} = \begin{cases} T^{-1}_{WC_i}\, T_{WO_k} & \text{if } O_k \text{ in the map,} \\ T^{-1}_{WC_i}\, T_{WC_j}\, T_{C_jO_k} & \text{if } O_k \text{ not in the map but } s \text{ known,} \\ T_{C_jO_k} & \text{if } s \text{ unkown and } i - j \le 2 \text{ secs.} \end{cases} \tag{8}$$

To obtain object candidates, we first extract ORB features from the query image. For every object instance O_k of which we can compute an expected pose $T^*_{C_iO_k}$, if it is visible from the current camera C_i, we project the object model 3D points $\mathcal{P}_O$ on the image to look for correspondences following the same procedure explained in Section 5.5. We estimate a 3D pose from these correspondences by solving the perspective-n-problem [10]. If it is successful, the utilized 2D features are removed from the image and an object observation $B^i_{O_k}$ is produced.

5.4. *General retrieval of object candidates*

After trying to recognize objects previously seen, the general retrieval of object candidates is performed to find new detections. This is described in Algorithm 2.

Objects can appear at any distance from the camera, so the detection of candidates should be robust against scale changes. Sliding window techniques [38, 39] are a common approach to face this difficulty by searching variant size areas of the image repeatedly. In contrast, we rely on dividing the image into regions of interest to perform detections in small areas, merging results if necessary. We run the Quick Shift algorithm [40] on the ORB features of the query image to group together those that are close in the 2D coordinate space to obtain regions of interest. Quick Shift is a fast non-parametric clustering algorithm that separates N-dimensional data into an automatically chosen number of clusters. In our case, each resulting 2D cluster defines a region of interest.

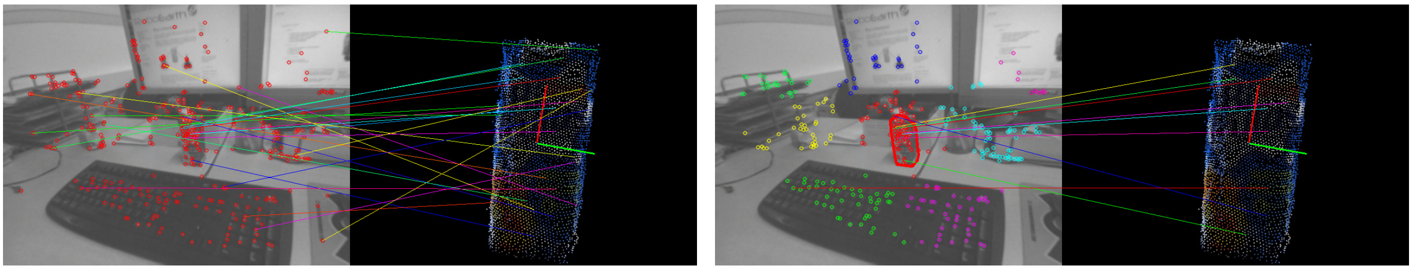
The ORB features of each region of interest are converted into a bag-of-words vector $\mathbf{v}$ that queries the object database individually. The dissimilarity between the query vector and each of the object models $\mathbf{w}$ in the database is measured with a

```
Input: Query image taken at position $T_{WC_i}$
Input: Set $O$ of objects previously observed
Output: Set $\mathcal{B} = \{B^i_{1O}, B^i_{2O}, \dots\}$ of observed objects
$\mathcal{B} \leftarrow \emptyset$
foreach $O_k \in O$ do
    Compute expected pose $T^*_{C_iO_k}$
    Project $\mathcal{P}_O$ on image with $T^*_{C_iO_k}$
    Find new 2D-to-3D correspondences
    Estimate 3D pose to obtain observation $B^i_{O_k}$
    if pose found then
        Remove image features $\mathcal{U}_i \in B^i_{O_k}$ from image
        $\mathcal{B} \leftarrow \mathcal{B} \cup \{B^i_{O_k}\}$
    end
end
return $\mathcal{B}$
```

Algorithm 1: Recognition of objects previously observed

```
Input: Query image taken at position $T_{WC_i}$
Output: Set $\mathcal{B} = \{B^i_{1O}, B^i_{2O}, \dots\}$ of observed objects
$\mathcal{B} \leftarrow \emptyset$
Divide image feature into regions of interest
foreach region of interest do
    Query object database
    Compute correspondences with the top-10 candidates
end
foreach object candidate do
    Join correspondences from all the regions
    Verify detection and obtain observation $B^i_O$
    if detection verified then $\mathcal{B} \leftarrow \mathcal{B} \cup \{B^i_O\}$
end
return $\mathcal{B}$
```

Algorithm 2: General object recognition



(a) Putative correspondences from the entire image

(b) Putative correspondences from regions

Figure 6: Example of putative correspondences obtained in an image with 356 features. In (a), all the features are used to query an object database of 500 models and to compute correspondences. The correct object is the 7th candidate after querying, and 24 putative correspondences are computed, where 18 are incorrect. The object pose cannot be successfully verified after 100 random iterations trying sets of correspondences. On the other hand, in (b), regions of features are used individually to query the database and to produce putative correspondences. In this case, the correct object appears in the 1st position (out of 500) when the region that contains it is queried. In total, 9 correspondences are computed, where just 3 are incorrect. This makes it possible to verify the object and obtain its pose after 34 iterations.

score based on the Kullback-Leibler (KL) divergence [41]. This score s_{KL} benefits from the inverted index since its computation requires operations between words in common only, while the properties of the KL divergence are kept (*cf.* Appendix A):

$$s_{\mathrm{KL}}(\mathbf{v}, \mathbf{w}) = \sum_{v_i \neq 0 \,\wedge\, w_i \neq 0} v_i \log \frac{\varepsilon}{w_i}, \tag{9}$$

where ε is a positive value close to 0. In Section 6.1 we compare the performance of the KL divergence with other popular metrics. The sparsity of vectors $\mathbf{v}$ and $\mathbf{w}$ highly differs, since an object model may comprise thousands of words whereas an image region just a dozen. This seems a difficulty for the retrieval of the correct model, however the bag-of-words scheme can handle this situation because it can compare vectors independently of their number of words. If a single word in a region matches a model, its tf-idf weight can already produce an object candidate with a s_{KL} score. Perceptual aliasing may yield wrong candidates, but the correct model is expected to produce more correct word matches, lowering its dissimilarity value. As a result, the correct object model is likely to be retrieved even if only a few words in common are found. Thus, the top-10 object models that offer the lowest dissimilarity score with vector $\mathbf{v}$ are selected as detection candidates for each region of interest.

Then, correspondences between the 2D image points and the 3D model points are obtained. This operation is sped up by using the direct index to filter out unlikely correspondences [8]. Our segmentation into regions bears a resemblance with other approaches. For example, MOPED [5] tries to recognize an object only with the correspondences obtained in each region, merging later the detections if their poses overlap. In contrast, we merge the correspondences of each region of interest according to their associated object model before computing any pose. This prevents from missing detections due to oversegmented regions with few correspondences. We finally find the pose of the object candidates, or discard them, in the object verification stage.

Figure 6 shows an example of how regions of interest can help find small objects: a carton bottle is searched for from a database with 500 object models. In Figure 6(a), the query image is not divided into regions and the entire image queries the database. As a result, the background makes the correct model appear as the 7th best candidate, and prevents from obtaining correct correspondences, missing the detection. On the other hand, as shown by Figure 6(b), when small regions are considered, we find the correct model as the 1st candidate of its region, obtaining a better inlier ratio of correspondences (6 out of 9), being able to verify the recognition. Thus, in addition to detect small objects, regions of interest are helpful to establish better point correspondences. Figure 7 shows an example in which a toy van is divided into two regions. Since region correspondences are finally merged, it can be correctly recognized.

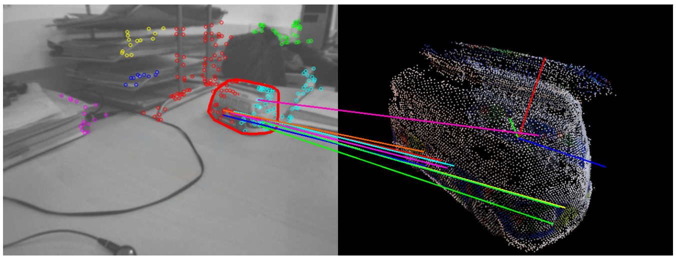
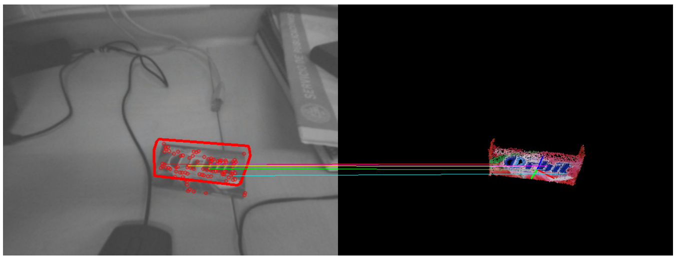


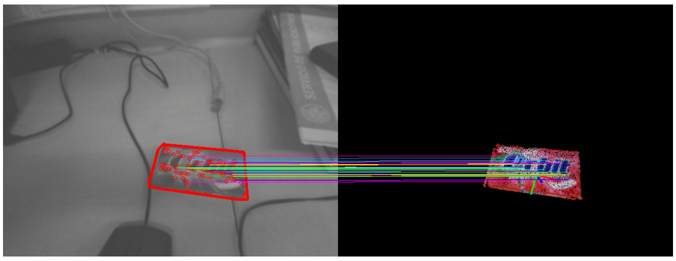
Figure 7: Example of a real object that lies in two different image regions. Since putative correspondences from different regions are merged, it is correctly found with 9 inliers.

5.5. *Object verification and pose estimation*

After obtaining putative 2D-to-3D correspondences between the query image and the object candidates, we try to verify and find the pose of each object by iteratively selecting random subsets of correspondences and solving the perspective-*n*-point problem (PnP) [10]. Plenty of algorithms based on random sample consensus (RANSAC) [10] exist to achieve this. For example, progressive sample consensus (PROSAC) [42] arranges the correspondences according to their distance in the descriptor space. Then, ordered permutations of low distance are selected as subsets for a parametrized number of tries, after which the algorithm falls back to RANSAC. This is usually much faster than RANSAC when the pose can be found. However, in the presence of mismatching correspondences with low descriptor distance (e.g. due to perceptual aliasing), PROSAC



(a) Some randomly selected putative correspondences yield a first pose.



(b) A final pose is computed from the new correspondences obtained after projecting the object model.

Figure 8: Computation of a robust candidate pose in a DISAC iteration.

may spend several tries trying subsets with outliers. Since we set a low number of maximum iterations (50) to limit the impact in the execution time, we propose a variation of PROSAC that eases the rigidity of the fixed permutations and is thus more flexible when there are low-distance mismatches. We coined it *distance sample consensus* (DISAC) and consists in drawing correspondences c_j from the set of correspondences $C = \{c_1, \dots, c_n\}$ with a probability inversely proportional to its Hamming distance h:

$$P(c_j) = \frac{1}{h(c_j) \displaystyle\sum_{k=1}^{n} \frac{1}{h(c_k)}}. \tag{10}$$

To avoid numerical inconsistencies, we set $h(c_i) = 1$ when the distance is exactly 0. Now, in the case of outliers, there is a non-zero probability to avoid them even in the first iterations of DISAC.

If a 3D pose is found with the selected subset of correspondences, we try to refine it by selecting additional correspondences that were not given by the direct index. For that, we project the object model 3D points $\mathcal{P}_O$ on the query image, obtaining a set of 2D points. For each visible point $\mathbf{x}_O \in \mathcal{P}_O$, we search a 7×7 patch centered at its projection for a matching ORB image feature $\mathbf{u}$. We consider them to match if any of the ORB descriptors associated to the corresponding 3D point is at a Hamming distance lower than 50 units, which assures a low ratio of mismatches [9]. If new correspondences are found, we compute a new refined pose $\hat{T}_{C_iO}$, as shown by Figure 8. We measure the quality of a pose $\hat{T}_{C_iO}$ with a reformulation of Torr & Zisserman's M-estimator [43] on the reprojection error that also takes into account the number of inlier correspondences:

$$s_{\text{DISAC}} = \sum_{\langle \mathbf{u}, \mathbf{x}_O \rangle} \max\left(0,\ \mu_e - \left\| \mathbf{u} - \text{CamProj}(\hat{T}_{C_iO}\, \mathbf{x}_O) \right\|\right), \tag{11}$$

where *CamProj* is the projection function presented in Section 4.2, and μ_e, a threshold in the reprojection error set to 3px.

Figure 9: Testing objects of the desktop dataset

Figure 10: Example of Nister & Stewenius's [25] objects

We keep the refined transformation of maximum s_{DISAC} score of all the DISAC samples for each object candidate, if any, verifying the recognition and finding the transformation T_{C_iO} between the current camera and the object. This, together with the collection of 2D-to-3D correspondences, composes the object observation $B^i_{O_k} = \langle T_{WC_i}, T_{C_iO_k}, \mathcal{X}_O, \mathcal{U}_i \rangle$ that feeds the SLAM algorithm.

6. Experimental evaluation

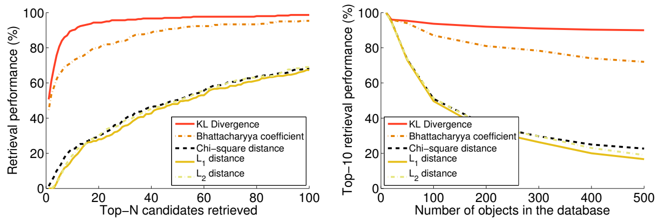
Our system has been implemented in C++, as modules of the Robot Operating System (ROS) [44], exploiting parallelization with OpenMP [45] in the object candidate detection and verification steps. All the tests were done on a Intel Core i7 @ 2.67GHz PC.

We evaluate our system in five different datasets with sets of from 7 to 500 objects: the *Desktop* dataset, used for testing purposes; one of the sequences of the *RGB-D SLAM Dataset* [46], which provides ground truth of the camera pose; the *Aroa's room* dataset, a child's real room with dozens of different objects; the *Snack* dataset, a sequence that shows several instances of the same object models and force camera relocation; and the *Snack with clutter* dataset, a small area with repeated objects in a small space with occlusion and background clutter.

6.1. Desktop dataset

The desktop dataset is a 6'26" sequence of 640 × 480 images collected with an Unibrain camera on a desktop area, which we used to test our object recognition algorithm. The dataset shows the 6 objects illustrated in Figure 9, whose largest dimension is between 10 and 20 cm. These were modeled with consumer photo cameras. In addition to these objects, we created models from the image dataset provided by Nister & Stewenius [25]. These are sets of 4 images depicting general objects, as those shown in Figure 10, under different points of view and illumination conditions. We used up to 494 sets of images to populate our object databases with models to be used as distractors for the object candidate detection step.

We show first the results of the object candidate retrieval step from single images.



(a) (b)

Figure 11: Performance of several similarity scores to retrieve candidates when (a) a database with 500 objects is queried, and (b) the top-10 candidates are retrieved from databases of different sizes.

Figure 12: Example of correct detections in the desktop dataset, with 500 objects in the database.

In addition to the KL divergence, we evaluated the performance to detect object models of other similarity metrics popular in bag-of-words approaches [47]. We selected a set of 300 640 × 480 images that show one object at a time from a distance of between 20 and 70 cm. Then, we manually masked out the background and query the database varying the amount of stored objects, computing the KL divergence, the Bhattacharyya coefficient, the χ^2 distance and the L_1-norm and L_2-norm distances. Figure 11 shows the retrieval performance of each metric, defined as the percentage of correct object candidates returned. Figure 11(a) shows the performance against a database of 500 models regarding the number of top results that we consider candidates. The KL divergence offers a higher performance in comparison with the rest of the metrics. We can see that the performance increases remarkably when we consider as candidates up to the top-10 results, where the performance stalls. Since increasing the number of candidates will hit the execution time in the verification step, we choose to select the top-10 object candidates for each region of interest. Figure 11(b) shows the evolution of the metric scores when the top-10 candidates are selected from databases comprising from 10 to 500 objects. We can see that the KL divergence always provides the best performance.

When running our complete Object SLAM approach in the

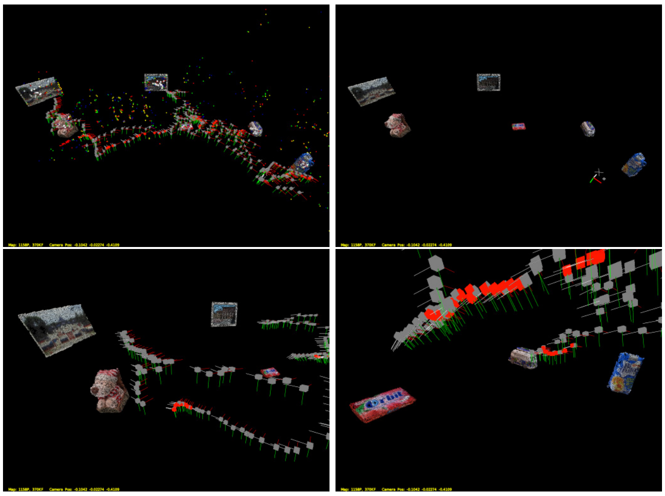


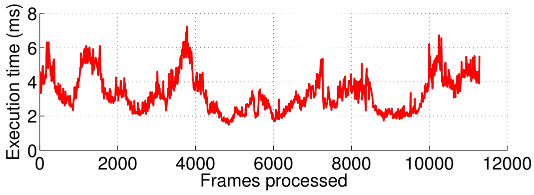
Figure 13: Resulting map of the desktop dataset with 500 models in the database. All the objects are correctly located in the space with no false positives.

(a) (b)

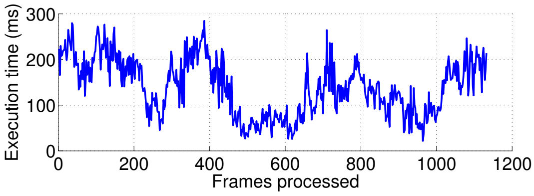
Figure 14: Example of robustness against inaccurate detections. a) Prior knowledge about the location of the objects (blue outline) is used to recognize objects. Some features (red dots) of the card are correctly matched with the model, but these are ill-distributed and the pose calculated in this single frame is inaccurate. b) In the next frame, the actual pose of the card remains correct because it is computed from all the accumulated observations. This allows to accurately detect the card again.

desktop dataset, the 6 objects were correctly located in the space, with no false positive detections. Figure 12 shows some correct object detections in single images. Figure 13 shows the obtained map, including objects, keyframes (gray cameras), semantic keyframes (red cameras) and points.

In some cases, the pose of an object obtained from a single-image detection may be inaccurate. Any algorithm is subject to this due to several factors, such as perceptual aliasing, or a bad geometrical conditioning. Since we do not rely on a single detection to locate an object in the space, our system is able to overcome from detection inaccuracies. For example, consider the case shown by Figure 14(a). We have two prior locations of the chewing gum box and a card (blue outline), from which the two poses of the objects are obtained (red outline). However, the points observed of the card (red dots) are not widely distributed, causing the recovered pose to be ill conditioned. This results in an inaccurate object pose although the 2D-to-3D correspondences are correct. Since we had additional geometrical information accumulated of previous observations of the card, its actually computed pose remained correct, as can be seen in



(a) SLAM Tracking



(b) Object recognition

Figure 15: System execution time in the Desktop dataset

	Median	Max.
Our system (Object recognition + SLAM)	0.14	0.34
MOPED [5] (Object recognition)	0.52	0.95

Table 1: Execution time of the object recognition stage of our system compared with MOPED, with a database of 500 objects (s/image).

	Our system	Our system, no priors	MOPED [5]
Bottle	137	41	81
Van toy	104	33	8
Box	91	56	63
Lion toy	80	5	0
Card 1	258	227	166
Card 2	200	118	121
Total detections	870	480	439

Table 2: Number of detections in the desktop dataset with a database of 500 objects. By exploiting the priors given by the SLAM map we can provide more detections, even compared with MOPED, a state-of-the-art single-image recognition algorithm.

the next detection, shown by Figure 14(b), where the recognition of the card is fully accurate.

The sequence is processed in real time. Figure 15(a) shows the execution time of the SLAM tracking (block averaged for readability), which takes 3.3 ms on average. Figure 15(b) shows the execution time taken by the object recognition process with each image, being 138 ms per image on average. Since the tracking and the recognition run in parallel, the SLAM map is created successfully in real time independently of the time taken by the object recognition. It is worth mentioning that the total execution time of our system, which performs object recognition and SLAM, is lower to the time consumption of other approaches that run object recognition only, as we show in Table 1. MOPED [5] is a state-of-the-art algorithm, highly optimized to use thread parallelization, that recognizes 3D objects from SIFT features and retrieve its pose in the space from single images. To obtain its results, we ran MOPED in the desktop dataset, on the same computer and with the same 500 models. Our object recognition algorithm yields similar results to those by MOPED, as shown in Table 2 (*our system, no priors*). Furthermore, this table shows that our full system is able to provide more detections of objects when we exploit the prior locations obtained by the SLAM optimization over time. The advantage of the prior information is well illustrated with the lion toy. It is a challenging object to recognize because its repetitive red and white stripes are prone to cause mismatches. Our algorithm triangulated its initial pose from 4 observations, setting a prior location that enabled subsequent successful detections. This highlights the fact that any object recognition algorithm can be enhanced by the SLAM approach we propose.

6.2. *RGB-D SLAM Dataset*

Sturm *et al.* [46] acquired several video sequences with a RGB-D Kinect camera to evaluate RGB-D SLAM systems. These are conveniently provided with the ground truth trajectory of the camera, obtained from a high-accuracy motion-capture system. We utilized this dataset to measure the accuracy on camera location by our system with monocular images.

We made use of several of their sequences: one to evaluate our Object SLAM, and the others to train the models of the objects that appear in the former. We ran our system on

Figure 16: Objects of the RGB-D SLAM Dataset

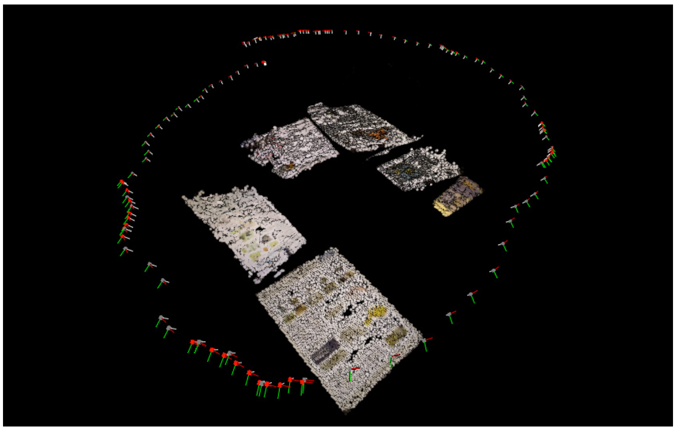


Figure 17: Map of objects created in the RGB-D SLAM sequence

	Translation (cm)	Rotation (deg)	RMSE (cm)
Our system	3.4 ± 2.5	1.4 ± 0.7	4.2
RGB-D SLAM [12]	9.6 ± 5.7	3.9 ± 0.6	11.2
PTAM [11]	5.0 ± 2.4	2.1 ± 0.9	5.6

Table 3: Absolute Trajectory Error. Translation and rotation mean error (mean ± std) and translation RMSE of our system in comparison with RGB-D SLAM and PTAM.

	Translation (cm)	Rotation (deg)	RMSE (cm)
Our system	5.1 ± 3.3	1.5 ± 0.9	6.1
RGB-D SLAM [12]	17.7 ± 10.5	4.9 ± 2.3	20.6
PTAM [11]	6.0 ± 3.1	1.1 ± 0.5	6.8

Table 4: Relative Pose Error . Translation and rotation mean error (mean ± std) and translation RMSE of our system in comparison with RGB-D SLAM and PTAM.

the sequence titled *freiburg3 nostructure texture near withloop* in which the camera describes a loop, moving around some posters lying on the floor (shown by Figure 16). We built the object models with the *validation* version of the previous sequence and with the sequence *freiburg3 nostructure texture far*, which shows the same posters but from different camera positions and distances. Thus, we do not use the same data for training and evaluation. We created the object models by taking sparse RGB images of each poster (~20) and processing them as explained in Section 5.1. We set their scale by reconstructing their 3D point clouds and measuring the real distance between pairs of points. As in the desktop dataset, besides these 8 models, we filled the database up to 500 models.

Our system is able to recognize and place all the posters in the scene but two: the smallest one, for which no detections are obtained due to its size, and the one in the middle of the scene, because the camera moves very close to the floor and it barely focuses the center of the trajectory.

The produced map is shown in Figure 17. Its scale is successfully estimated from the triangulations of the objects. The average error obtained in the poses of the keyframes (the only poses that are optimized by the BA) is 3.4 cm in translation and 1.4 degrees in rotation.

We compared our system with the RGB-D SLAM algorithm [12] and PTAM [11]. RGB-D SLAM creates a graph with the poses of the camera, linking the nodes with the relative transformation between them. These are obtained by computing 3D-to-3D correspondences from SURF features between pairs of images, by using the RGB and depth data of a Kinect camera. The graph is then optimized with g2o [32]. Since their sensor provides depth, their map is at real scale.

A qualitative comparison of the trajectories estimated by our system, RGB-D SLAM and PTAM is shown in Figure 18 along with the ground truth. We obtained this figure by aligning, according to the timestamps of the images, the evaluated trajectories with the ground truth by means of Horn's method [48]. Since PTAM is a monocular system, we computed the scale by aligning the first meter of the trajectory in 7DoF.

For a quantitative comparison we ran the sequence on each system 10 times and report the average Absolute Trajectory Error (ATE) and Relative Pose Error (RPE) [46]. ATE compares the absolute distances between the estimated and the ground truth trajectories after alignment; the results are on Table 3. For the rotation error, we computed the circular mean and standard deviation of the angles between the orientation of each pose with its corresponding one in the ground truth. We also show the root-mean-square error (RMSE) on translation. We report the relative pose error on Table 4, which measures the local accuracy of the trajectory over a fixed time interval and corresponds to the drift of the trajectory. Instead of restricting to evaluate in a fixed time interval, we compute the average over all possible time intervals [46]. The average ATE yielded by RGB-D SLAM is 9.6 cm in translation and 3.9 degrees in rotation. We observed this system creates a bias in the scale of the trajectories in some datasets that is producing this error. However, the origin of this bias is not clear. We conclude that by introducing objects, our monocular system can retrieve the real

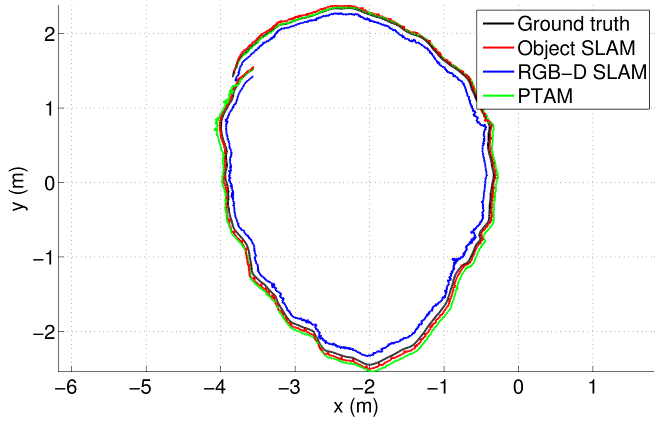
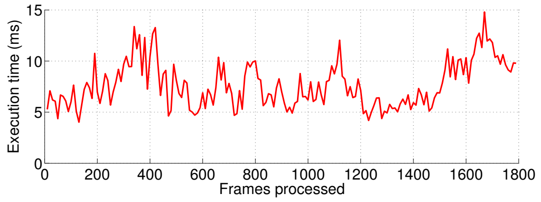
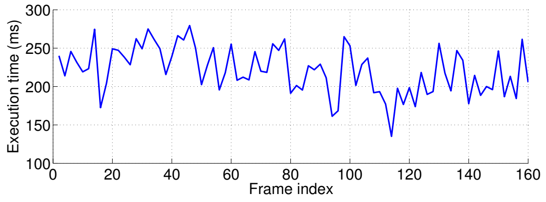


Figure 18: Trajectory and scale estimated by our Object SLAM system in comparison with RGB-D SLAM [12] and PTAM [11] in the RGB-D SLAM Dataset.



(a) SLAM Tracking



(b) Object recognition

Figure 19: System execution time in the RGB-D SLAM dataset

Figure 20: Aroa's room and 4 of the 13 objects modeled

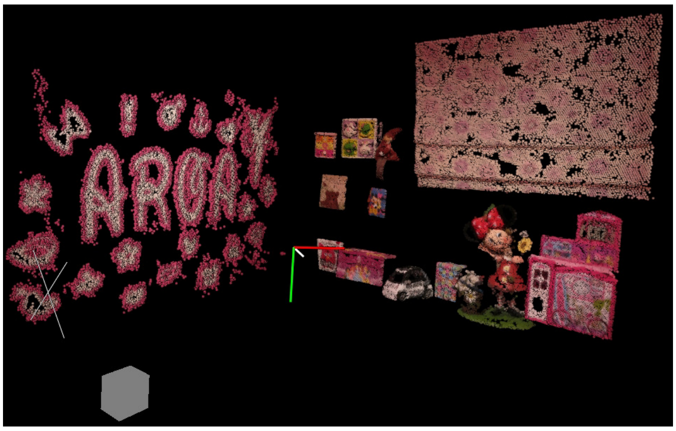
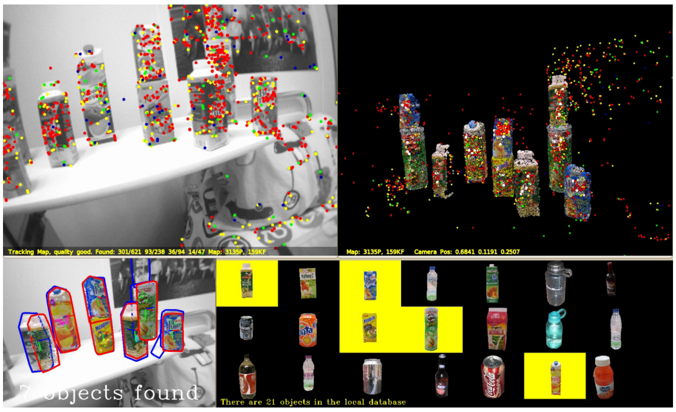


Figure 21: Map of objects created in Aroa's room

Figure 22: The 5 models out of 21 that appear in the Snack dataset



```
Tracking Map, quality good. Found: 301/621 93/238 36/94 14/47 Map: 3135P, 159KF
```

```
Map: 3135P, 159KF    Camera Pos: 0.6841 0.1191 0.2507
```

```
7 objects found
```

```
There are 21 objects in the local database
```

Figure 23: Object SLAM running in the Snack dataset

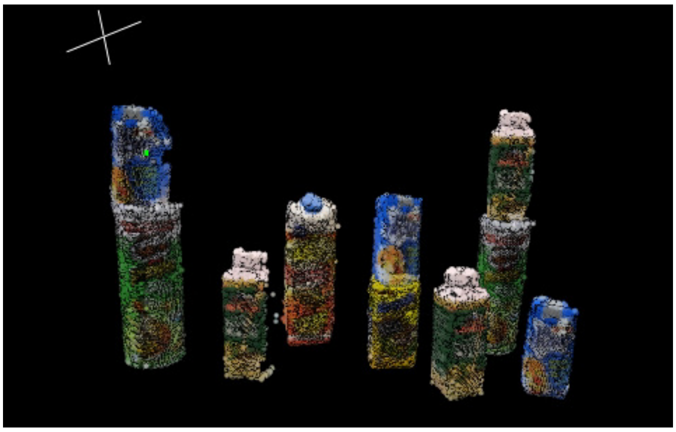


Figure 24: Map of objects created in the Snack dataset

scale of the scene and create maps that are more accurate and contain richer information (3D objects and points).

Figure 19(a) shows the execution time of SLAM tracking, which takes 7.6 ms on average. The execution time taken by the object recognition process is 220 ms per image on average with a database of 500 objects, and it is shown in Figure 19(b).

6.3. *Aroa's Room and Snack datasets*

We collected three more sequences to show qualitative results of our system in challenging scenarios.

The *Aroa's room* dataset was collected with a Kinect camera (using the RGB sensor only) in a child's real room, where we modeled a set of 13 objects (toys and pieces of furniture, such as blinds and a wall poster) of diverse size, with a consumer photo camera. Figure 20 shows the environment and some of the objects. The main challenge of this scenario is the highly textured clutter that can produce mismatches in the object recognition. Figure 21 shows the resulting map, where all the objects are located. The full execution can be watched on video[1].

The *Snack* dataset shows a sequence recorded with a Unibrain camera, where 10 bottles and cans, some of them identical, are placed together on a table. The database is filled with 21 models of snacks. Figure 22 shows the 5 models that actually appear on the table. In this sequence, we intentionally made SLAM lose tracking of the camera on two occasions. The observations of the objects are still merged once the camera is relocated, obtaining successful triangulations. Figure 23 shows the system running. The two windows on top show the PTAM tracking and the 3D map created so far. Below, on the left, the current object detections (red outline) and the prior knowledge about their position (blue outline). Figure 24 shows the objects in the final map. The full sequence can be watched on video[2].

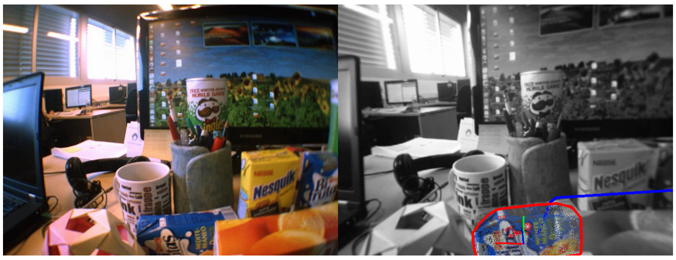
The *Snack with clutter* dataset shows a sequence with 6 bottles and cans, some of them identical, in the highly cluttered and textured scenario depicted by Figure 25. The main challenge of this scenario for the object recognition is that objects are placed very close each other and with remarkable occlusion, so that regions of features may not separate objects accurately. In spite of that, the object detector yields successful results in single frames, as those depicted by Figure 26. In Figures 26(a)

---

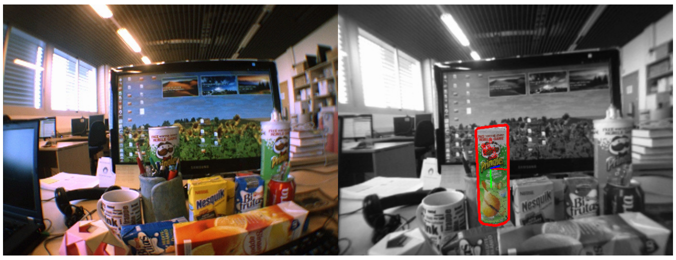
[1] http://youtu.be/cR_tkKpDZuo

[2] http://youtu.be/C3z62h6NPt4

Figure 25: Snack with clutter scenario



(a) An occluded bottle is detected without prior information



(b) An occluded can is detected without prior information

(c) All objects are detected with prior information

Figure 26: Successful detections in the Snack with clutter scenario

and 26(b) both the bottle and the can are found before any prior information is available, and even if only a small part of them is visible. Although all the objects present in those two frames are not recognized, those detections create prior information that is exploited in next frames, making it possible to detect all the object, as shown by Figure 26(c). This exhibits the ability of our system to exploit the information provided by a sequence of images instead of working in a single image basis. Finally, our system produces the map shown by Figure 27. The full sequence can be watched on video[3].

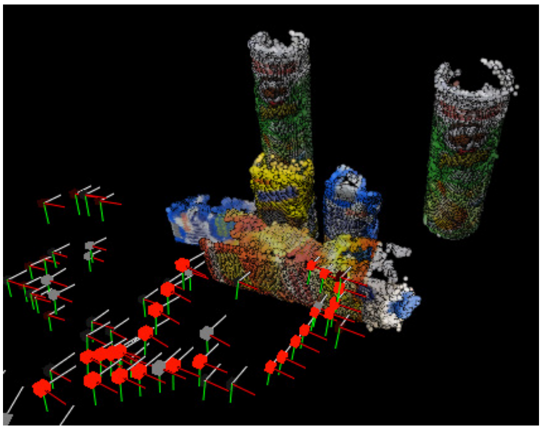


Figure 27: Map of objects created in the Snack with clutter dataset

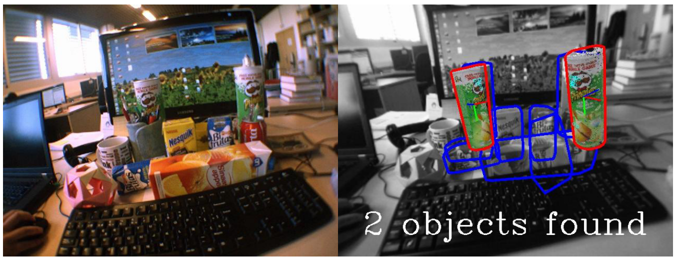


Figure 28: Wrong priors are created around the blue bottles by inaccurate detections, but they are are not triangulated and the map remains correct.

These three sequences show that our system can create consistent 3D maps of points and objects handling very different objects at the same time, dealing with several instances of the same models, in highly cluttered and occluded scenes and even in cases in which track is lost.

Our system provides safety checks at different stages to keep the map consistent. The erroneous observations are due to spurious detections. These rarely occur because of the feature match constraints imposed in the object recognition stage. If they happen or the computed pose is little accurate, the observation accumulation stage prevents the wrong detections from damaging the map because several consistent observations with wide parallax are very unlikely. Figure 28 illustrates this case. The blue lines outline the pose provided by the prior information. There is a prior around each object and two additional wrong ones around the blue bottles. These were created by inaccurate object detections. Since these observations did not match the correct detections, they were accumulated as new instances of the bottle. However, they are not triangulated because they are not supported by other observations. In the rare case that a wrong instance was triangulated and anchor points created, the Huber robust influence function (equation 6) would decrease its impact in the optimization stage when there were enough correct anchor points in the map, keeping the map consistent.

---

[3] http://youtu.be/u8gvKahWt1Q

7. Conclusions

We have presented an object-aware monocular SLAM system that includes a novel and efficient 3D object recognition algorithm for a database up to 500 3D object models. On the one hand, we have shown how embedding the single frame bag-of-words recognition method in the SLAM pipeline can boost the recognition performance in datasets with dozens of different objects, repeated instances, occlusion and clutter. We believe that this benefit is not only achievable by this technique but by any other recognition method embedded within the SLAM pipeline that can exploit the accumulated observations of objects.

On the other hand, inclusion of objects adds to the SLAM map a collection of anchor points that provides geometrical constraints in the back-end optimization and enables the real map scale estimation. We have shown our system can yield more accurate maps than other state-of-the-art algorithms that use RGB-D data.

There is a case we have not addressed in this work: when the first object inserted in the map is originated by wrong observations. This would cause a first incorrect scale estimate and lead to a missized map. This may be tackled by inspecting the variance of the scale estimates given by each object triangulation, so that any observation with an inconsistent scale could be eliminated. Alternatively, the problem might also be avoided if an initial rough scale estimate is available; for example, from the odometry or IMU sensors with which robots and mobile devices are usually equipped. Nevertheless, because of the safety steps of our approach, this case can rarely occur and it did not happen in our experiments.

Including objects in maps paves the way to augment them with semantic data, providing enriched information to a user, or additional knowledge about an environment to an operating robot [49]. We can use this knowledge in a future work to reason about the mobility of objects, making it possible to allow object frames to move in the 3D space, creating dynamic maps.

Appendix A. Efficient KL-divergence computation

Let $\mathbf{v}$ and $\mathbf{w}$ be two vectors such that $\|\mathbf{v}\|_1 = \|\mathbf{w}\|_1 = 1$, $\mathcal{V} = \{i \mid v_i \neq 0\}$, $\overline{\mathcal{V}} = \{i \mid v_i = 0\}$, and analogously for $\mathcal{W}$ and $\overline{\mathcal{W}}$. The Kullback-Leibler divergence is defined as

$$\mathrm{KL}(\mathbf{v}, \mathbf{w}) = \sum_{\mathcal{V}} v_i \log \frac{v_i}{w_i}. \tag{A.1}$$

To avoid undetermined values, we substitute w_i by a constant value $\varepsilon \to 0^+$ when $w_i = 0$, so we can rewrite (A.1) as

$$\begin{aligned}
\mathrm{KL}(\mathbf{v}, \mathbf{w}) &= \sum_{\mathcal{V}\cap\mathcal{W}} v_i \log \frac{v_i}{w_i} + \sum_{\mathcal{V}\cap\overline{\mathcal{W}}} v_i \log \frac{v_i}{\varepsilon} && \text{(A.2)} \\
&= \sum_{\mathcal{V}\cap\mathcal{W}} v_i \log \frac{v_i}{w_i} + \sum_{\mathcal{V}} v_i \log \frac{v_i}{\varepsilon} - \sum_{\mathcal{V}\cap\mathcal{W}} v_i \log \frac{v_i}{\varepsilon} && \text{(A.3)} \\
&= \sum_{\mathcal{V}} v_i \log \frac{v_i}{\varepsilon} + \sum_{\mathcal{V}\cap\mathcal{W}} v_i \log \frac{\varepsilon}{w_i}. && \text{(A.4)}
\end{aligned}$$

Since one of the addends depends only on vector $\mathbf{v}$, we remove it when we want to compare the divergence between $\mathbf{v}$ (a query vector) with other vectors $\mathbf{w}$ (object models). Therefore, our score results in

$$s_{\mathrm{KL}}(\mathbf{v}, \mathbf{w}) = \sum_{\mathcal{V}\cap\mathcal{W}} v_i \log \frac{\varepsilon}{w_i}. \tag{A.5}$$

Acknowledgments

This research has been partly funded by the European Union under project RoboEarth FP7-ICT-248942, the DGA-FSE (T04 group), the Dirección General de Investigación of Spain under projects DPI2012-36070, DPI2012-32168 and the Ministerio de Educación (scholarship FPU-AP2010-2906).

Special thanks to Aroa Gálvez for allowing us to use her room and play with her toys.

References

[1] R. F. Salas-Moreno, R. A. Newcombe, H. Strasdat, P. H. J. Kelly, A. J. Davison, SLAM++: Simultaneous Localisation and Mapping at the Level of Objects, in: IEEE Computer Vision and Pattern Recognition (CVPR), 2013.

[2] J. Civera, D. Gálvez-López, L. Riazuelo, J. D. Tardós, J. M. M. Montiel, Towards semantic slam using a monocular camera, in: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2011, pp. 1277 –1284.

[3] R. Castle, D. Murray, Keyframe-based recognition and localization during video-rate parallel tracking and mapping, Image and Vision Computing 29 (8) (2011) 524 – 532.

[4] M. Lutz, D. Stampfer, C. Schlegel, Probabilistic object recognition and pose estimation by fusing multiple algorithms, in: IEEE International Conference on Robotics and Automation (ICRA), 2013, pp. 4244–4249.

[5] A. Collet, M. Martinez, S. S. Srinivasa, The MOPED framework: Object Recognition and Pose Estimation for Manipulation, The International Journal of Robotics Research 30 (10) (2011) 1284 – 1306.

[6] T. Grundmann, W. Feiten, G. Wichert, A gaussian measurement model for local interest point based 6 dof pose estimation, in: IEEE International Conference on Robotics and Automation (ICRA), 2011, pp. 2085 –2090.

[7] D. Pangercic, V. Haltakov, M. Beetz, Fast and robust object detection in household environments using vocabulary trees with sift descriptors, in: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Workshop on Active Semantic Perception and Object Search in the Real World, San Francisco, CA, USA, 2011.

[8] D. Gálvez-López, J. D. Tardós, Bags of binary words for fast place recognition in image sequences, IEEE Transactions on Robotics 28 (5).

[9] E. Rublee, V. Rabaud, K. Konolige, G. Bradski, ORB: An Efficient Alternative to SIFT or SURF, in: IEEE International Conference on Computer Vision, 2011, pp. 2564–2571.

[10] M. A. Fischler, R. C. Bolles, Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography, Communications of the ACM 24 (6) (1981) 381–395.

[11] G. Klein, D. Murray, Parallel Tracking and Mapping for Small AR Workspaces, in: IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR), 2007, pp. 225–234.

[12] F. Endres, J. Hess, N. Engelhard, J. Sturm, D. Cremers, W. Burgard, An evaluation of the RGB-D SLAM system, in: IEEE International Conference on Robotics and Automation (ICRA), 2012, pp. 1691–1696.

[13] R. Castle, G. Klein, D. Murray, Combining monoslam with object recognition for scene augmentation using a wearable camera, Image and Vision Computing 28 (11) (2010) 1548 – 1556.

[14] H. Strasdat, J. M. M. Montiel, A. J. Davison, Real-time monocular slam: Why filter?, in: IEEE International Conference on Robotics and Automation (ICRA), 2010, pp. 2657–2664.

[15] S. Bao, S. Savarese, Semantic structure from motion, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2011, pp. 2025 –2032.

[16] S. Bao, M. Bagra, Y.-W. Chao, S. Savarese, Semantic structure from motion with points, regions, and objects, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2012, pp. 2703 –2710.

[17] N. Fioraio, L. Di Stefano, Joint detection, tracking and mapping by semantic bundle adjustment, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2013, pp. 1538–1545.

[18] R. A. Newcombe, A. J. Davison, S. Izadi, P. Kohli, O. Hilliges, J. Shotton, D. Molyneaux, S. Hodges, D. Kim, A. Fitzgibbon, Kinectfusion: Real-time dense surface mapping and tracking, in: 10th IEEE international symposium on Mixed and augmented reality (ISMAR), 2011, pp. 127–136.

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

[20] T. Wang, X. He, N. Barnes, Learning structured Hough voting for joint object detection and occlusion reasoning, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2013.

[21] O. Barinova, V. Lempitsky, P. Kholi, On detection of multiple object instances using hough transforms, IEEE Transactions on Pattern Analysis and Machine Intelligence 34 (9) (2012) 1773–1784.

[22] M. Sun, G. Bradski, B.-X. Xu, S. Savarese, Depth-encoded hough voting for joint object detection and shape recovery, in: European Conference on Computer Vision (ECCV), Vol. 6315 of Lecture Notes in Computer Science, 2010, pp. 658–671.

[23] J. Sivic, A. Zisserman, Video Google: A text retrieval approach to object matching in videos, in: IEEE International Conference on Computer Vision, Vol. 2, 2003, pp. 1470–1477.

[24] D. G. Lowe, Distinctive image features from scale-invariant keypoints, International Journal of Computer Vision 60 (2) (2004) 91–110.

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

[26] J. Matas, O. Chum, M. Urban, T. Pajdla, Robust wide-baseline stereo from maximally stable extremal regions, Image and vision computing 22 (10) (2004) 761–767.

[27] M. Muja, D. G. Lowe, Fast approximate nearest neighbors with automatic algorithm configuration, in: International Conference on Computer Vision Theory and Application (VISSAPP), 2009, pp. 331–340.

[28] I. Gordon, D. G. Lowe, What and where: 3D object recognition with accurate pose, in: Toward category-level object recognition, Springer, 2006, pp. 67–82.

[29] T. Sattler, B. Leibe, L. Kobbelt, Fast image-based localization using direct 2d-to-3d matching, in: IEEE International Conference on Computer Vision (ICCV), 2011, pp. 667 –674.

[30] E. Hsiao, A. Collet, M. Hebert, Making specific features less discriminative to improve point-based 3D object recognition, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2010, pp. 2653–2660.

[31] O. Miksik, K. Mikolajczyk, Evaluation of local detectors and descriptors for fast feature matching, in: 21st International Conference on Pattern Recognition (ICPR), 2012, pp. 2681–2684.

[32] R. Kummerle, G. Grisetti, H. Strasdat, K. Konolige, W. Burgard, G2o: A general framework for graph optimization, in: IEEE International Conference on Robotics and Automation (ICRA), 2011, pp. 3607–3613.

[33] F. Devernay, O. Faugeras, Straight lines have to be straight, Machine Vision and Applications 13 (1) (2001) 14–24.

[34] R. I. Hartley, A. Zisserman, Multiple View Geometry in Computer Vision, Cambridge University Press, ISBN: 0521540518, 2004.

[35] N. Snavely, S. Seitz, R. Szeliski, Photo tourism: exploring photo collections in 3D, in: ACM SIGGRAPH 2006 Papers, 2006, pp. 835–846.

[36] Y. Furukawa, J. Ponce, Accurate, dense, and robust multi-view stereopsis, IEEE Trans. on Pattern Analysis and Machine Intelligence 32 (8) (2010) 1362–1376.

[37] G. Griffin, A. Holub, P. Perona, Caltech-256 object category dataset, Tech. Rep. 7694, California Institute of Technology (2007).
URL http://authors.library.caltech.edu/7694

[38] K. E. A. Van de Sande, J. R. R. Uijlings, T. Gevers, A. W. M. Smeulders, Segmentation as selective search for object recognition, in: IEEE International Conference on Computer Vision (ICCV), 2011, pp. 1879–1886.

[39] C. H. Lampert, M. Blaschko, T. Hofmann, Beyond sliding windows: Object localization by efficient subwindow search, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2008, pp. 1–8.

[40] A. Vedaldi, S. Soatto, Quick shift and kernel methods for mode seeking, in: Proceedings of the European Conference on Computer Vision (ECCV), 2008.

[41] S. Kullback, R. A. Leibler, On information and sufficiency, The Annals of Mathematical Statistics (1951) 79–86.

[42] O. Chum, J. Matas, Matching with PROSAC-progressive sample consensus, in: IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), Vol. 1, 2005, pp. 220–226.

[43] P. Torr, A. Zisserman, MLESAC: A New Robust Estimator with Application to Estimating Image Geometry, Computer Vision and Image Understanding 78 (1) (2000) 138 – 156.

[44] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Y. Ng, ROS: an open-source Robot Operating System, in: Workshop on open source software. IEEE International Conference on Robotics and Automation (ICRA), Vol. 3, 2009.

[45] L. Dagum, R. Menon, OpenMP: an industry standard API for shared-memory programming, IEEE Computational Science Engineering 5 (1) (1998) 46–55.

[46] J. Sturm, N. Engelhard, F. Endres, W. Burgard, D. Cremers, A Benchmark for the Evaluation of RGB-D SLAM Systems, in: Proc. of the International Conference on Intelligent Robot Systems (IROS), 2012.
URL http://vision.in.tum.de/data/datasets/rgbd-dataset

[47] J. Sivic, A. Zisserman, Efficient visual search of videos cast as text retrieval, IEEE Transactions on Pattern Analysis and Machine Intelligence 31 (4) (2009) 591 –606.

[48] B. K. P. Horn, Closed-form solution of absolute orientation using unit quaternions, Journal of the Optical Society of America A 4 (4) (1987) 629–642.

[49] M. Waibel, M. Beetz, J. Civera, R. D'Andrea, J. Elfring, D. Gálvez-López, K. Haussermann, R. Janssen, J. Montiel, A. Perzylo, B. Schiessle, M. Tenorth, O. Zweigle, R. van de Molengraft, Roboearth, IEEE Robotics Automation Magazine 18 (2) (2011) 69–82.

View publication stats