Semi-task-dependent and uncertainty-driven world model maintenance

Auton Robot (2015) 38:1–15
DOI 10.1007/s10514-014-9393-0

Semi-task-dependent and uncertainty-driven world model maintenance

Jos Elfring · René van de Molengraft · Maarten Steinbuch

Received: 15 July 2013 / Accepted: 12 June 2014 / Published online: 2 July 2014

Abstract Nearly every task a domestic robot could potentially solve requires a description of the robot's environment which we call a world model. One problem underexposed in the literature is the maintenance of world models. Rather than on creating a world model, this work focuses on finding a strategy that determines when to update which object in the world model. The decision whether or not to update an object is based on the expected information gain obtained by the update, the action cost of the update and the task the robot performs. The proposed strategy is validated during both simulations and real world experiments. The extended series of simulations is performed to show both the performance gain with respect to a benchmark strategy and the effect of the various parameters. The experiments show the proposed approach on different set-ups and in different environments.

Keywords Task dependency · World model verification · Information gain

1 Introduction

As domestic robots are moving into human-populated environments, they are confronted with unstructured and dynamically changing environments. In order to fulfill typical household tasks, such as fetching objects or human-robot interaction, an accurate description of the environment is indispensable. In this work, we will refer to such an environmental description as world model. A world model must (i) contain object poses and (ii) deal with the data association problem, *i.e.*, associate measurements with false detections, newly appeared objects or existing objects in the world model. Ideally, the world model should in addition be able to (iii) predict object positions beyond the last associated measurement and (iv) filter out sensor noise and localization inaccuracies. The world model used throughout this work can be interpreted as a semantic map that tracks object positions and attributes over time. The world model contains information about poses of semantically labeled objects and is complementary to lower level (occupancy) maps used for localization and navigation. For more subtle human-robot interaction or to allow for improved data association, it is beneficial if the world model includes or links to information about other object attributes as well, *e.g.*, size, color, or shape.

In human-populated environments, object positions and attributes may change over time. Therefore, it is important to maintain a world model once it is available, *i.e.*, to keep it up to date. Keeping a world model up to date involves monitoring the environment for new objects and updating the object positions currently present in the world model whenever this is needed. This paper focuses on the latter of these subtasks. This subtask is complicated by the limited observation region and computational resources a robot has; verifying all object attributes all the time often is impossible or infeasible. The main goal of this work is to develop a strategy that can be used to maintain a world model in an efficient way. If needed, the strategy should pro-actively steer the robot while respecting motion constraints implied by other tasks the robot has to perform. Most robots are multi-sensor systems in which each sensor has its own characteristics. The strategy should be aware of the strengths of the various sensors and exploit this knowledge during world model maintenance. Finally, the strategy should enable using simple perception routines by exploiting prior knowledge about where to expect which object.

J. Elfring (✉) · R. van de Molengraft · M. Steinbuch
Eindhoven University of Technology,
5600 MB Eindhoven, The Netherlands
e-mail: joselfring@gmail.com

2 Related work

Various fields of research investigate similar problems. This section attempts to present findings of the most relevant ones.

In the field of persistent simultaneous localization and mapping (SLAM) the aim is to allow an autonomous robot to simultaneously localize, map and navigate despite (visual) changes in the real world (Milford and Wyeth 2010). RatSLAM (Milford and Wyeth 2010) uses an 'experience map' and adds experiences if the robot visits new locations or if previously observed places have a new visual appearance. In Churchill and Newman (2012) the map is extended if localization in a previously visited area fails given previous experiences, *e.g.*, due to a different visual appearance caused by changing weather conditions, and (Konolige and Bowman 2009) use similar ideas to create lifelong visual maps which are maintained online at a frame rate of 30 Hz. Because of their focus on maps used for localization and navigation instead of semantic object maps (Milford and Wyeth 2010; Churchill and Newman 2012; Konolige and Bowman 2009), and the references therein, are considered to be complementary to the proposed approach rather than being alternatives. In Ekvall et al. (2007), knowledge about object poses is incorporated in a SLAM approach, however, due to the computational complexity only ten objects are tracked. This is considered insufficient for real world environments.

In Pangercic et al. (2012) a first generation semantic object maps (SOMs) is extended. The so-called SOM$^+$ map builds upon earlier work (Rusu et al. 2009; Tenorth et al. 2010) and stores information about the pose, appearance and category of objects. However, the SOM$^+$ map directly stores the object poses estimated by perceptual routines. No filtering of, *e.g.*, measurement noise or localization inaccuracies, is performed, data association is excluded and objects are assumed to not change position after being detected. This set of assumptions eliminates the need for a world model maintenance strategy but is considered to be too restrictive in most real world environments. In Mason and Marthi (2012), a world model containing object positions is stored in a database and used for change detection. Contrary to ours, their focus is not on active coordination of world model maintenance. Instead, a robot follows a given set of way points and looks for objects all the time. The main application of Mason et al. (2012) is detecting object disappearance. It presents an unsupervised system which can be used for object disappearance detection and does not focus on finding a world model verification strategy that actively steers the robot and determines when to update which object position.

In active perception literature, the focus of attention of a sensor is steered actively. The work of Unterholzner et al. (2012) introduces active perception in an autonomous driving setting. A utility function combines object importance with the expected reduction of the uncertainty of a world state estimate after detecting an object. By maximizing the utility, the sensor orientation is determined. The work of Davison and Murray (2002) presents the first example of a system applying active vision to SLAM. The mutual, relative uncertainty between features and robot is used to determine which one of the currently visible features should be measured to end up with the best use of the available resources. References Whaite and Ferrie (1997); Stachniss (2006); Sommerlade and Reid (2008) use similar ideas with the purpose of active (scene) exploration. Continuous area sweeping literature like (Ahmadi and Stone 2005, 2006) aims at finding strategies for repeated tasks such as 'keep the building clean', whereas the security sweeping task investigated in Kalra et al. (2004) requires a robot to sweep some area as quickly as possible 'while preventing adversaries from remaining in the area undetected at the end of the sweep'. Both continuous area sweeping and security sweeping require a robot to autonomously decide where to go next. Typical solutions to sweeping problems, like (Kalra et al. 2004; Ahmadi and Stone 2005), calculate robot trajectories by minimizing a cost function using expected (location dependent) rewards.

Rather than calculating *if* an update is needed for a specific object, both the sweeping and active perception works determine which robot move is best by minimizing some cost function. A second difference is that this work demonstrates how the multi-sensor nature of a robot can be exploited, whereas the aforementioned works on active perception and sweeping skip the problem of selecting the most appropriate sensor. Finally, we propose a strategy that is task-dependent whereas these works focus on one specific task.

In the work of Xu et al. (2010) an autonomous city explorer robot is made autonomous by enabling vision-guided motion. Their aim is steering a robot and its focus of attention based on both bottom-up, *e.g.*, information that stands out clearly in the sensor data, and top-down, *e.g.*, prior context knowledge, attention selection mechanisms. A typical task investigated in the work of Xu et al. (2010) is finding and approaching road signs. A similar combined bottom-up/top-down approach can be found in other work on attention selection literature such as (Rasolzadeh et al. 2007; Xu et al. 2009). The survey of Frintrop et al. (2010) provides an extensive overview of visual attention systems. Our aim differs from the attention selection literature in that we focus on steering an autonomous robot with the purpose of maintaining a world model, rather than based on the current sensor data and what appears 'interesting' within this data.

Typical tracking approaches perform well in (highly) dynamical environments, such as robot soccer (Elfring et al. 2011) or people tracking (Schultz et al. 2003). However, in those works all objects are updated whenever possible. An approach that works fine if the field of view covers the majority of the environment and the number of objects is limited but may lead to many superfluous updates or becomes

impossible in domestic scenes with many objects where most objects behave less unpredictable but the field of view only covers a small fraction of the full scene. The first ideas underlying the proposed approach were introduced in Elfring et al. (2012). Contrary to this work, the approach in Elfring et al. (2012) did not incorporate the information gain and possible action costs. Furthermore the criterion for updating an object had to be specified in advance for each possible object and was independent of the task.

3 Contributions

The aim of this work is to present an approach that coordinates the maintenance of a world model. Although Sect. 2 presented many works dealing with related problems, to the best of our knowledge, none of these works investigates world model maintenance as a problem on its own. Our main contribution is an uncertainty-driven coordination strategy that determines when to update which object. Part of this strategy is a mechanism that extends a purely uncertainty-driven strategy such that it becomes task-dependent. We will refer to the combined strategy as semi-task-dependent for reasons explained later. A large number of simulations is performed to quantify the performance gain relative to a benchmark strategy. In addition, the simulations are used to illustrate the effect of the parameters used to configure the proposed strategy. Finally, we demonstrate the improved performance resulting from the semi-task-dependent strategy during a set of experiments on different set-ups including an environment with a fast and unpredictable moving cat and the AMIGO robot (Fig. 1) solving the RoboCup @Home clean up challenge. The set of experiments in addition shows how the proposed strategy considers all available sensors and their relevant characteristics.

The remainder of this paper is organized as follows. Section 4 explains the world modeling algorithm and the world state representation used throughout this work. Section 5 introduces the proposed strategy and Sect. 6 introduces the module executing the verification tasks generated by this strategy. Sections 7 and 8 present the result of series of simulations and experiments that were performed. Finally, Sect. 9 summarizes the most important conclusions and contains an outlook to possible directions for future work.

4 World modeling algorithm

In order to be able to maintain a world model, an algorithm creating and maintaining a world model is required. In this work the probabilistic multiple hypothesis anchoring algorithm introduced in Elfring et al. (2013a) and implemented by

Fig. 1 Photograph (by Bart van Overbeeke) of AMIGO grabbing an object during the RoboCup @Home final 2013

the WIRE[1] stack in ROS is used. Measurements from sensors are associated with objects in the world model using a multiple hypothesis-based data association approach and object instances are linked to grounded symbols using anchoring. Object positions and attributes are tracked using the Bayesian predict–update cycle. The uncertainty in the world state estimate increases during propagation and decreases after measurement updates. A more detailed explanation can be found in Elfring et al. (2013a).

The resulting world model $\mathcal{W}$ contains objects with estimated positions and attributes. Each world model object is associated with a unique identifier (ID) and each position or object property is represented probabilistically *e.g.*, a Gaussian distribution representing the position in some reference frame or a probability mass function over possible colors. If a robot's world model maintenance strategy dictates the verification of some object attribute, data association between the verification measurement and the world model objects is greatly simplified: the verification measurement for example represents the refined position of `cup-5` rather than the position of some random object which was 'accidently' observed. As explained in Sect. 3, the goal of this work is to come up with some strategy that only performs measurements of object instances present in the world model whenever this is considered to be useful. The world model is represented by the WIRE block in the schematic overview given in Fig. 2.

5 Uncertainty driven world model verification

This section explains how the uncertainty in the estimated world state $\mathcal{W}$ will be transformed into a verification task $\mathcal{V}$ given the set of available sensors $\mathcal{S}$. First, the general con-

[1] http://wiki.ros.org/wire.

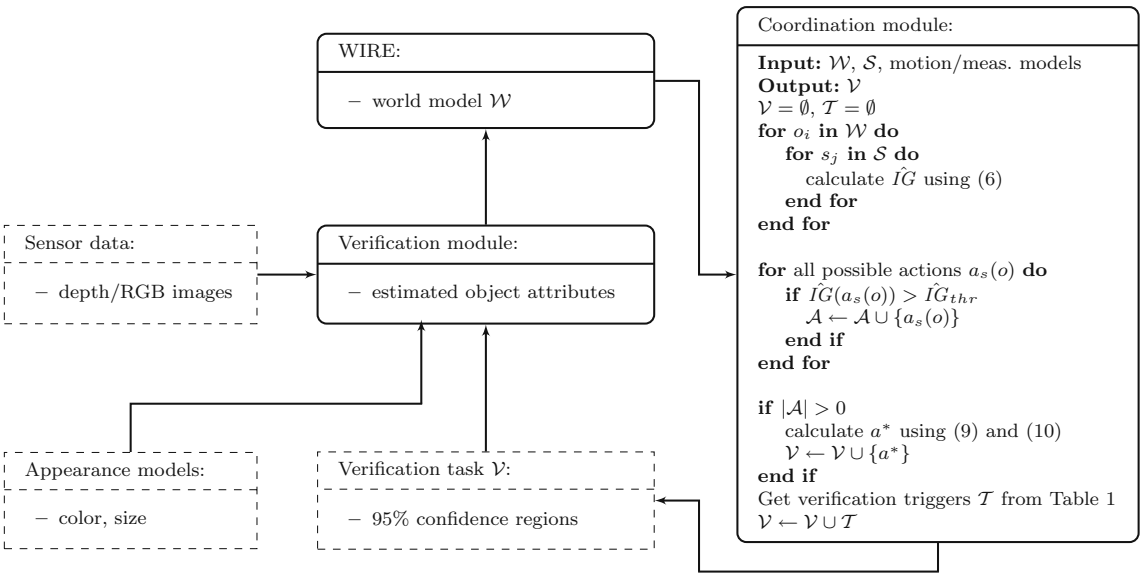

Fig. 2 Schematic overview of the proposed approach. *Dashed boxes* represent data, whereas the other *boxes* represent modules

cept of entropy in information theory is briefly explained in Sect. 5.1. Then Sects. 5.2 and 5.3 introduce the core of the proposed strategy. Finally, Sect. 5.4 explains how the proposed strategy is made semi-task-dependent. All this is done in the coordination module block in Fig. 2.

5.1 Entropy

In the context of information theory, the entropy is a measure of the uncertainty in a random variable. In this work, the world state $\mathcal{W}$ is estimated by a density function and the goal is to control the uncertainty in the estimated world state. For a probability density $p(\mathbf{x})$ over a number of continuous random variables collected in a vector $\mathbf{x}$, the differential entropy $H[\mathbf{x}]$ can be calculated using Bishop (2006):

$$H[\mathbf{x}] = -\int p(\mathbf{x}) \ln p(\mathbf{x}) d\mathbf{x}. \tag{1}$$

By taking the log instead of the ln, the entropy unit can be changed from nats to bits.

The densities $p(\mathbf{x})$ in the world state estimate are not limited to be of any type, however, in the remainder of this section we use Gaussian distributions as a running example, since we believe these are the most popular densities for object state estimation in robotics. For reasons of completeness, we do however also give an entropy estimate for particle filters, since these have proven to be useful for estimating more complex probability distributions.

Consider a multivariate Gaussian distribution with n-dimensional random variable $\mathbf{x}$:

$$\mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{n/2} |\Sigma|^{1/2}} \cdot \exp\left(-\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu})\right) \tag{2}$$

where $\boldsymbol{\mu}$ is the n-dimensional mean vector and Σ is the $n \times n$ covariance matrix. The entropy for (2) can be calculated using Bishop (2006):

$$H_g[\mathbf{x}] = \frac{n}{2} + \frac{n}{2} \ln(2\pi) + \frac{1}{2} \ln |\Sigma|. \tag{3}$$

For a particle filter with N particles each characterized by a pair $(\mathbf{q}_t^i, w_t^i)$, where t is the discrete time step, $\mathbf{q}^i$ is the particle state vector, w the associated normalized weight and $i = 1, \ldots, N$, the particle index, the entropy of the posterior density $p(\mathbf{q}_t \mid Z_t)$ in bits can estimated by Boers et al. (2010):

$$H_{pf}\left[p(\mathbf{q}_t \mid Z_t)\right] \approx \log\left(\sum_{i=1}^{N} p(\mathbf{z}_t \mid \mathbf{q}_t^i) w_{t-1}^i\right) - \sum_{i=1}^{N} \log\left(p(\mathbf{z}_t \mid \mathbf{q}_t^i) \sum_{j=1}^{N} p(\mathbf{q}_t^i \mid \mathbf{q}_{t-1}^j) w_{t-1}^j\right) w_t^i, \tag{4}$$

where $Z_t = \{\mathbf{z}_0, \ldots, \mathbf{z}_t\}$ is the set of measurement vectors up to time step t.

5.2 Information gain

After updating an object in the world model, the uncertainty reduces and, therefore, the entropy drops. We have defined the information gain for object o, after updating it with a measurement $\mathbf{z}$ of the object state $\mathbf{x}_o$ using sensor s:

$$IG(o, s, \mathbf{x}_o, \mathbf{z}) = H\left[p(o)\right] - H\left[p(o \mid s, \mathbf{x}_o, \mathbf{z})\right]. \quad (5)$$

The first term on the right hand side is the entropy before updating, whereas the second term represents the same entropy after an update with sensor s. The explicit time dependence is left out for ease of notation.

Updating an object's attribute only makes sense if the information gain is 'sufficiently' high, *i.e.*, if the information gain is above some threshold. This threshold directly affects the robot's behavior. If it is set very high, minimal effort is put into updating object attributes at the cost of an increased overall world model uncertainty. A very low threshold, on the other hand, leads to many potentially superfluous updates but a lower overall uncertainty. Here, a threshold IG_{thr} is set manually. Future work includes making IG_{thr} both task and object dependent. The effect of IG_{thr} will be thoroughly investigated in Sect. 7.1.

One problem with the information gain as defined in (5) is that it can only be calculated after updating an object, whereas it is needed to determine if an update is useful. To overcome this problem we propose a forward simulation which allows for calculating an expected information gain $\hat{IG}$; objects can be updated with a fake measurement $\hat{\mathbf{z}}$ at the location currently estimated.

$$\hat{IG}(o, s, \mathbf{x}_o, \hat{\mathbf{z}}) = H\left[p(o)\right] - H\left[p(o \mid s, \mathbf{x}_o, \hat{\mathbf{z}})\right]. \quad (6)$$

A forward simulation is considered to be representative since at this point we are interested in the change in the state uncertainty rather than the exact value of the state. In this work measurements are represented by Gaussian distributions, hence the fake measurement $\hat{\mathbf{z}}$ used to update object o has a mean corresponding to the estimated object state $\mathbf{x}_o$. The covariance matrix Σ_o associated with the measurement equals the covariance of the last measurement used to update object o if only a single sensor can be used for performing measurements. In a multi-sensor setting, each sensor has its own characteristics hence each sensor has its own measurement covariance matrix for each of the objects.

Performing 'fake' updates might seem quite an effort at first glance. If all objects are within the field of view of the robot, fake updates save computation time with respect to real measurements. Typically only a subset of the world model objects is visible hence the decision to update one object automatically excludes another object. In this case fake updates are inevitable for calculating the information gains which allow for a thought-out decision. A real update usually requires a real measurement, which besides an object detection often requires a navigation task. Therefore, real measurements would typically take orders of magnitude longer than the milliseconds needed for the proposed strategy.

Action $a_s(o)$, represents the action 'update object o using sensor s'. The forward simulation allows for calculating the expected information gain $\hat{IG}\left[a_s(o)\right]$ associated with an action $a_s(o)$. This is done by first determining the required path to the associated object and then discretizing the path in a finite number of way points based on the expected velocity and the discrete time step. At each time step a propagation is done and the possible updates are performed. This way, the expected execution time and the related number of predict-update cycles is considered during the calculation of the expected information gain $\hat{IG}\left[a_s(o)\right]$ for an action $a_s(o)$. If and only if this expected information gain is above the threshold IG_{thr}, the corresponding action will be added to the set $\mathcal{A}$:

$$\mathcal{A} \leftarrow \mathcal{A} \cup \{a_s(o)\}. \quad (7)$$

5.3 Expected utility

Section 5.2 explained how potential verification actions are omitted based on the expected information gain. This section explains how a single action is selected from the remaining set of actions $\mathcal{A}$ associated with a sufficient information gain.

The action cost for updating object o with sensor s is defined as follows:

$$V\left[a_s(o)\right] = \frac{d_{o,s}}{V_{scale}}, \quad (8)$$

where V_{scale} is a constant scaling factor and $d_{o,s}$ is a distance term. In this work $d_{o,s}$ equals the distance-to-be-traveled and V_{scale} is the length of the room the robot is in. An alternative choice could be:

$$V_{scale} = \max_{o \in \mathcal{W}, s \in \mathcal{S}} d_{o,s}.$$

At this point, we decided to not define an action cost for the actual measurement of an object, *i.e.*, the cost of detecting an object is considered to be equally expensive for each of the objects and each of the sensors. Whether or not this assumption is reasonable depends on the perception modules and sensors available for performing object detections. The assumption should be relaxed if actions of varying complexity are considered, *e.g.*, weighting an object or opening a box to determine whether or not it is empty is typically more expensive than a colored blob detection, or if the available sensors operate at different time scales, *e.g.*, a camera which runs at 30 Hz versus a tilting laser range finder which needs 1 s to make a full scan. Actions associated with a varying cost can be dealt with by introducing an additive 'cost-of-the-measurement' term in (8). A natural measure affecting the magnitude of this term would be the time interval associated with the measurement.

An expected utility for action $a_s(o)$ is defined as being:

$$U\left[a_s(o)\right] = -\alpha V\left[a_s(o)\right] + \frac{\hat{IG}\left[a_s(o)\right]}{IG_{norm}}, \tag{9}$$

where α is a weight that can be used to tune the relative importance of the action cost and the expected information gain and IG_{norm} is a normalizing term. The weight α can be set based on the desired behavior. An extensive analysis of the effect of α on the resulting robot behavior is presented in Sect. 7.2.

Finally, the optimal action with respect to the expected utility is selected:

$$a^* = \underset{a_s(o)\in\mathcal{A}}{\operatorname{argmax}}\, U\left[a_s(o)\right]. \tag{10}$$

Note that dependent on the settings $\mathcal{A}$ can be empty and at most has the size of the product of the number of objects and sensors. The optimal action is added to the verification set $\mathcal{V}$:

$$\mathcal{V} \leftarrow \mathcal{V} \cup \{a^*\}, \tag{11}$$

hence at this point $|\mathcal{V}| \leq 1$.

A few remarks can be made with respect to the optimality criterion (10). First of all, the information gain is a relative quantity that does not consider the current entropy. Equal information gains are considered equally valuable independent of the underlying entropies. One reason for this decision is that the measurement accuracy typically differs per object and sensor hence there is no guarantee that the entropies of different objects can be brought down to the same level. A second observation is that the strategy might favor actions involving many objects with a small information gain over updating a single object with a large information gain, since the overall uncertainty is minimized rather than the worst case uncertainty. If single objects are involved in tasks their uncertainty reduction will be enforced by the task-dependent component explained later. Section 7 will provide an in-depth analysis of the consequences of the proposed strategy on both the overall and the worst case object entropy.

So far, it was tacitly assumed that the world model maintenance strategy was allowed to actuate the full robot. However, it depends on, *e.g.*, the robot's task or sensor configuration if this assumption holds. If other components require actuating the robot, *e.g.*, to check the availability of the path while driving, the set of objects considered must be limited to the objects within the robot's field of view. As a result, the length of the path $d_{o,s}$ in (8) equals zero and the expected utility only contains one term. In such cases the proposed strategy remains unchanged, however, the number of possible actions reduces. Only the subset of actions respecting the robot motion constraints implied by other modules will have to be considered. This does not limit the freedom of other components to set verification triggers as explained in the next section.

5.4 Semi-task-dependent verification strategy

The strategy as proposed in Sect. 5.3 is based on the world model uncertainty only. Ideally, this strategy depends on the task at hand too. If a robot has to pick up a mug from a table, the allowed position uncertainty of the mug is much smaller than when passing this table during a navigation task. However, reasoning about which objects are and which objects are not relevant during a task is a very complex problem which depends on, *e.g.*, the task specification, the success of subtasks and the dynamics of the environment, and falls outside the scope of this work.

In order to not keep this relevant problem unexplored the coordination module that calculates the expected utilities and decides when to update which object attributes using which sensor also accepts verification triggers collected in the set $\mathcal{T}$. If a reasoning engine or some module executing a plan considers it relevant to update an object attribute, the module can send a trigger to the coordinator and the coordinator will enforce an update, independent of the information gain. This way the robot is guaranteed to have the most accurate estimate possible at that specific time given its context.

By triggering this update mechanism before executing pre-defined primitive actions, a semi-task-dependent behavior can be obtained. In the experiments presented later, the task 'pick up object X' adds the action 'verify the position of object X' to $\mathcal{T}$. This way, object X will be updated if (i) the expected utility of object X enforces an update, or (ii) the object must be picked up. With this mechanism, a navigation task along a table with a mug on top of it and a pick-up task of that same mug may lead to different verification strategies. The idea is that a limited number of action/verification trigger pairs can cover the majority of the task-dependent updates required for typical tasks under the assumption of normal operation. Table 1 summarizes some of the pairs implemented in this work. Before the coordination module sends out a verification task, $\mathcal{T}$ is merged into the set generated by the uncertainty driven component of the strategy:

$$\mathcal{V} \leftarrow \mathcal{V} \cup \mathcal{T} \tag{12}$$

Verification triggers are prioritized over the verification task which is based on the expected utility. If a robot sends out multiple verification tasks, they are executed chronologically in the order of arrival. Currently, no merging of verification tasks is performed. If both the uncertainty-driven component and the task-dependent component dictate the refinement of the position of some object Y, its position will be verified twice.

Table 1 Overview of action/verification trigger pairs on primitive robot actions.

Action	Verification trigger
Grab object X	Refine position object X
Navigate to location A	–
Navigate to object Y	Refine position object Y
Release object Z	–

```
<object_class name="kiwi">
  <size x="0.5" y="0.6" z="0.6"/>
  <color_channel channel="RGB" />
  <color_range ch1_min="0" ch1_max="255"
    ch2_min="0" ch2_max="255"
    ch3_min="0" ch3_max="10" />
</object_class>
```

Fig. 3 Code snippet showing how the appearance model used during the first experiment is defined

6 Verification module

Once the world model maintenance strategy introduced in the previous section generated a verification task $\mathcal{V}$, the task must be executed. This section briefly explains the verification module used throughout this work. The verification module is independent of the proposed world model verification strategy and can be replaced based on, *e.g.*, sensor configuration or available data and perceptual routines.

Throughout this work, a verification task $\mathcal{V}$ sent from the coordination module to the verification module only contains the 3D region of interest (ROI) based on the 95 % confidence interval of the predicted 3D position, a unique identifier and a semantic label. In case object attributes like color or size are tracked by the world model, the coordination module can in addition add these to the verification task.

In case insufficient object appearance information is provided by the coordination module, this information is loaded from a knowledge base. Figure 3, shows a part of the knowledge used throughout this work. It describes the appearance of a cat named Kiwi. Since the current implementation contains 3D positions instead of 6D poses, the size is large in all three directions. Together with the 3D ROI provided by the coordination module, the size of the object determines which part of the sensor data has to be considered. Within this sensor data, the actual verification can be performed.

Once the position is verified, it is fed to the world model together with the unique ID contained in the verification task. The ID simplifies the data association within the world model. Two special cases might appear.

1. If the object is occluded, no verification can be done, hence no information is sent to the world model.
2. If the object cannot be found at the expected location, the object is considered to be lost. The object position will be represented by a uniform distribution over the environment. Contrary to an irreversible removal of the object from the world model, this allows for re-associating the object once it is re-observed later. Incorporating more advanced models such as 'the object is somewhere outside the current ROI' is outside the scope of this work.

7 Simulations

This section presents simulation results. The purpose of the set of simulations is twofold. It quantifies the relative performance compared to a benchmark strategy and it shows the effect of the parameters on the robot behavior. Simulations were chosen for this purpose since they allow for a large number of trials under the exact same conditions and therefore enable both a fair comparison and a statistical analysis. The effect of the information gain threshold IG_{thr} is investigated in Sect. 7.1 and the effect of the relative weight α is analyzed in Sect. 7.2. Then Sect. 7.3 discusses the impact of unexpected object movements unreliable perception.

7.1 Effect of the information gain threshold

The benchmark strategy lets a perfectly localized robot with one sensor follow a predefined path along all objects. All objects within the field of view are updated and the sensor, inspired by a Kinect (Khoshelham and Elberink 2012), is able to detect objects within a range of [0.4, 2.0] m. Obstacle avoidance is omitted and as a result the robot always drives in a straight line from its current position to the target position.

All simulations were performed 250 times and the simulated time per simulation was 16 min and 40 s. One square room with a length of 10 m contained ten objects that were randomly spread. The objects did not move and measurements were filtered using a Kalman filter. The motion model uncertainty represented by the covariance matrix of the Gaussian process noise equals $\sigma_P I$, where σ_P is randomly sampled from the interval $[0.0, 0.01]$ and I is an identity matrix of appropriate size. The scalar σ_P is made object dependent to represent the fact that the dynamics of different objects in a room are known with different levels of confidence. The measurement noise was represented by a Gaussian distribution with covariance matrix $\sigma_M I$. Again I is an identity matrix and σ_M was fixed at an arbitrary value of 0.01, *i.e.*, during the simulations all object positions are assumed to be detected with the same accuracy. A brief parameter study showed that different values for σ_P and σ_M led to results similar to the ones presented in this section. The object state was equal to the object position, the relative weight

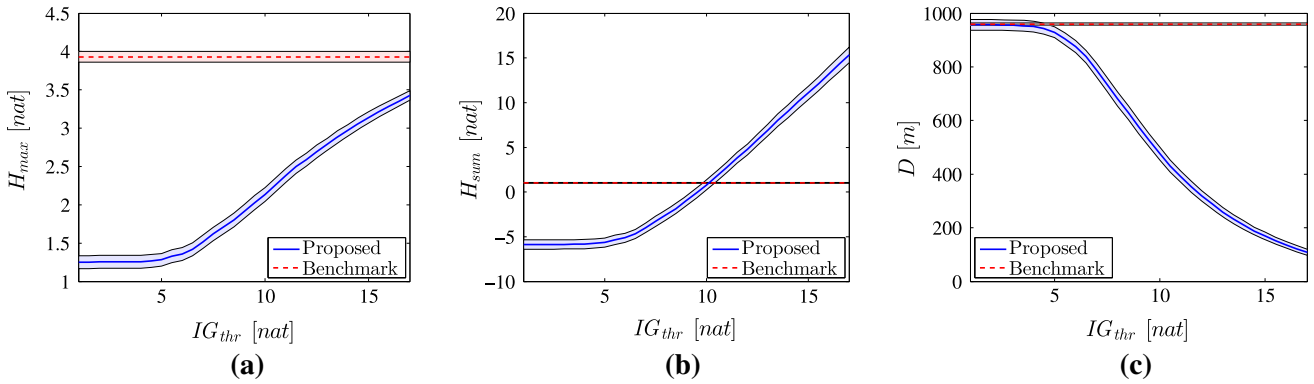

Fig. 4 Maximum and total entropy of the estimated world model and the distance D driven during the simulation as a function of IG_{thr}

$\alpha = 0.75$ and $IG_{norm} = 5$ nat was used as a normalizing constant during all simulations in this section.

During each simulation and at each time step, the entropy in nat for the probability distribution representing an object state estimate was calculated. The maximum entropy was calculated using:

$$H_{max} = \max_{o_i \in \mathcal{W}} H[o_i]. \tag{13}$$

Figure 4a shows the results together with the 95 % confidence intervals for both the proposed and the benchmark strategy and for a range of different values for the threshold on the information gain $\hat{IG}[a_s(o)]$. A few conclusions can be drawn.

Compared to the benchmark strategy, the worst case object entropy is lower when following the proposed strategy over the full range of values set for IG_{thr}. This is the result of selecting the action that maximizes the information gain. The larger the uncertainty, the larger the entropy and the larger the expected information gain, hence updating uncertain objects is preferred if the action costs are similar. For larger values of IG_{thr}, the worst case object entropy increases, since larger uncertainties are required in order to let the estimated information gain fall above the threshold IG_{thr}. For low values of IG_{thr}, *i.e.*, $IG_{thr} < 4$ nat, $\mathcal{A}$ equals the set of all possible actions and H_{max} will not be affected by further lowering IG_{thr}.

The second variable that was investigated is the sum of the entropies of all probability densities representing object positions, which is a measure for the total uncertainty in the world state estimate:

$$H_{sum} = \sum_{i=1}^{|\mathcal{W}|} H[o_i]. \tag{14}$$

Figure 4b shows the results together with the 95 % confidence intervals.

The curve associated with the proposed strategy shows a linear increase for a $IG_{thr} > 6$ nat, since a higher threshold inevitably leads to fewer object updates and hence an increased overall uncertainty. In this set of simulations, lowering the value below 4 nat, like in Fig. 4a, did not have a significant effect on the overall uncertainty; objects were updated at all the time steps already. A second observation is that the proposed strategy keeps the overall uncertainty below the overall uncertainty obtained by the benchmark strategy for values up till $IG_{thr} = 10$ nat.

The distance D driven by the robot during the full simulation is plotted in Fig. 4c. For low values of IG_{thr}, the distance is more or less constant; like with the benchmark strategy updates are performed all the time. However, for a threshold $IG_{thr} > 5$ nat, the distance driven by the robot drops rapidly. For a threshold of $IG_{thr} = 9$ nat, the distance is about half of the distance driven when executing the benchmark strategy, for a threshold around $IG_{thr} = 16$ nat, the distance is almost a factor ten lower. Figure 5a and b show the paths driven by the robot for one of the simulations over a time window of 100 s for both the proposed and the benchmark strategy. For an information gain threshold of $IG_{thr} = 15$ nat, the robot is mostly idle since the information gains are below the threshold most of the time. For $IG_{thr} = 5$ nat, the robot moves much more. The benchmark strategy lets the robot follow a fixed path over and over again. However, Fig. 4 already indicated that the benchmark route is not as thought-out as the path enforced by the proposed strategy for a low threshold.

In summary, it can be concluded that the proposed approach is able to keep the worst case object entropy significantly lower than the benchmark strategy for reasonable values of the threshold on the information gain IG_{thr}. At the same time, the overall uncertainty is lower if IG_{thr} is selected with care. These results are achieved with a robot that drives less, *i.e.*, by cleverly selecting actions both the worst case and the overall uncertainty are significantly lowered while the required effort reduces too.

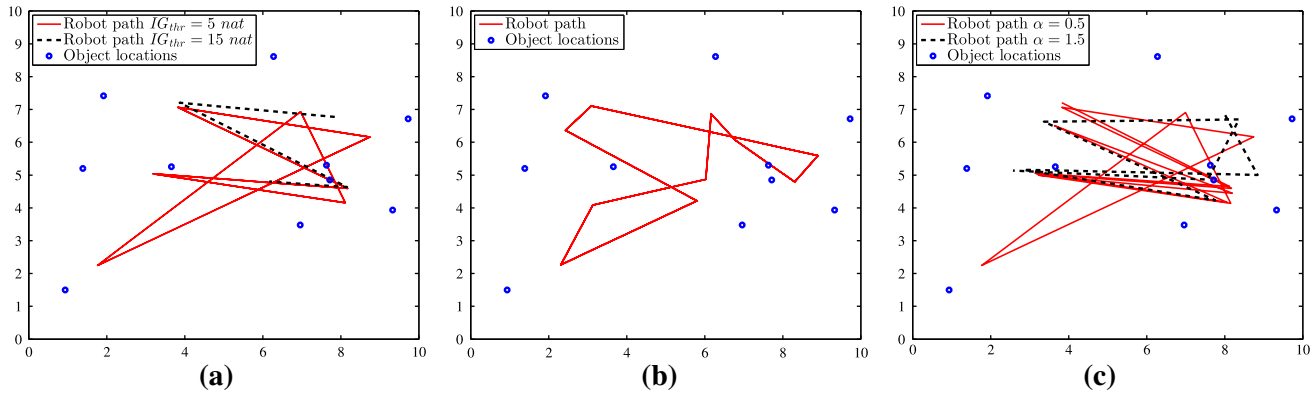

Fig. 5 Example trajectories driven by the robot as a result of both the benchmark strategy and the proposed strategy for different settings

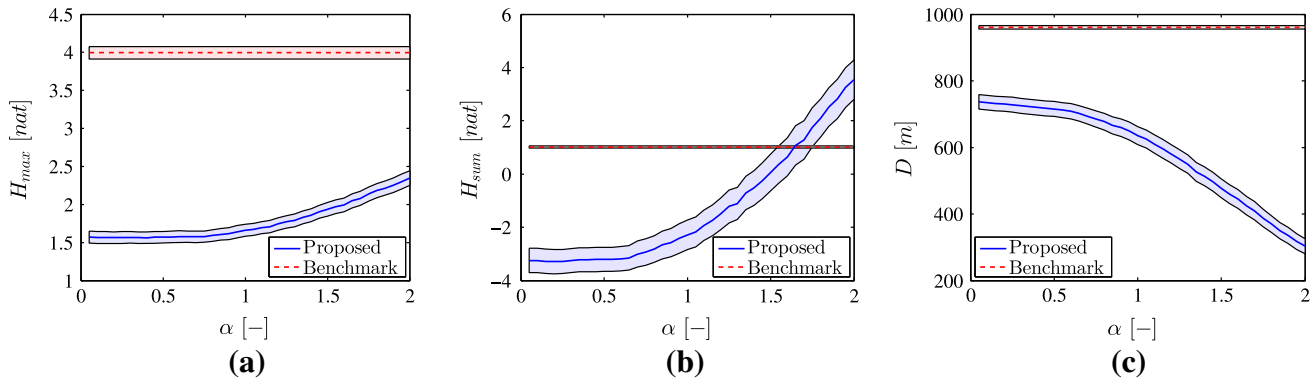

Fig. 6 Maximum and total entropy of the estimated world model and the distance D driven during the simulation as a function of the relative weight α

7.2 Effect of the relative weight

The same analysis is performed for the relative weight α. The expected information gain threshold was set to $IG_{thr} = 8$ nat, α was varied between 0.05 and 2.0 and all other settings were as in Sect. 7.1. Figure 6 shows the results of the analysis.

If the relative weight α increases, the action costs gain importance. As a result, updating nearby objects is preferred over a higher information gain at faraway objects. Therefore, the total distance driven by the robot reduces with increasing α, as shown in Fig. 6c. Over the full range of values to which α was set, the distance driven is significantly less than with the benchmark strategy. Increasing α leads to an increase in both the total and the worst case world model object entropies as shown in Fig. 6a and b. Again, the proposed strategy outperforms the benchmark strategy. The worst case entropy is lower for all values to which α was set, whereas the total uncertainty is lower for $\alpha < 1.7$. The path driven by the robot for different values of α is shown in Fig. 5c. This figure confirms an increased path length for lower values of α. Paths obtained with low α values are similar to paths obtained by setting a low IG_{thr}, as shown in Fig. 5a and c.

To be able to further interpret the differences, two bar charts are given for one representative simulation during which $\alpha = 1$. Figure 7a shows the number of times each of the ten objects is updated during the simulation, whereas Fig. 7b shows how many times each of the objects was approached, *i.e.*, the number of times each action is selected.

For the benchmark strategy, Fig. 7b is uniform over all actions, whereas the proposed strategy shows a more complex distribution in which some actions are never selected. This can be explained by the fact that the associated objects are sufficiently updated during other actions, *e.g.*, because they are close to other objects or can be updated on the way to another object, as can be concluded from Fig. 7a. Another thing that can be observed in these bar charts is that the benchmark strategy performs 263 actions versus 104 when using the proposed strategy, and does more updates, 973 versus 515. These differences demonstrate the improved efficiency resulting from the proposed strategy with respect to the benchmark strategy; with a lower number of actions, a lower number of updates and while driving less, a lower level of uncertainty is achieved. In the real world, this means that using the proposed strategy, a robot is able to

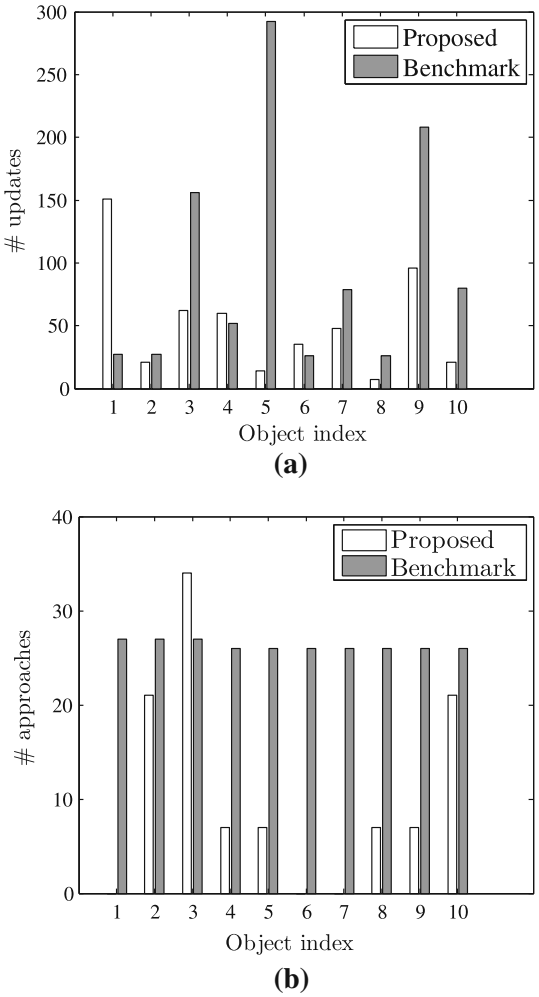

Fig. 7 Number of times each object gets updated in (**a**) and number of times each action is selected in (**b**) during a simulation with $\alpha = 1$

make an informed decision about when to update which object.

7.3 Impact of unpredicted object movements and unreliable perception

It is important to notice that the results presented in this section are simulation results. In the real world, unexpected object movements and failure of perceptual routines or sensors may lead to false negatives (not detecting an object which is present), false positives (detecting an object which is not present) or object loss in the world model.

A first consequence of object loss will be sub-optimality since the expected utilities $U\,[a_s(o)]$ in (10) do not match the actual utilities. The proposed strategy requires a position estimate whereas the position of a lost object is represented by a uniform distribution over the environment, as explained in Sect. 6. As a second consequence, lost objects are not considered by the maintenance strategy and will never be approached, unless they are accidently re-observed. Active object search strategies like the ones described in Joho et al. (2011); Elfring et al. (2013b) can increase the probability of re-detecting objects which are lost as a result of unexpected movements. However, active object search is time-consuming and the success of the search cannot be guaranteed. Whether or not active object search is desirable therefore depends on the task at hand. Investigating strategies which combine active object search and world model maintenance in a task-dependent manner is left for future work.

A single false negative could have a large impact on the system's performance, *e.g.*, the object might unfairly be considered lost. During the experiments, from which the results are presented later, each verification task is associated with a certain time interval which depends on the time needed by the perception module used for the re-detection and is typically smaller than a second. Instead of re-detecting the object in a single camera image, point cloud or laser scan a series of images, point clouds or scans is considered. As a result, the effect of temporary false negatives is limited. If false negatives are persistent over the time interval of the verification task, *e.g.*, due to the failure of a sensor or software crashes, the object will unfairly be considered lost. As stated before, lost objects will cause optimal actions to be sub-optimal in retrospective and in the light of (9). Adding mechanisms that let the robot observe an object from different viewpoints could lead to an increased robustness against persisting false negatives. However, false negatives did never appear problematic during the experiments and for that reason such mechanisms are not further investigated in this work.

The number of false positives generated by perceptual routines used for verification tasks is reduced by exploiting prior knowledge available through the world model. 'Refine the position of the green cup with a given geometry and which should appear in a known ROI within the data from sensor X' is easier than 'see if there is some object somewhere within the sensor data'. If despite the prior knowledge a false positive is generated, the world model will incorrectly update the object position. As a result, the world model will be overly confident about incorrect information. This a direct consequence of the probabilistic models used in the world model. These models consider a false positive which matches both the expected object's appearance and position unlikely compared to a true positive and therefore prefer an object update over the hypothesis of a false positive object detection. The impact of a single false negative is small compared to the impact of a single false positive. For that reason, the perception modules used are tuned to be somewhat conservative, *i.e.*, in case of doubt an object detection will not be published to the world model. As a result the perception modules used for verification tasks did not generate any false positive during all of our experiments.

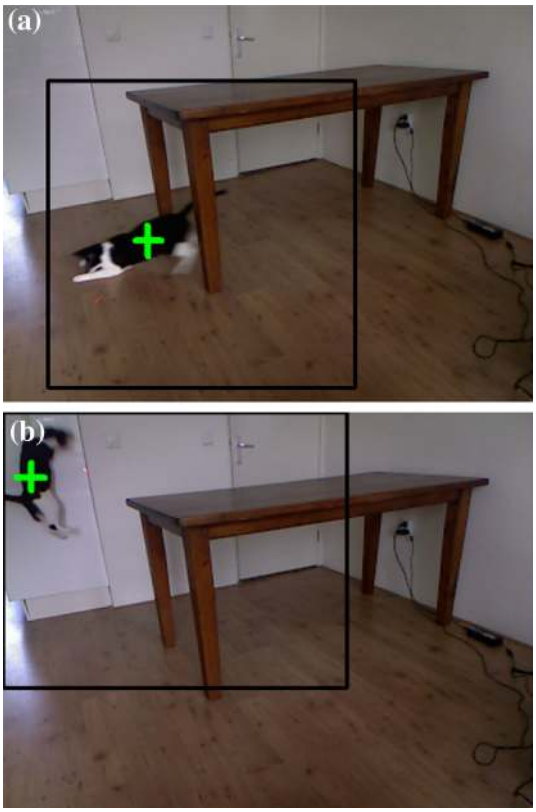

Fig. 8 Running cat in (**a**) and jumping cat in (**b**), the *green crosses* represent the position estimates generated by the verification module, the *black rectangle* represents the 3D ROI based on the 95 % confidence interval of the predicted position projected on the 2D image

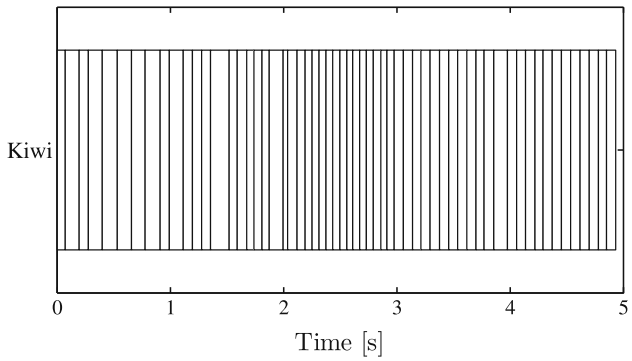

Fig. 9 *Bar chart* showing when the position of the cat named Kiwi was verified

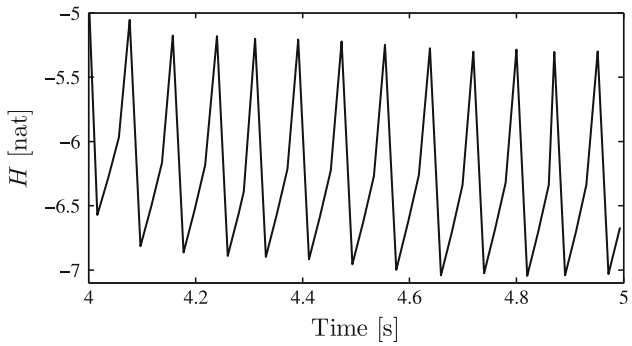

Fig. 10 Entropy during part of the cat tracking experiment

8 Experiments

8.1 Single sensor experiment

The goal of the first real world experiment is to investigate the update rate resulting from the proposed strategy in a highly dynamical scene. In the experiment, one Kinect is used to keep the position of a cat accurate despite its highly unpredictable and fast movements. Various short videos were recorded all leading to similar results. The results presented here are obtained on a movie in which the cat runs back and forward and includes a jump. Figure 8 shows two representative screenshots. The Kinect has a frame rate of 30 Hz, hence updates can be performed at most at this frequency. The cat is tracked using a Kalman filter with a constant velocity motion model and zero-mean Gaussian process noise with a constant and diagonal covariance matrix. Since the expected position of the cat is known, re-detecting the cat boils down to finding a black blob with a specific size in a 3D region of interest based on the 95 % confidence interval of the predicted position, see Fig. 3. The detection takes less than 1 ms on average and Fig. 8 shows two typical detections together with the 2D projection of the 3D ROI on the camera image.

The main result of this experiment is shown in Fig. 9. This figure shows when the position of the cat is updated. As shown in this figure, the update rate is time dependent and varies between approximately 10 and 15 Hz. The reason for this time-dependent update rate lies in the non-constant measurement variance and the resulting time dependent expected information gain. This can be explained by the resolution of the depth information, which drops with the distance hence the measurement variance on the position changes when the cat moves.

Figure 10 shows the entropy associated with the cat's position estimate during a part of the experiment. During periods without measurement updates the uncertainty associated with the position estimate increases. As a result, the expected information gain resulting from a position update increases over time and every third frame, this expected information gain is sufficient to enforce a measurement update. The position uncertainty, and therefore the world model entropy, decreases after the updates. Over the time interval of this figure, measurements are getting more and more accurate. More accurate measurements contain more information and as a result the entropy after an update decreases over this time interval.

This experiment shows how the proposed strategy combines (i) the current uncertainty resulting from the unpre-

Table 2 Time line of the experiment. Locations are indicated in Fig. 11

Location	Event	Recognized objects	World model objects	Verification tasks
1	Speech command	–	–	–
2	Navigate to first pre-defined location	–	–	–
2	Switch on object recognition	Coke can, tea pack	Coke can, tea pack	–
2	Grab coke can	–	Coke can, tea pack	Refine position coke
3	Dispose coke can	–	Tea pack	–
4	Navigate to second pre-defined location	–	Tea pack	–
4	Switch on object recognition	Coke can	Tea pack, coke can	–
4	Grab coke can	–	Tea pack, coke can	Refine position coke
3	Dispose coke can	–	Tea pack	–

dictable nature of the object movements with (ii) the measurement uncertainty of the specific sensor and then comes to a conclusion about whether or not an update is useful.

8.2 Experiments using AMIGO

The goal of this experiment is to show the task-dependent component of the proposed strategy on the AMIGO robot. A photograph showing AMIGO during the final of RoboCup 2013 in Eindhoven, at which team Tech United ended up third, was already shown in Fig. 1. The results presented here are obtained during a 'rehearsal' two weeks before the actual tournament.

During the RoboCup @Home clean-up challenge, the robot is asked to clean up a room. After being told which room, AMIGO drives towards the room and looks for objects at some pre-defined locations, *e.g.*, on tables, cabinets, shelves. AMIGO stops at each of the pre-defined locations and switches on his object recognition module, see Fig. 12. Objects are recognized using a custom implementation of the Linemod algorithm (Hinterstoisser et al. 2011). This algorithm combines color information with 2D and 3D gradients to recognize the class of an object. An overview of the different stages of the experiment is given in Table 2. The numbers in the first column refer to locations indicated in the 2D occupancy map used for navigation shown in Fig. 11.

First, AMIGO enters the room (first location in Fig. 11) and gets the command to clean up a room, in this case the bed room. It drives to the cabinet (location 2) and switches on object recognition. A coke can and a tea pack are observed and added to the world model. According to the task description, coke cans must be disposed into trash bins, hence AMIGO plans to pick up the coke can. In order to do so, AMIGO repositions itself with respect to the coke can and a verification trigger is enforced by the mechanism explained in Sect. 5.4. This verification task asks to 'refine the position of the coke can that will be picked up' and is needed since the available position estimate is insufficiently accurate for

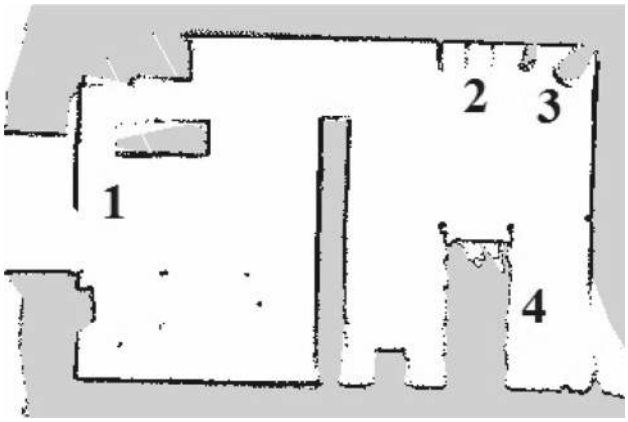

Fig. 11 Occupancy map used for localization with the various relevant locations

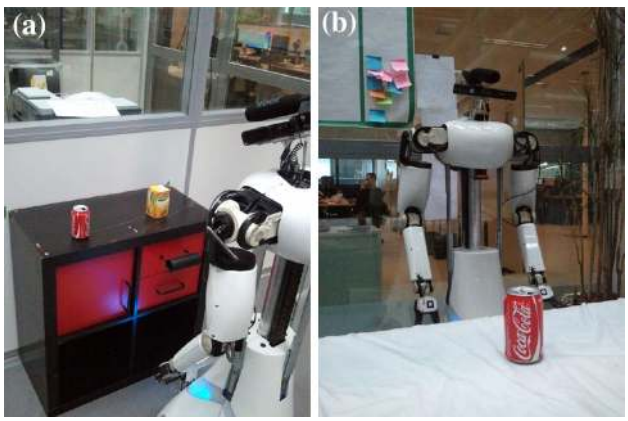

Fig. 12 Photographs taken during the experiment. AMIGO looking for objects

a successful grab action from AMIGO's current location. The new position estimate is fed to the world model and the updated object position is used by AMIGO to grab the object, see Fig. 13. After grabbing, AMIGO disposes the coke can into the trash bin (location 3). The task description does not mention a desired location for the tea pack hence AMIGO ignores the tea pack and moves towards the next pre-defined location (location 4).

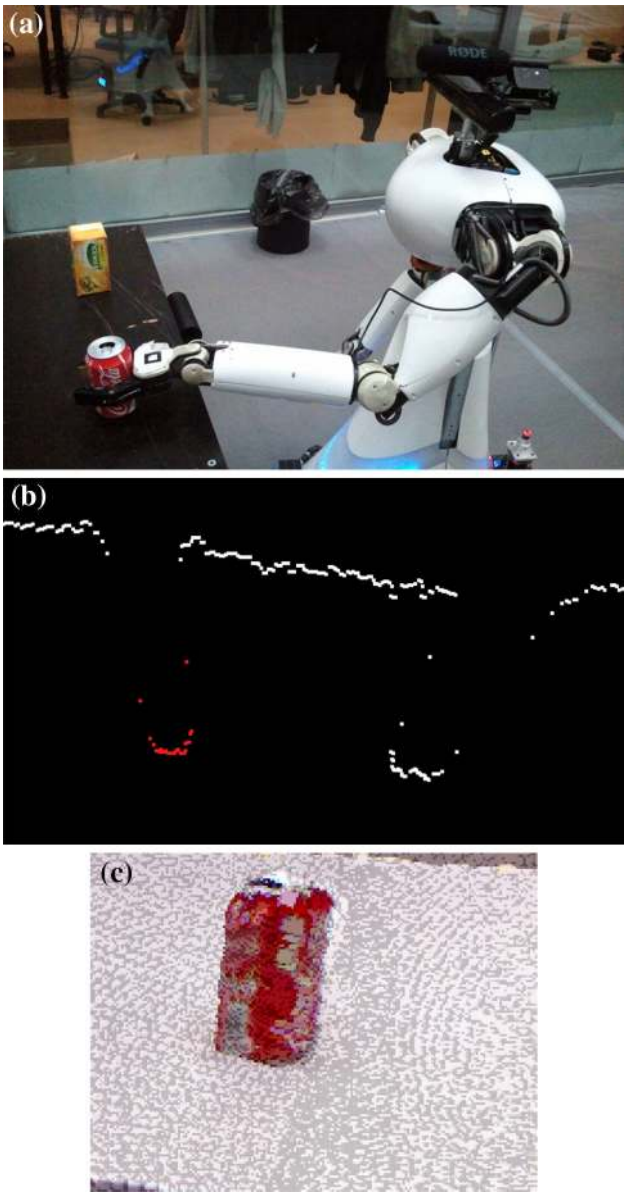

Fig. 13 Photograph taken during the experiment in (**a**). The part depicted shows the grab action based on the verified object position. Part of the laser data used to verify the position is shown in (**b**). In *red* the data that falls within the region of interest. Data within a typical ROI using the Kinect for verification is shown in (**c**)

At this location, AMIGO again switches on his perception module and recognizes a coke can. The coke can must be picked up hence AMIGO prepares a grab action and sends a verification trigger. After refining the position of the coke can, it is picked up and thrown into the trash bin (location 3). Now, the cleaning of the room is finished and AMIGO drives back towards its initial position (location 1).

AMIGO has two different sensors that can be used to solve a verification task. It can either use the laser scanner mounted at his torso or its Kinect. In case the torso laser is used, first, the spindle height is adjusted such that the object gets within the field of view of the laser. Then with the region of interest provided by the coordination module a clustering algorithm suffices to determine an accurate position estimate. The laser data used for the verification measurement is shown in Fig. 13b. The laser data within the ROI is colored red. Apart from the laser scanner AMIGO can also decide to use the Kinect. In this case, the object is segmented by filtering out the horizontal plane supporting the object and clustering the points that remain within the ROI. Typical point cloud data within the ROI is shown in Fig. 13c.

The option using the torso laser generates a more accurate position estimate due to the low measurement noise of the laser scanner with respect to the Kinect. As a result, it leads to a higher information gain. For that reason, the laser scanner is used for re-detecting the coke at location 2. During this experiment, the action costs were equal, since both sensors did not require a navigation task. One could influence the preference for the sensor used by associating different action costs with the different sensors as explained in Sect. 5.3. If the object that needs to be picked up stands at a height of approximately 0.7 m or less, the object is below the minimum height of the torso laser, as shown in Fig. 12b. In this case only the Kinect is available for verifying the object position. For that reason, the Kinect is used for the verification task at location 4.

During this experiment, the objects are known to only move as a result of robot actions. For that reason the uncertainty associated with the object's motion model is very low and the expected information gain is never sufficient for enforcing an object update.

9 Conclusions and future work

For successful operation in complex and dynamically changing environments robots need an up-to-date environmental description. Updating all object positions all the time is both infeasible and impossible. This work investigated how such descriptions can be maintained in an efficient manner. A strategy which is semi-task-dependent and uncertainty-driven strategy and determines when to update which object using which sensor was presented.

Adopting this strategy that combines the expected information gain with the action cost led to both a reduction in the worst case object position uncertainty and the average object position uncertainty while driving less during an extended analysis involving many long lasting simulations. Real world experiments with a cat moving fast and unpredictable showed how the strategy enforces updates whenever this is useful. The proposed strategy leads to a non-constant update rate based on, *e.g.*, the measurement accuracy and the motion model uncertainty. Furthermore, experiments with

the AMIGO robot showed how the task-dependent component increases the performance during the RoboCup @Home clean up challenge.

Future work could be to further investigate the task-dependent component of the strategy, *e.g.*, by incorporating state of the art reasoning techniques for triggering updates based on the task and context. In addition, such reasoning strategies can be used to dynamically set the threshold on the information gain. Furthermore, optimal values for the various parameters could be learned from data.

Finally, the relevance of a system like the one proposed increases with the size of the environment and the number of objects present. Collecting and sharing data in much larger environments than the ones considered throughout this work, *e.g.*, a complete floor, could drive interest in the topic of this work. Providing benchmark data on such environments is nontrivial: the strategy steers the robot, the robot's path affects its sensor data and the sensor data influences the strategy. Finding ways to set a benchmark nevertheless is considered to be a very interesting direction for future work.

Acknowledgments The research leading to these results has received funding from the European Union Seventh Framework Program FP7/2007-2013 under Grant agreement No. 248942 RoboEarth. We would like to thank the anonymous reviewers for their useful comments and valuable feedback.

References

Ahmadi, M. & Stone, P. (2005). Continuous area sweeping: A task definition and initial approach. In *Proceedings 12th International Conference on Advanced Robotics, 2005. ICAR'05*, (pp. 316–323). IEEE.

Ahmadi, M. & Stone, P. (2006). A multi-robot system for continuous area sweeping tasks. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, ICRA 2006*, (pp. 1724–1729). IEEE.

Bishop, CM. (2006) Pattern recognition and machine learning. New York: Springer.

Boers, Y. Driessen, H. Bagchi, A. & Mandal, P. (2010). Particle filter based entropy. In *13th Conference on Information Fusion (FUSION)*, 2010 (pp. 1–8). IEEE.

Churchill, W. & Newman, P. (2012). Practice makes perfect? managing and leveraging visual experiences for lifelong navigation. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2012, (pp. 4525–4532). IEEE. doi:10.1109/ICRA.2012.6224596.

Davison A, & Murray, D. (2002) Simultaneous localization and map-building using active vision. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24(7), pp. 865–880. doi:10.1109/TPAMI.2002.1017615.

Ekvall, S., Kragic, D., & Jensfelt, P. (2007). Object detection and mapping for service robot tasks. Robotica: *International Journal of Information, Education and Research in Robotics and Artificial Intelligence*, *25*(2), 175–187.

Elfring, J. van de Molengraft, M. J. G., Janssen, R. J. M, & Steinbuch, M. (2011). Two level world modeling for cooperating robots using a multiple hypotheses filter. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2011 (pp. 815-820). IEEE.

Elfring, J. van den Dries, S. van de Molengraft, R. & Steinbuch, M. (2012). Task-based world model verification. In *ICRA 2012 Workshop on Semantic Perception and Mapping for Knowledge-enabled Service Robotics*.

Elfring, J. van den Dries, S. van de Molengraft, M. & Steinbuch, M. (2013a). Semantic world modeling using probabilistic multiple hypothesis anchoring. *Robotics and Autonomous Systems, 61*(2), pp. 95–105. doi:10.1016/j.robot.2012.11.005.

Elfring, J. Jansen, S. van de Molengraft, R. & Steinbuch, M. (2013b). Active object search exploiting probabilistic object-object relations. In *Proceedings of 2013 RoboCup Symposium*.

Frintrop, S., Rome, E., & Christensen, H. I. (2010). Computational visual attention systems and their cognitive foundations: A survey. *ACM Transactions on Applied Perception (TAP)*, *7*(1), 6.

Hinterstoisser, S. Holzer, S. Cagniart, C. Ilic, S. Konolige, K. Navab, N. & Lepetit, V. (2011). Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes. In *IEEE International Conference on Computer Vision (ICCV)*, 2011, pp. 858–865. IEEE. doi:10.1109/ICCV.2011.6126326.

Joho, D. Senk, M. & Burgard, W. (2011). Learning search heuristics for finding objects in structured environments. *Robotics and Autonomous Systems 59*(5):319–328. doi:10.1016/j.robot.2011.02.012. Special Issue ECMR 2009.

Kalra, N, Stentz, T, & Ferguson, D. (2004). *Hoplites: A market framework for complex tight coordination in multi-agent teams*. DTIC Document: Technical report

Khoshelham, K., & Elberink, S. O. (2012). Accuracy and resolution of kinect depth data for indoor mapping applications. *Sensors*, *12*(2), 1437–1454.

Konolige, K. & Bowman, J. (2009). Towards lifelong visual maps. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS* 2009, (pp. 1156–1163). IEEE. doi:10.1109/IROS.2009.5354121.

Mason, J. & Marthi, B. (2012). An object-based semantic world model for long-term change detection and semantic querying. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE.

Mason, J. Marthi, B. & Parr, R. (2012). Object disappearance for object discovery. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Milford, M. Wyeth, G. (2010). Persistent navigation and mapping using a biologically inspired slam system. *The International Journal of Robotics Research 29*(9):1131–1153. doi:10.1177/0278364909340592.

Pangercic, D. Tenorth, M. Pitzer, B. Beetzh, M. (2012). Semantic object maps for robotic housework—representation, acquisition and use. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Vilamoura.

Rasolzadeh, B. Targhi, A. T. & Eklundh, J. O. (2007). An attentional system combining top-down and bottom-up influences. In *Attention in cognitive systems. Theories and systems from an interdisciplinary viewpoint*, (pp. 123–140). Berlin: Springer

Rusu, R. B. Marton, Z. C. Blodow, N. Holzbach, A. & Beetz, M. (2009). Model-based and learned semantic object labeling in 3D point cloud maps of kitchen environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, St. Louis, MO.

Schultz, D., Burgard, W., Fox, D., & Cremers, A. B. (2003). People tracking with mobile robots using sample-based joint probabilistic data association filters. *The International Journal of Robotics Research*, *22*(2), 99–116.

Sommerlade, E. Reid, I. (2008). Information-theoretic active scene exploration. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR* 2008, (pp. 1–7). IEEE.

Stachniss, C. (2006). Exploration and mapping with mobile robots. PhD thesis, Department of Computer Science, University of Freiburg.

Tenorth M, Kunze L, Jain D, Beetz M (2010) Knowrob-map—knowledge-linked semantic object maps. In *10th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2010, pp. 430–435. doi:10.1109/ICHR.2010.5686350.

Unterholzner, A. Himmelsbach, M. & Wuensche, H. J. (2012). Active perception for autonomous vehicles. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2012, (pp. 1620–1627). doi:10.1109/ICRA.2012.6224879.

Whaite, P. & Ferrie, F. (1997). Autonomous exploration: driven by uncertainty. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19(3), pp. 193–205. doi:10.1109/34.584097.

Xu, T. Chenkov, N. Kuhnlenz, K. & Buss, M. (2009). Autonomous switching of top-down and bottom-up attention selection for vision guided mobile robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS* 2009, (pp. 4009–4014). IEEE. doi:10.1109/IROS.2009.5354674.

Xu, T. Kühnlenz, K. & Buss, M. (2010). Autonomous behavior-based switched top-down and bottom-up visual attention for mobile robots. *IEEE Transactions on Robotics26*(5), 94–954. doi:10.1109/TRO.2010.2062571.

Jos Elfring (1985) received the M.Sc. degree in mechanical engineering from Eindhoven University of technology (TU/e) in 2009. In 2014, he received the Ph.D. degree from TU/e. His Ph.D. thesis was entitled Semantic World Modeling for Autonomous Robots. He was involved in the FP7 RoboEarth project which focuses on sharing knowledge among robots. Besides this, he is a member of the Tech United RoboCup@Home team and he successfully finished the Dutch Institute of Systems and Control (DISC) course program. His research interests are among others, creating, maintaining and sharing semantic world descriptions in robotics.

René van de Molengraft (1963) received the M.Sc. (cum laude) and Ph.D. degrees in Mechanical Engineering from TU/e in 1986 and 1990. In 1991 he fulfilled his military service. Since 1992 he is a staff member of the CST group. Since 2005, he is project leader of the Tech United RoboCup team, which placed second in the 2008-2011 RoboCup Middle Size League world championships and first in 2012. Since 2008, he is an associate editor of IFAC Mechatronics. He is co-author of more than 40 papers in refereed scientific journals and more than 85 papers in refereed scientific proceedings.

Maarten Steinbuch (1960) is professor in Systems and Control and head of the Control Systems Technology group at Eindhoven University of Technology, The Netherlands. He received the M.Sc.and Ph.D. degrees in 1984 and 1989 resp. From 1987-1999 he was with Philips Electronics. He is Editor-in- Chief of IFAC Mechatronics. He is Scientific Director of the Centre of Competence High Tech Systems of the Federation of Dutch Technical Universities, and he is director of the TU/e Graduate Program Automotive Systems. His research interests are in the field of mechatronics, robotics, automotive power trains and control of fusion plasmas.