Surveillance based on Visual Objects Recognition

Filipe Fialho, José António Gaspar
Instituto Superior Técnico / UTL, Lisbon, Portugal
filipefialho@tecnico.ulisboa.pt, jag@isr.tecnico.ulisboa.pt

ABSTRACT

The firing of an alarm in a home environment raises the question "what happened?". We propose surveillance based on visual objects recognition, applicable to a mobile robot that, navigating through the environment and comparing it with a previous stored navigation, finds scene changes, namely missing, new and modified objects and, thus, to answer the raised question.

We combine a neural network based object detector with a Simultaneous Localization and Mapping system (SLAM). In this way, the information about the object class and 2D bounding boxes is complemented with a collection of 3D points associated to each object. In addition our approach applies the concept of Object Permanence [18] and takes into account the different visibility states of the objects as they can be inside or outside the camera field of view (FOV).

Different experiments are performed in a real home scene with objects of interest, in order to validate the proposed solution in different contexts. Promising results are achieved as our proposed solution is able to find missing, new and modified objects. Our system is robust against homonymous objects (same class) and against objects that become visible and not visible across the time, according to the motion of the camera.

I. INTRODUCTION

Surveillance consists in the monitoring of behaviors, activities or information for public and domestic safety purposes. One of the main forms of surveillance is through video surveillance systems.

A sentinel mobile robot is defined in this thesis as an autonomous robot capable of moving around and do video surveillance. The robot is equipped with a camera and a self-localization system. The robot is also equipped with data processing interpreting information from the surrounding environment.

One main objective of this thesis is to propose a solution to a problem that a sentinel mobile robot may face in its activity: given an environment with objects of interest, like a home, an alarm that goes off raises the question "what happened?" in the scene. To find out, a sentinel mobile robot navigates through the house registering the objects present. The comparison with a previously stored registration of objects, before the firing of the alarm, gives the answer to the initial question, i.e., allows to find the changes in the scene.

In this work we consider video data processing for objects detection and naming as a starting point. More in detail, we consider the state-of-the-art real-time object detection system

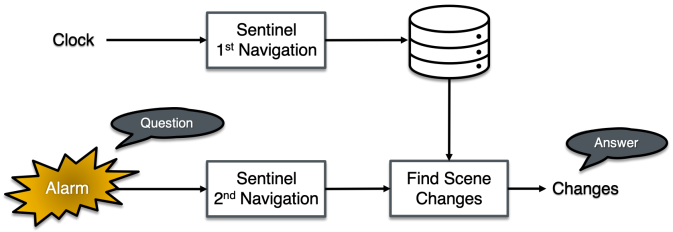


Fig. 1: **Basic Diagram of the proposed solution**. Given the trigger of an alarm, the sentinel robot navigates through an environment searching for scene changes in respect to a previously navigation stored on hard drive.

YOLOv4 [2]. It is a system based on neural networks, which provides a list of detected objects in a video frame. Being an image frame based system, some challenges remain, as how to assign constant object-names (identifiers) to objects that are constantly present in the scenario but sometimes are not detected, or handling spurious false detections or misnaming. In the end, we propose matching object detections on different video sequences, as in figure 2, in order to find scene changes.

A. Surveillance, Objects Detection and Re-identification

As referred, the focus of this work is the vision based surveillance of an indoor environment with objects of interest. The objects are detected and listed in a scene learning navigation, and later re-identified, in a surveillance navigation, so that the changes in the scene can be found.

Figure 2 shows state of the art object detection and identification with YOLOv4 [2]. One observes some objects are constantly being detected and not detected, while effectively the objects are within the field of view all the time. For example, the detection of the *cup* in the plot (b) has an unstable nature. In another extreme, a non-existing object, a *chair*, is found in both plots (a, b). These non-ideal responses are expected. However, one arguably considers object detection and naming a starting point of a surveillance work because it helps communicating with humans. One desires a surveillance robot that refers to objects with meaningful names.

One additional challenge of doing visual surveillance based on objects re-identitication is the typical asynchronism between navigations. The asynchronism of navigations is clearly visible in the case of the *cup*. In the first plot the *cup* is visible in the first half of the time sequence, while in the second plot the *cup* is only visible in the second half of the time sequence. This asynchronism, together with the referred detections unstable nature, are challenging for comparing navigations and, consequently, in finding scene changes.

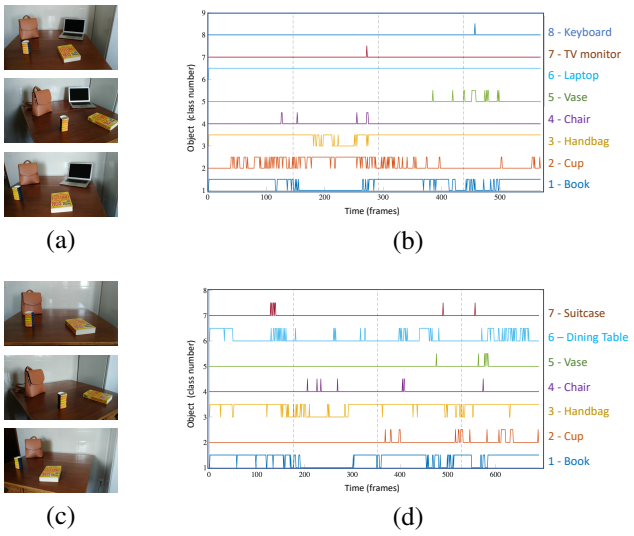


Fig. 2: **Missing laptop in the second navigation**. (a) First navigation objects: book, cup, handbag and laptop. Shown frames 10, 200 and 400 of 571. (b) Four objects correctly detected, and four false detections: chair, vase, tvmonitor and keyboard. (c) Second navigation objects: book, cup and handbag. Shown frames 10, 250, 550 of 694. (d) Three objects correctly detected, and three false detections: chair, vase and suitcase. Equal object classes are represented using the same color in both plots.

We propose approaching these problems by processing the data extracted by the object detector, as well as the use of an additional concept, complementary to objects detections only, that allows to make correct inferences about the permanence of the objects along time. The unstable detection nature may be mitigated using a low pass filter based approach that attenuates the fast appearance and disappearance changes in object detections. The asynchronism can be handled by matching the objects in different navigations, as well as tracking the objects along them. To do this, it is useful to know the 3D position of the objects, once this information about the objects is more stable and invariant to changes, as in Armeni et al. [1].

B. Problem Formulation and Approach

A monocular vision based surveillance mobile robot is desired to find changes occurred with objects of interest in a scene. Object detection methodologies are interesting, as they provide labeling readable by humans, however their real-time single-frame base naturally limits obtaining detected-objects-permanence proprieties.

Some works, also focused in detecting the changes or the dynamic elements in a scene or in re-localizing the objects in changing indoor environments [13], [6], [21], have the drawbacks of using expensive RGB-D cameras or 3D object models and of not being real-time systems. Our goal is to propose a potentially real-time system that uses a single pinhole camera, like a standard smartphone camera, as it is low cost and massively used nowadays. Thus, such a system could be used in any home.

Our approach consists of complementing the object identification information given by a neural network based object detector with 3D space information given by a Simultaneous Localization and Mapping (SLAM) system, so that the aforementioned limitations of the object detector could be surpassed. Additionally, a notion of Object Permanence [18] is also applied, so that the fast changes in object detections could be attenuated based on a low pass filter approach.

In summary, we propose a find scene changes system that, given an actual navigation through an environment and a previous stored navigation of the same environment, is able to find the changes in the scene objects, namely missing, modified and new objects (figure 1).

The rest of the paper is structured as follows. Section 2 presents the state-of-the-art methods used, while Section 3 describes the methodology developed and implemented in the process to find changes in scene objects. Section 4 provides an overview of the experiments executed as well as the results attained. Finally, Section 5 highlights the main achievements obtained and proposes some points of further work.

II. Background and State of the Art

A. Scene Representations

State-of-the-art methods use scene graphs to represent environments, as they compactly store information that describes a scene and are robust enough to deal with noise and small changes.

Kim et al. [9] propose a 3D scene graph model that enhances the autonomy of intelligent agents like robots. It allows robots to collect accurate information, to complete diverse tasks easily and to expand the knowledge of the environments incrementally.

Armeni et al. [1] introduce a unified structure that hosts multiple types of semantics. Given a 3D mesh and registered panoramic images, they construct a 3D scene graph that covers an entire building including semantics on objects, rooms and cameras, as well as the relationships between them.

Rosinol et al. [16] take a step forward presenting 3D dynamic scene graphs that also represent moving agents, like humans and robots, and include spatio-temporal relations and topology at different levels of abstraction. This actionable information supports tasks like planning and decision-making.

More recently, Rosinol et al. [17] attempts to reduce the gap between robot and human perception by bringing some maturity to previous works [16].

a) Scene Graphs: A graph is a mathematical structure constituted by nodes and edges. When we look to a space like a room in a building, we automatically detect and recognize elements and also establish relationships among them. This room can, thus, be represented by a graph where nodes are the elements and edges are the relationships between them.

b) Elements: Armeni et al. [1] introduce the concept of a four-layered 3D Scene Graph where each layer has a set of nodes and each node has a set of attributes that describe its properties. Each layer represents a different level of abstraction of the environment.

The 3D Dynamic Scene Graph proposed by Rosinol et al. [16] adds a collection of time-stamped 3D poses to describe the trajectory of moving agents over time.

Wald et al. [22] introduce the 3D Semantic Scene Graph where nodes are defined as instances and label them by hierarchical classes. Attributes are divided into three types: static properties, dynamic properties and affordances.

c) Relationships: In the graph introduced by Armeni et al. [1], since there are nodes in all layers, edges may link nodes either from the same layer or from different layers.

The same happens in the graph presented by Rosinol et al. [16], but here edges represent pairwise spatio-temporal relations (e.g., "agent A is in room B at time t").

Wald et al. [22] divide relationships in three types: support, proximity and comparative relationships.

d) Applications: A scene graph can be used in several tasks in the robotics field. Kim et al. [9] perform some application demonstrations like visual question and answering (VQA) and task planning. Wald et al. [22] introduce a task that consists in identifying the 3D scene from a list of scans given a single 2D image with potential changes. Rosinol et al. [16] reflects on how a 3D dynamic scene graph can be used to find an object by directly infer the closest place in the graph it has to reach and plan a feasible path to that place.

B. Monocular Vision based Navigation

Simultaneous Localization and Mapping systems (SLAM) are well defined in the robotics community as the question of moving sensors building a representation of its environment on the fly while concurrently estimating its ego-motion [5].

SLAM systems differ from Visual Odometry (VO) and Structure from Motion (SfM) systems as the first ones focus on computing the agent's ego-motion only and not on building a map [3], while the second ones are essentially offline in nature and need to analyze an entire image sequence [5].

Davison et el. [5] introduce a real-time algorithm, MonoSLAM, which can recover the 3D trajectory of a monocular camera based on the online creation of a sparse but persistent map of high quality features.

Civera et al. [4] present a novel combination of Random Sample Consensus (RANSAC) and Extended Kalman Filter (EKF) which uses the available prior probabilistic information from the EKF.

More recently, Campos et al. [3] propose ORB-SLAM3, the first system able to perform visual, visual-inertial and multi-map SLAM with monocular, stereo and RGB-D cameras, using pin-hole and fisheye lens models.

C. Real Time Object Detection

One of the main topics in computer vision, nowadays, is object detection and is growing rapidly.

a) Neural Network Based Detector, YOLO: You Only Look Once (YOLO) is a family of object detection models that has been dominating this field due to its increasing speed and accuracy. Unlike other detectors that are two-stage detectors, this one predicts object bounding boxes and object classes in a single forward pass through the network, hence the name.

YOLO was firstly introduced in 2016 by Redmon et al. [15] and it was a step forward in the history of object detection models due to its capability of real-time detection with a better accuracy. After this first version, several newer versions of YOLO have been released with increasing accuracy and speed.

b) General Architecture: YOLO is an one-stage object detector which means that it makes the predictions in a single forward pass through the network, whose general architecture is composed by the backbone, neck and head modules.

c) Backbone: In the backbone, the features are extracted from the input image. Models commonly used here can be ResNet [7], DenseNet [8] or VGG [19].

d) Neck: The layers in the neck module make the connection between the backbone and the head. They are responsible for extracting different feature maps of different stages of the backbone to obtain features with different scale. FPN [10], PANet [12] and Bi-FPN [20] are, usually, used in the neck part.

e) Head: It is responsible for doing the detection of bounding boxes. The final output includes the information of the predicted bounding box, the object score and the probability of each object class.

D. Object Permanence

Object permanence is the ability of understanding that objects continue to exist even when they are not seen or sensed. This concept is extremely important to effectively modeling dynamic scenes as, in reality, objects naturally and dynamically occlude and contain each other [18].

A fundamental concept in the field of developmental psychology, it was firstly introduced by Piaget J. that argued that this is one of the most important milestones of the development of a child, as, without it, objects would not exist permanently and separately [14].

In computer vision field, object tracking is a widely studied topic. As for the child, reasoning about the location of an object that is not visible is also much more harder for a vision-based agent. Accordingly to Shamsian et al., localizing a target object in a video scene involves four different tasks, based on the state of the target object. It depends if the object is visible, occluded, contained or carried [18].

The first case is when the object is visible and is, naturally, the simplest task. Nowadays, every common object detector is able to accomplish it. The second case is when the object is temporally occluded by another moving agent (e.g., a person occluded by a moving car in the street). In long-term occlusions, this task can be very challenging.

The third case is when the object becomes not visible because it is contained inside another object (e.g., an object that is put inside a static bag). Finally, the fourth case is when the object becomes not visible because is located inside another moving object (e.g., an object carried inside a bag by a running thief).

III. Detection of Changes on Scene-Objects

In this section, it is explained how the sentinel mobile robot is able to find scene changes. Firstly, how it localizes and

represents the objects. Then, it is described how the objects and their detections are associated along the navigations and between navigations. Finally, it is exemplified how the catalog of the scene with the objects present in each navigation is built. Comparing these catalogs, the sentinel mobile robot is able to find the changes in scene objects.

A. Objects Representation, Localization and Permanence

Object localization has been a major topic of research, in the last years, specially with the recent improvements in visual and visual-inertial SLAM systems, like ORB-SLAM3 [3]. SLAM systems robustly localize and map objects. The recent advances in deep learning have also contributed to this task, once the use of CNNs to estimate depth from a single image has also shown promising improvements.

Localizing or detecting objects when they are partially or fully occluded or when they are simply outside the camera field of view (FOV), is challenging.

We approach the partial visibility problem using the object permanence concept described in section II-D. The permanence concept endows the system with the ability to understand that the objects continue to exist even when they are not seen by the camera.

This work considers three possible states for the observation of an object: object visible and detected; object visible and not detected; and object not visible. A visible object is inside the camera FOV, while a detected object is identified by an object detector. The state machine represented in figure 3 shows the different states that an object can take during a navigation.

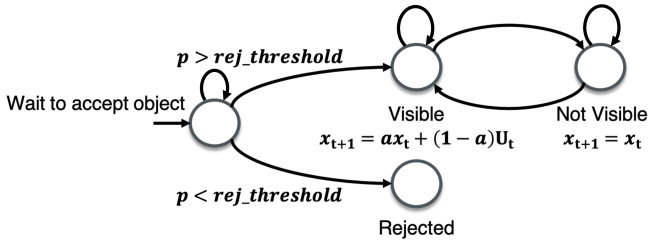


Fig. 3: **Objects State Machine**. The detected objects can be accepted or rejected. Along the navigation, the objects may be inside or outside the camera FOV and, thus, visible or not visible. The permanence of an object is updated accordingly.

The detected objects can become visible or not visible to the camera along the navigation. In each of these states, the object permanence value is updated according to the associated equation present in figure 3. Considering the object permanence across the time, the objects can be accepted or rejected.

When an object is visible and detected by the object detector, its permanence increases and its existence is reinforced. When an object is visible but not detected, i.e. is in the camera FOV but not detected by YOLO, its permanence decreases and its existence is weakened. When an object is out of the camera FOV, since one cannot say that the object does not exist, especially if it was already detected, its permanence is kept constant.

This thesis also considers the representation of objects and their relationships in indoor environments. To be able to find changes in scene objects, is essential that a sentinel mobile robot can internally represent the objects present in the scene as well as the spatial and semantic relationships between them and with the environment itself.

Two main data structures are used, one for the camera and one for the objects. The camera is represented by the projection matrix, with constant intrinsic parameters K:

$$C_t = K[R_t \ t_t] \tag{1}$$

where R_t and t_t denote the camera pose along time. The camera pose is estimated with a SLAM system, in each frame. The size of the image is included in the camera structure, in order to allow representing the 3D volume (pyramid) corresponding to the field of view of the camera.

Similarly to the state-of-the-art works [1], [9], [16], [22] that use graphs to represent a scene, the proposed objects data structure $\mathsf{G} = (N, \ E)$ is a graph where the nodes N correspond to the objects in the scene and the edges E represent inter-objects relationships.

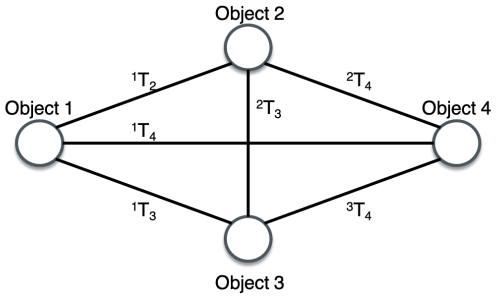


Fig. 4: **Representation of the Objects Graph G**. The objects data structure is a graph constituted by nodes and edges. The nodes represent the objects in the scene and the edges the translation $^j\mathsf{T}_i$ between object j and object i.

More precisely, the nodes in the graph G contain the following information about the objects: numerical identifier (ID), label (e.g., "book"), centroid and permanence. The way each object gets an unique ID is explained in section III-B. The label is given by an object detector and the centroid results from computing the mean of the space point features associated to each object, given by a SLAM system. Objects permanence is updated based on filtering the objects re-identification (binary flag) along time. The information in a node is referred in the following as the state of the object.

Edges, as referred before, represent inter-objects relationships. In particular, edges represent visibility properties. For instance, two objects can be imaged side by side, or one may be occluding the other. May also happen not being possible to see both objects in the same image, e.g. objects hanged in opposite faces of a long wall. A simple indicator of the visibility case is the translation vector between the centroids of the objects j and i, $^j\mathsf{T}_i = \mathsf{t}_j - \mathsf{t}_i = \left[\Delta x_{ji}, \ \Delta y_{ji}, \ \Delta z_{ji}\right]^\mathsf{T}$. If the optical axis of the camera is parallel to $^j\mathsf{T}_i$, then the objects are likely to be occluding each other. In case of orthogonality,

the likely case is a view of the objects side-by-side. When the objects cannot be seen side by side, the graph edge and the vector ${}^{j}\mathsf{T}_{i}$ are left undefined.

B. Objects Association and Tracking

Object tracking is the task of localizing objects over time by taking an initial set of object detections, assign the respective IDs and then track those objects as they move along a video, maintaining the assigned IDs. It is still a necessary task on the analysis of video sequences as YOLO [15] just detects objects in images.

To be able to find the changes in the scene objects, our system needs to localize and track the objects present along the navigations. Our system is based on MonoSLAM [4], that gives space point features detected in each frame of the video navigation, and the object detector YOLO [2], that gives object class labels also in each frame of the video navigation. We propose associating object class labels in each frame with the features found on the frames of the navigations.

This objects association is based on the distance between object centroids and occurs as follows. In each frame, the space point features are associated to the detected objects. Then, the objects detected in the current frame are matched with existing objects, receiving their previously assigned ID, or imply creating new objects, receiving new IDs. In other words, the objects state and the data structures C and G are updated, as it is represented in figure 5, where t denotes the current frame.

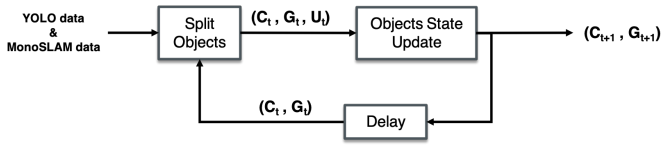


Fig. 5: **Blocks Diagram of Objects Association Along a Navigation**. In each frame t, the space point features are associated to the detected objects. Then these objects are split and their state updated. Finally, the data structures C_t and G_t are updated. U_t is the object detection flag.

a) Association of Features to Objects: YOLO [2] and MonoSLAM [4] provide complementary data: the former gives information about detected objects, as bounding boxes and class labels, and the latter gives information about the localization of objects, as image point features and the correspondent space point features mapped on a 3D scene map.

The association between object labels and space point features starts by associating the objects bounding boxes and the image point features. Frequently, two or more bounding boxes overlap and there are image point features whose position is in the area shared between the two bounding boxes.

Consequently, one needs to associate the common point features to the correct bounding box, i.e. to the correct object. We start by assuming that the image point features extracted by MonoSLAM [4], FAST corners detected in scene frames, whenever inside the bounding box of an object detected by YOLO, belong to that object.

In the case of point features inside intersections of multiple bounding boxes, it is performed a k-nearest neighbors search, based on the Euclidean distance, between each common point feature and each set of exclusive point features. In this way, each common point feature is associated with the closest object.

b) Split Objects: Once the point features are associated to the detected objects, these objects are split into existing objects, receiving the previously assigned IDs, or new objects, receiving new IDs. This allows tracking different objects, possibly of equal classes, along an entire navigation.

The attribution of the object IDs is based on the distance between object centroids and occurs as follows. Each navigation is explored from the beginning to the end, frame by frame. In a given frame t, each object centroid is associated with the closest object centroid found with the same label in the previous frames of the navigation, receiving its assigned ID. If the closest object centroid found is not close enough, i.e., is at a distance $d_k > threshold$, it is considered that the object of current frame t is new receiving a new ID.

In the end, it could be possible that some associations made are not correct and some IDs are badly attributed. This could be due to the objects being too close from each other and to some noise that can be present in the data given by MonoSLAM. These problems show the need to exist the concept of object state.

c) Objects State Update: The state of an object is updated according to if the object is visible or not visible. Once we pretend to build a representation per navigation of each object in the scene, the update of the state takes into account the values of the state in the previous frames.

The new permanence value of an object that is visible, i.e., inside the camera FOV, is updated according to

$$x_{\mathsf{t}+1} = a x_{\mathsf{t}} + (1-a)\mathsf{U}_\mathsf{t} \tag{2}$$

where x_t is the permanence value in frame t, $a \leq 1$ is the update coefficient and U_t is the detection flag that indicates if the object is detected or not by the object detector in frame t.

So, when an object is being constantly detected, its permanence is constantly growing, until a maximum value of 1. When an object starts not being detected, its permanence starts decreasing until a minimum value of 0.

The value of a defines how fast the permanence varies, once this is a linear and time invariant system and a is its only pole. Greater the a, faster is the system response, less is the settling time and, so, less time the system needs to accept new objects. In fact, this approach works as a simple lowpass filter, smoothing fast changes in object detections.

The new permanence value of an object that is not visible, i.e., is outside of the camera FOV, is updated according to

$$x_{\mathsf{t}+1} = x_{\mathsf{t}} \tag{3}$$

where x_t is the permanence value in frame t. So, in this case the permanence value is kept constant, just as aforementioned.

The permanence of objects along the navigation is essential to accept or reject those objects. If the permanence of an object goes under a rejection threshold, it is considered that

this detection is not consistent enough to be associated to a real object and, thus, it is rejected. If the object is not rejected and its permanence value overcome an acceptance threshold, it is accepted and it will be part of the catalog of the scene.

Besides the permanence, also the centroid of each object gets updated along the navigation, so that in the end each object can be related to one 3D position.

The new value of each centroid in each frame is given by

$$\mathbf{x}_{t+1} = a\mathbf{x}_t + (1-a)\mathbf{c} \tag{4}$$

where $\mathbf{x}_t$ is the 3×1 vector containing the coordinates of the global object centroid in frame t, a is the update coefficient and $\mathbf{c}$ is the 3×1 vector containing the coordinates of the local object centroid. Here, the global object centroid is the one that has been updated along the navigation, according to equation (4), and the local object centroid is the mean of the object point cloud of current frame t.

Besides the state update of existing objects, also the new objects are added to the objects graph G. If an object detection has a new ID, a new node is created in graph G with the respective information of the object, as well as the transformations between the new object and the others.

C. Navigations Matching

To successfully compare the environment before and after the changes are made, the two correspondent navigations have to be matched, as they are independent and, thus, not comparable.

This matching consists in identifying and associating the objects that are present in the scene in both navigations. Matched objects receive the previously assigned IDs in the first navigation, while new objects receive new IDs.

This process can be divided into 2 parts. In the first part, groups of centroids with the same label are created. Within these groups, the centroids are separated according to the navigation they come from. Thus, are created two groups of centroids for each label, group 1 and group 2.

In the second part, centroids of group 1 and group 2 of each label group are associated through k-nearest neighbors search. This ensures that the objects that are closest are associated. If the closest centroids are not close enough, i.e., if the distance $D_k > threshold$, they are not associated.

D. Scene Catalog

Objects are accepted or rejected based on the evolution of the respective permanence along the navigation. The objects that are accepted are part of the scene catalog.

The plot of figure 6 represents the permanence of a group of detected objects across a navigation. In this example, the detected objects are: book, laptop, cup, handbag, chair and vase. The acceptance threshold is also represented as a dashed line. In this simple case, there is no rejection threshold.

The objects whose permanence overcome the acceptance threshold and, therefore, enter the scene catalog are the book, the laptop, the cup and the handbag. The chair and the vase are rejected, since their permanence do not reach the same threshold.

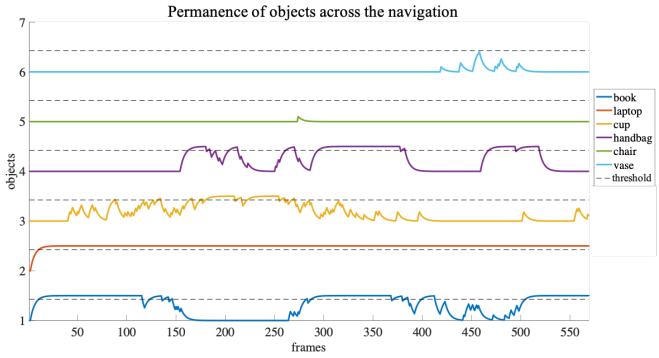


Fig. 6: **Permanence of detected objects across a navigation**. The objects detected along the frames are identified by different colors and IDs. The dashed line represents the acceptance threshold whose value is $acc = 0.85$. The objects whose permanence overcome this threshold are the book, the laptop, the cup and the handbag.

While the permanence curves of the accepted objects reach, in general, higher values and during much more significant periods of time, the permanence curves of the rejected objects only have some isolated peaks that do not grow sufficiently.

From the permanence plot like the one in figure 6 results a scene catalog that contains the ID and the label of each accepted object. In this case, the objects that are accepted are exactly the ones which are in the scene.

Remembering the initial plots of objects detected along a navigation, in figure 2, one can verify the advantages of this new approach. Since the object detections are much more smoothed, the mentioned instability is decreased, which allows to have a much better notion of the objects existence along the navigations. Besides that, it rejects objects whose detections are not sufficiently robust.

Sometimes, the object detectors misidentify some objects with similar aspect to another known ones. These incorrect detections occur, commonly, during very short periods of time. With this approach, the majority of these incorrect detections are overtaken as the system discard them.

Finally, once the scene catalogs are obtained, the new and missing objects of a scene are found by comparison of these catalogs. A lost object is an object that was in the scene in the first navigation, but not anymore in the second one. A new object is an object found in the second navigation, that was not present in the scene in the first moment.

E. Summary of the Proposed Method

As mentioned earlier, the goal of this work is to endow a moving agent with the capability to find changes in scene objects. For example, a sentinel mobile robot, equipped with a camera, navigates through a house looking for new and missing objects after the alarm has been fired.

The main steps of the proposed Find Scene Changes system are represented in the diagram of figure 7. This system receives as inputs two sentinel navigations, one before and one after the firing of the alarm, and returns the changes in the scene.

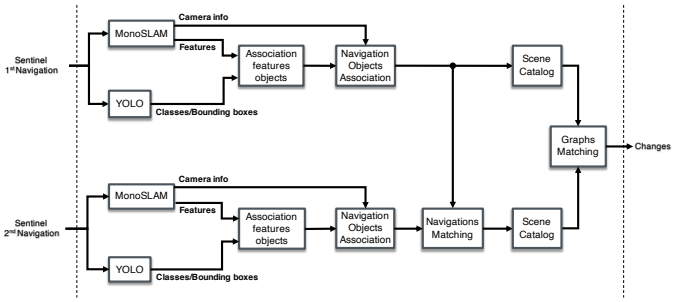


Fig. 7: **Diagram of the proposed Find Scene Changes system**. Given two Sentinel Robot navigations, scene features and object labels are associated. Then, objects are associated along each navigation and both navigations are matched. Finally, is built a catalog of the scene for each navigation, which allows to find the changes in the scene.

Each one of the navigations goes by MonoSLAM and YOLO. The first one obtains the 3D position of scene features detected in the navigations, as well as information about the camera. The second one detects the objects, obtaining its classes and associated bounding boxes.

Then, these point features are associated to the objects, through its classes and bounding boxes. With this association and with the information about the camera, the next step is to associate, along the entire navigation, the objects detected in each frame. Once these two navigations are independent of each other, they have to be matched. So, the objects present in both navigations are associated.

In this way the sentinel robot is able to build a catalog of the scene, for each navigation, with the objects present and the respective IDs. Finally, matching both catalogs, he is able to find what changed in the scene objects.

IV. Experiments and Results

In this section are introduced three experiments performed with Find Scene Changes system and one, in particularly, is described in detail. In these experiments are used two different setups: one where all the objects have static textures and one where some objects have varying textures.

A. Experimental Setups and Implementation Details

Although similar in the objects and environment used, the two setups used try to evidence the different textures that the objects can have. In the first setup the objects are completely static, while in the second setup some objects have varying textures that change over time.

We tried to use objects that met the following conditions: i) be objects that are found in any common home, to simulate a home environment in the best way; ii) be objects whose labels are present in the dataset used to generate pre-trained weights used by the object detector, so that the objects are easily detectable; iii) be objects with some height and with protruding corners, so that MonoSLAM can be able to detect FAST corners and extract the associated features.

a) Setup I - Static Textures: In this first experimental setup, a set of objects on top of a table is used to simulate a home environment. The objects can be found in any common home and they are the following: two different books, one laptop, one cup and one handbag. Some of these objects are manually removed or added from one navigation to the other to produce changes in the scene. As the table where the objects are on top is detectable by the object detector, it is manually ignored by the system. Beyond not being relevant to the context, its detection also would overlap and affect the detection of other objects.

To produce three different sentinel robot navigations, the setup has three variations, as represented in figure 8. Each one of the variations corresponds, naturally, to each one of the navigations. The differences between them are simple removals or additions of objects. With these three navigations are performed the experiments 1 and 2.

(a) Setup I, navigation 1

(b) Setup I, navigation 2

(c) Setup I, navigation 3

Fig. 8: **Setup I**. (a) In navigation 1 exists one book, one laptop, one cup and one handbag. (b) In navigation 2 exists one book, one cup and one handbag. (c) In navigation 3 exists two different books, one laptop and one handbag.

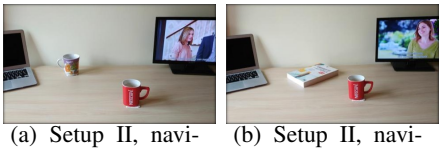
b) Setup II - Varying Textures: In this second experimental setup, a set of objects on top of a table is also used to simulate a home environment. The objects can be found in any common home and they are the following: two different cups, one laptop, one book and one tv monitor. The tv monitor is playing a generic movie and, therefore, different images are showed across time, giving, thus, the varying textures of this experimental setup. Some of these objects are manually removed or added from one navigation to the other to produce changes in the scene.

As the table where the objects are on top and the persons that appear in the tv monitor are highly detectable by the object detector, they are manually ignored by the system. Beyond not being relevant to the context, their detection also would overlap and affect the detection of other objects.

To produce two different sentinel robot navigations, the setup has two variations, as represented in figure 9. Each one of the variations corresponds, naturally, to each one of the navigations. The differences between them are the replacement of objects and the different images showed on the tv monitor. With these two navigations is performed the experiment 3.

c) Implementation Details: The sentinel robot navigations are simulated by an human recording the scene with a camera of an average smartphone available in the market. It is also used a MacBook Air from 2015 with a 1,6 GHz Intel Core i5 CPU and a Intel HD Graphics 6000 1536 MB graphics card.

Regarding the software issues, the object detector used is Yolov4 [2], running on Google Colaboratory, that allows to



(a) Setup II, navigation 1 (b) Setup II, navigation 2

Fig. 9: **Setup II**. (a) In navigation 1 exists one tv monitor, one laptop and two different cups. (b) In navigation 2 exists one tv monitor, one laptop, one cup and one book.

use an external GPU. The pre-trained weights used in object detections were generated with the MS COCO dataset [11]. To obtain a map of the scene, the localization of the camera and the associated detected features is used MonoSLAM [4], running on MATLAB R2017b. Finally, all the algorithms used in the task of detecting changes in the scene are also developed in the MATLAB programming language.

The values used, for the thresholds and coefficients referred along the work, are presented in table I for the experiment, here, presented in detail.

Name	Notation	Description	Value
Distance threshold	d_k	Maximum distance between associated centroids	1
Update coefficient	a	Coefficient used in the permanence update equation (2)	0.95
Acceptance threshold	acc	Minimum value of permanence to accept an object	0.85
Rejection threshold	rej	Maximum value of permanence to reject an object	0.01
KNN distance threshold	D_k	Maximum distance between associated objects using k-nearest neighbors search	1

TABLE I: **Thresholds and coefficients used**. The distances values are in unit meters.

d) Experiments List: From the combination of the various sentinel robot navigations, were performed three different experiments. In the first experiment, it is evidenced the missing of an object. In the second experiment, it is evidenced the replacement of objects and the existence of homonymous objects. Finally, in the third experiment, it is evidenced the case in that objects become visible and not visible across time. The latter is the experiment, here, presented in detail, as it is the most complex one.

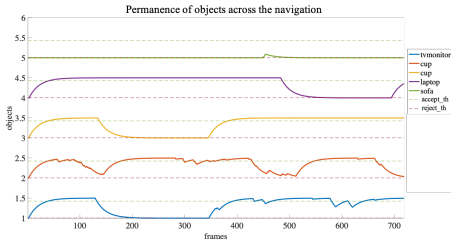
Firstly, it is shown how would be the permanence evolution of the objects if the system would not be able to know if they are inside or outside the camera FOV. Secondly, this same evolution is shown when the system is able to recognize that the objects are inside or outside the camera FOV.

Thirdly, it is built a representation of the scene in each navigation through a graph structure. Finally, after the matching of the navigations, are shown the detected scene changes and the objects catalog built by the sentinel robot, where it is possible to retrieve the lost and new objects found.

B. Objects In and Out of Camera FOV

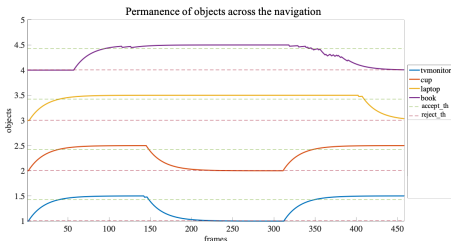
The pair of navigations used in this experiment is navigation 1 and navigation 2 of setup II. In this case, besides existing

(a) Navigation 4, frames 10, 160, 550.



(b) Permanence of detected objects across the first navigation.

(c) Navigation 3, frames 10, 200, 350.



(d) Permanence of detected objects across the second navigation.

Fig. 10: **FOV cropping not considered**. (a) Objects present in the first navigation: laptop, two cups and tv monitor. (b) Accepted objects in the first navigation: cup. (c) Objects present in the second navigation: laptop, book, cup and tv monitor. (d) Accepted objects in the second navigation: laptop and book. In (b) and (d), equal object classes have the same color.

an object replacement and homonymous objects, there is also objects that enter and exit the camera field of view (FOV) and, thus, are visible and not visible along the navigations.

There are two different cups in the first navigation and one of these cups is replaced by a new book in the second navigation. All the objects in the scene are not visible during some period of the navigation, except the violet cup in first navigation and the book in the second one.

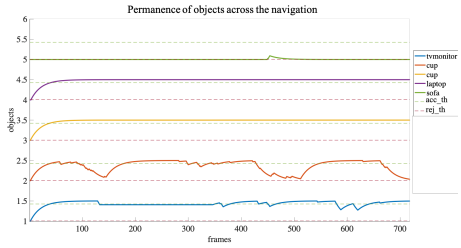
In order to show what would be the behavior of our system if it would not take into account the different states of visibility of the objects, i.e., objects that enter and exit the camera FOV, figure 10 shows the objects permanence along both navigations, together with a representative sequence of frames where it is possible to see the objects present in the scene.

In this case, the sentinel robot would only find a cup in the first navigation and a laptop and a book in the second navigation. One can conclude that the changes in scene objects found would not be the correct ones.

Figure 11 shows the evolution of the objects permanence along the same two navigations but, now, resultant of our proposed approach.

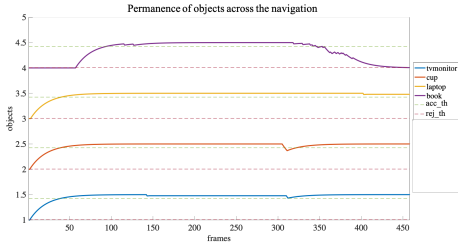
In the first navigation, from the five detected objects, the permanence of the object sofa does not even reach the acceptance threshold, while the remaining four objects after reaching the acceptance threshold do not go under the rejection threshold. These four objects are the ones that are present in

(a) Navigation 4, frames 10, 160, 550.



(b) Permanence of detected objects across the first navigation.

(c) Navigation 3, frames 10, 200, 350.



(d) Permanence of detected objects across the second navigation.

Fig. 11: **Objects in and out of camera FOV**. (a) Objects present in the first navigation: laptop, two cups and tv monitor. (b) Accepted objects in the first navigation: laptop, two cups and tv monitor. (c) Objects present in the second navigation: laptop, book, cup and tv monitor. (d) Accepted objects in the second navigation: laptop, book, cup and tv monitor. In (b) and (d), equal object classes have the same color.

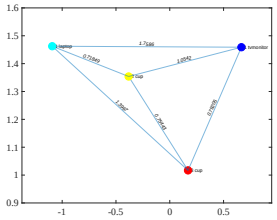
the frames of figure 11a and, thus, we can conclude that this permanence-based model of detection filtering works well in this navigation, as the objects are correctly accepted.

In the second navigation, the permanence of all the detected objects overcomes the value of the acceptance threshold and, after that, do not go under the rejection threshold. These four objects are the ones that are present in the frames of figure 11c and, thus, we can conclude that this permanence-based model of detection filtering also works well in this navigation, as the objects are correctly accepted.

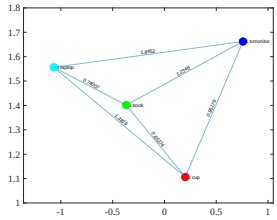
From the permanence plots of figure 11, it is possible to obtain the graphs representation of the figure 12a and 12b. These graphs, obtained already after the navigations matching, spatially represent the objects in the scene. Looking from the first one, in figure 12a, to the second, in figure 12b, one can see that the cup with ID 2 (yellow node) is replaced by the book with ID 6 (green node).

One can see that the distance between the object centroids are not exactly the same in both graphs. Given that the objects that are present in the scene do not move from one navigation to the other, one can conclude that this variation of the distance between objects, as indicated by the scene graphs, are not real. Since the centroids position depends on the position of the point features associated to each object and this, in its turn, depends on the trajectory made with the camera in each navigation, it is comprehensible that these differences exist.

In figure 12c, is represented the output of the Find Scene Changes system for this experiment, where the lost object is detected in a frame of the first navigation and the new object is detected in a frame of the second navigation. Here, the red bounding box identifies the cup with ID 2 as a lost object and the green bounding box identifies the book with ID 6 as a new object. It is notorious the spatial correspondences between the real objects in the frames of figure 12c and the object nodes in the graphs of figures 12a and 12b (in top view).



(a) Graph of the first navigation



(b) Graph of the second navigation

(c) Lost and new objects detected

Fig. 12: **Spatial Representation of the Graphs and Correspondent Detected Scene Changes**. In (a) and (b) the nodes represent the objects and the edges the translation between them. The nodes are represented in the position of the object centroids and have the respective ID and label associated. The edges have the distance between centroids associated. From (a) to (b) the cup with ID 2 is lost and a new book with ID 6 shows up. In (c) it is represented the output of the Find Scene Changes system. On the left, an image frame of the first navigation with the detected missing cup with ID 2 marked with a red bounding box. On the right, an image frame of the second navigation with the detected new book with ID 6 marked with a green bounding box.

With the obtained scene graphs, represented in figure 12, it is possible to build the scene catalogs of this experiment. These catalogs, represented in table II, contain the ID and the label of the objects. Comparing them, it is clear the missing object and the new one.

So, the scene changes initially mentioned are well detected. The system is able to detect the missing of an existing object and the appearance of a new one and is also able to distinguish different objects with equal labels. This experiment also shows that our system is able to recognize an object that he has not seen for a while. So, the sentinel robot is able to understand that the objects in the scene still exist even when he does not see them, which matches the definition of Object Permanence explained in section II-D and allow us to conclude that the results obtained in this experiment are great.

ID	Label
1	tv monitor
2	cup
3	cup
4	laptop

(a) Scene catalog of the first navigation.

ID	Label
1	tv monitor
3	cup
4	laptop
6	book

(b) Scene catalog of the second navigation.

TABLE II: **Scene catalogs of this experiment**. The objects that stay in the scene are the tv monitor with ID 1, the cup with ID 3 and the laptop with ID 4. The cup with ID 2 is lost and the book with ID 6 is new.

V. Conclusion and Future Work

The work described in this thesis aimed to propose a vision-based surveillance mobile robot that, given the firing of an alarm in a home, is able to find the changes in the scene objects by navigating through the house and comparing it with a previous stored navigation. The only sensor used is a video camera, a pinhole one.

Our approach is based on the complementary relationship between a neural network based object detector and a SLAM system. Additionally, the concept of Object Permanence [19] is also applied. Two challenges were identified: (i) difficulty of the object detector in being consistent along the time; (ii) the noise in the MonoSLAM results affects the accuracy of the 3D position attributed to each object, as well as the distance between objects.

The proposed solution, based on the combination of SLAM and object detection, improves the capability of the objects detector to track the objects, so that it can be used to find changes in scenes. Firstly, by attenuating the fast changes in object detections that causes great instability, as it is shown in figure 2. Secondly, by adding the camera FOV notion, where an object is treated according to its state of visibility.

Three main experiments were performed. The first one explored the most common change in scene objects: missing objects. The second one explored the context where an object is replaced by another and there are objects with the same label. Finally, the third experiment explored objects that become visible and not visible across time.

The results of the experiments were found promising. Our system is able to correctly detect the missing, replaced and new objects in the scene. It is robust to homonymous objects and to objects that enter and exit the camera FOV.

In future work, our solution has room to evolve by, i) being able to find changes in more than one environment, like all the different rooms of an house, which would raise the question of knowing the room where the sentinel mobile robot would be, and by ii) being able to find changes in dynamic environments with moving agents or changes in real-time, which would raise the questions of how to lead with dynamic agents and their actions in the environment.

References

[1] Iro Armeni, Zhi-Yang He, Amir Zamir, Junyoung Gwak, Jitendra Malik, Martin Fischer, and Silvio Savarese. 3d scene graph: a structure for unified semantics, 3d space, and camera. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5663–5672, Seoul, Korea (South), October 2019. IEEE.

[2] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *ArXiv*, abs/2004.10934, 2020.

[3] Carlos Campos, Richard Elvira, Juan J. Gomez Rodriguez, Jose M. M. Montiel, and Juan D. Tardos. ORB-SLAM3: An accurate open-source library for visual, visual–inertial, and multimap SLAM. *IEEE Transactions on Robotics*, 37(6):1874–1890, dec 2021.

[4] Javier Civera, Oscar G. Grasa, Andrew J. Davison, and J. M.M. Montiel. 1-Point RANSAC for extended Kalman filtering: Application to real-time structure from motion and visual odometry. *Journal of Field Robotics*, 27(5):609–631, 2010.

[5] Andrew Davison, Ian Reid, Nicholas Molton, and Olivier Stasse. Monoslam: real-time single camera slam. *IEEE transactions on pattern analysis and machine intelligence*, 29:1052–67, 07 2007.

[6] Marius Fehr, Fadri Furrer, Ivan Dryanovski, Jurgen Sturm, Igor Gilitschenski, Roland Siegwart, and Cesar Cadena. TSDF-based change detection for consistent long-term dense reconstruction and dynamic object discovery. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5237–5244, Singapore, Singapore, May 2017. IEEE.

[7] Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[8] Gao Huang, Zhuang Liu, and Kilian Q. Weinberger. Densely connected convolutional networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2261–2269, 2017.

[9] Ue Hwan Kim, Jin Man Park, Taek Jin Song, and Jong Hwan Kim. 3-D Scene Graph: A Sparse and Semantic Representation of Physical Environments for Intelligent Agents. *IEEE Transactions on Cybernetics*, 50(12):4921–4933, 2020.

[10] Tsung-Yi Lin, Piotr Dollár, Ross B. Girshick, Kaiming He, Bharath Hariharan, and Serge J. Belongie. Feature pyramid networks for object detection. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 936–944, 2017.

[11] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014.

[12] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. Path aggregation network for instance segmentation. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8759–8768, 2018.

[13] Emanuele Palazzolo and Cyrill Stachniss. Fast Image-Based Geometric Change Detection Given a 3D Model. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6308–6315, Brisbane, QLD, May 2018. IEEE.

[14] Jean Piaget. *The construction of reality in the child*. Routledge, 1954.

[15] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016.

[16] Antoni Rosinol, Arjun Gupta, Marcus Abate, J. Shi, and Luca Carlone. 3d dynamic scene graphs: Actionable spatial perception with places, objects, and humans. *ArXiv*, abs/2002.06289, 2020.

[17] Antoni Rosinol, Andrew Violette, Marcus Abate, Nathan Hughes, Yun Chang, J. Shi, Arjun Gupta, and Luca Carlone. Kimera: From slam to spatial perception with 3d dynamic scene graphs. *The International Journal of Robotics Research*, 40:1510 – 1546, 2021.

[18] Aviv Shamsian, Ofri Kleinfeld, Amir Globerson, and Gal Chechik. Learning object permanence from video. *ArXiv*, abs/2003.10469, 2020.

[19] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2015.

[20] Mingxing Tan, Ruoming Pang, and Quoc V. Le. Efficientdet: Scalable and efficient object detection. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10778–10787, 2020.

[21] Johanna Wald, Armen Avetisyan, Nassir Navab, Federico Tombari, and Matthias Niessner. Rio: 3d object instance re-localization in changing indoor environments. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 7657–7666, Seoul, Korea (South), October 2019. IEEE.

[22] Johanna Wald, Helisa Dhamo, Nassir Navab, and Federico Tombari. Learning 3d semantic scene graphs from 3d indoor reconstructions. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3960–3969, Seattle, WA, USA, June 2020. IEEE.