Observation planning for object search by a mobile robot with uncertain recognition

Matthieu Boussard and Jun Miura

Abstract In order to handle complex tasks in an unknown environment, a robot has to build a map with both free space information and objects type and location. We present an active vision system which first detects candidate objects using global detection mechanism, and later identifies them by moving the robot closer and by using a local recognition mechanism. Having multiple candidates and uncertain algorithm outcomes, we cast the problem as a Markov Decision Process. We exhibit the modelization process, the capability of online solver to quickly find a good action, and finally the implementation on a real robot. This implementation consists of a set of Robot Technology Components (RT components) implementing each part of our method.

Acknowledgements This work is supported by NEDO (New Energy and Industrial Technology Development Organization, Japan) Intelligent RT Software Project.

1 Introduction

In order to perform more complex tasks, robots have to get a better understanding of their environment. We are considering an indoor environment where a robot can interact with humans. One of its most important tasks is to preserve its integrity, and thus has to know where it may move safely. This has led to many SLAM algorithms (Simultaneous Localization and Mapping)[12]. To fulfil complex tasks, the free space map is not sufficient anymore. Instead, the robot has to know what kind of objects are in its environment and should be able to locate them. For a task like serving coffee, the robot has to know what a "coffee cup" is and where it can be

Matthieu Boussard and Jun Miura
Active Intelligent Systems Laboratory, Department of Computer Science and Engineering,
Toyohashi University of Technology, Toyohashi, Japan

found. The problem of searching and locating those objects on the map is called the object search problem.

In this paper we recognize the objects by using computer vision algorithms. To recognize an object with a very high accuracy, the robot needs a high resolution image of it. This can be obtained by either being close to the object or by zooming in on it. This process will be extremely slow if exhaustively applied to a whole room. We thus introduce a two-step object recognition process. In the first step, the robot detects candidate objects using a fast and non robust algorithm where many false positive can remain but with as few false negative as possible. Then the robot will identify them by coming closer and applying a recognition method. Since the computer vision algorithms are uncertain, this leads to a planning problem under uncertainty.

2 Observation planning

The observation planning is an active vision process where a robot has to build a sequence of actions in order to increase its knowledge about the environment. The nature and the effects of those actions depend on the capabilities of the robot. For instance the robot can move, turn, use different kind of sensors, take high definition pictures, etc. It can also has various goals, like SLAM, object recognition, exploration. Our goal is to recognize and locate objects with high accuracy on a freespace map. From only one viewpoint[1], because of its pose, occlusion, lighting, or distance, the recognition may fail. Therefore the robot has to define a set of different viewpoints. To be efficient, the robot should select carefully the viewpoints and we should make an observation policy upon them, which will define the order and the location of every observation action. Many work have been done in observation planning, each having their own assumption. So, we will present existing solutions, then describe more formally our problem and show the associated MDP, and finally detail the implementation on a real robot.

2.1 Related Works

Sjöö et al. [11] present an attention mechanism and methods for depth computation, used to control the zoom level in order to perform a SIFT matching at an accurate distance measure. Here the map is provided before the object search phase, so the robot can compute a navigation graph. The observation plan is based on several greedy methods which may lead to non optimal behaviour. The object detection is made online using two kinds of algorithms, one for adapting the zoom level, and one to recognize objects once a final window is defined.

[1] We define a viewpoint as the robot pose from where it can observe a particular candidate object.

We are interested here in an unknown environment, where only the boundaries of the exploration are is given. In [8], Meger et al. present *Curious George*, a combination between an attention system and a SLAM algorithm. This attention system allows the robot to take high definition pictures of potentially interesting areas, which are used offline to perform the object detection. When the robot finishes to observe an area, it uses a frontier-based exploration to select the best next viewpoint, goes there to perform a new observation step. This process is repeated until the whole room is explored. The principle of alternatively performing move and observation action is used by Shubina and Tsotsos in [10], where they compute a probability of objects presence and probability of object detection using a certain type of recognition algorithm. They later use the function to compute a utility value, the actions being selected in a best-first manner according to it. Experiments show how they can tune this utility function in order to get a desirable behaviour.

Some authors tried not to use greedy approaches to improve the quality of the result. To obtain a long term plan, Aydemir et al. [2] are using a high level planner to select low level strategies to find a target object. Masuzawa and Miura [7] present an online object recognition in an unknown room. The algorithm first detects candidates objects and computes also frontier for exploration. Then, instead of directly selecting the best next viewpoint, their method computes a long term policy. The main issue here is that the authors rely on an ad-hoc world modelization in order to speed up their planning algorithm, but still, since they are performing an exhaustive search, the algorithm is too slow to be solved for larger problem online.

In this paper, we aim at using a general MDP modelization to find a good observation policy. MDPs have been widely studied [9]. They offer a strong theoretical background for action planning under uncertainty, optimality proofs, and many specialized algorithms dealing with planning in large state space problems where the classical algorithms cannot be used. The details of the model and the solver are not fixed and can be changed according to any assumption or improvement, so are the object detection and recognition algorithm.

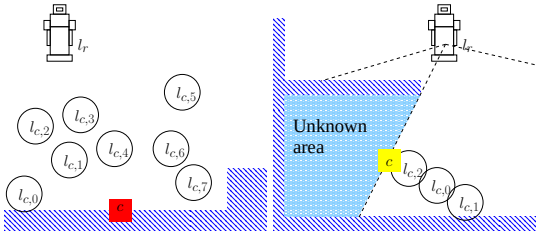
2.2 Problem statement

The robot has to make a map of an unknown room, and to locate objects on this map, see Fig.1. We make the same assumptions as in [7] : the only information provided is the object's models, defined by a (or a set of) picture and the object dimension. Their location and their occurrences are unknown (one object may not even be present in the room). The movement of the robot are assumed certain, i.e. given the current location and the goal location, the probability of reaching the goal is 1, the associated cost is also constant. This is a strong assumption since the actual robot effectors are not perfect, and the computed map can also have some errors.

We use two kinds of object recognition algorithms, one for candidate objects detection ($Detect$), and the other for object recognition ($Ident$). The candidates detection is a color histogram-based detection algorithm. It is fast and can detect object

(a) Objects (b) Simplified room

Fig. 1 Mapping situation



(a) Candidate object c and a set of viewpoints $\{l_{c,i}\}$ (b) Frontier-based candidate detection

Fig. 2 Representation of the environment

from a far distance, but it is not very robust, can return false positive, and cannot make the difference between objects of the same color. The other algorithm is a SIFT matching algorithm [6]. It can recognize very accurately the objects, but it is more time consuming, and needs a good image of the candidate to be matched. Nevertheless, several parameters can impact the detection, like the distance to the object, the object's pose, lighting, occlusions. We assume that we can compute, for every object recognition algorithm, a probability of recognizing one object according to given parameters, like the distance and priors information.

Fig.2 shows the formal description of an environment where the robot has to find objects. The object search process uses a 2-steps approach. Let l_r be the current robot pose. It first uses $Detect$, that returns a set C of candidates objects. Each candidate $c \in C$ has a set $L_c = \{l_{c,i}\}_{i=0...n_c}$ of n_c viewpoints, $l_{c,i} \in \mathbb{R}^2$, from where it can be identified ,see Fig.2(a). The second step uses $Ident$ on candidate c, and $Ident(l_{c,i}) \rightarrow \{F,T\}$ returns the result of the identification. Since this process is uncertain, we note the probability of successful identification by $P_{Ident}(l_{c,i})$. Finally we would like the robot to minimize the overall mission time.

We use an approach similar to frontier-based ones to find candidates in an unobserved area, see Fig.2(b). When updating the model, the frontier between known and unknown environment is computed, and a set of viewpoints to detect objects in this unknown environment is computed. This set behaves exactly like any $\{l_{c,i}\}_{i=0...n_c}$ with two differences : $Detect$ is used instead of $Ident$ and $Detect$ always succeed, $P_{Detect}(l_{c,i}) = 1$

3 Observation Planning problem as a Markov Decision Processes

Markov Decision Process [9] formalizes a sequential decision problem under uncertainty. This process is supposed to be fully observable, i.e. the observed state is the actual state of the system. An MDP is a 4-tuple $\langle S,A,P,R \rangle$, where :

- S is the (finite) set of states,
- A is the (finite) set of actions,

- $P : S \times A \times S \rightarrow [0;1]$ is the transition function,
- $R : S \times A \rightarrow \mathbb{R}$ is the reward function.

A policy π is a function $\pi : S \rightarrow A$ that gives for every state $s \in S$, the action $a \in A$ to perform. Following this policy, we can compute the expected reward $E^{\pi}[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s]$ starting from state s and following the policy π. We use the total expected discounted reward, with a discount factor $0 \leq \gamma < 1$ and r_t the reward at step t. For every state $s \in S$, the unique optimal value function is given by the Bellman equation. $\forall s \in S$:

$$V^*(s) = \min_{a \in A} \left(r(s,a) + \gamma \sum_{s' \in S} p(s,a,s') V^*(s') \right) \tag{1}$$

From the optimal value function, we compute one optimal policy, noted π^*. V^* can be classically computed using Value Iteration, but it is not suited neither for large state space nor for online application. In order to use MDP to solve the observation planning problem, we need to define its four components, namely $\langle S, A, T, R \rangle$.

3.1 States set S

Since the transitions are independent to history (Markov property) the current state has to contain only the information needed to take the decision. Thus a state is composed by :

- the current location $l \in \{l_{c,i}\} \cup l_r$. The decision are taken only on a viewpoint $l_{c,i}$, thus we do not consider any other location in the decision process.
- the set $\big\{\{l_{0,0}, l_{0,1}, \ldots\}, \ldots, \{l_{c,0}, l_{c,1}, \ldots\}\big\}$ representing, for every n candidate object, the visited viewpoints from where an identification has been tried. This allows the robot not to observe twice the same object from the same viewpoint. We only consider a limited history for each object, define by Max_Obs, thus the maximum size of this set is $\binom{|L_c|}{Max\_Obs}$.
- A set of Boolean I_c representing the information whether the candidate c still needs to be checked.

Hence, a state s is defined by : $s = \langle Id, \big\{\{l_{0,0}, \ldots, l_{0,n_0}\}, \ldots, \{l_{c,0}, \ldots l_{c,n_c}\}\big\}, \{I_0, \ldots I_c\}\rangle$. The size of the complete state space is given by the Eq.2 :

$$|S| = |\bigcup_{c \in C} L_c| * 2^{|C|} * \prod_{c \in C} \binom{|L_c|}{Max\_Obs} \tag{2}$$

This set is large and adding an object increases its size exponentially, but it is yet possible to merge some states together. Since a state has to contain all information needed to take the decision, any superfluous information can be forgotten. So, once an object does not need to be checked anymore, only the I_i information is kept. Still,

the state space should not be used completely, and the resolution algorithms should only extend the necessary states (see Sec.4).

3.2 Actions set A

The robot has only one kind of action : a macro action preforming a move action followed by an observation action (*Detect* or *Ident*). The robot selects a viewpoint, goes there and observes the related object. Its outcomes will be described by the transition function. Note that if a viewpoint is not reachable then the robot cannot execute the related macro action. If an object has no viewpoints, it is ignored.

3.3 Transition function P

We assume that the moves of the robot, made by another piece of software, are deterministic. In this paper, we are not considering any uncertainty on candidate detection. So, after applying *Detect* to an unknown area, this area is considered as known, and no detection should be performed again in that area. Furthermore, in order to limit the expansion of the search tree, we state that the agent can only go on a non-visited viewpoint of a non-recognized object. *Ident* is stochastic and the transition function is defined by $P_{Ident}(l_{c,i})$. For instance, in this work we are using SIFT features to recognize the objects. We can estimate $P_{Ident}(l_{c,i})$ by studying the effect of the distance on the recognition probability. From experiments, we can compute the mean and the standard deviation of the number of SIFT matches according to several distance for various objects. With those information we estimate $P_{Ident}(l_{c,i}), \forall l_{c,i}$. Furthermore, we define a maximum number of observations *Max_Obs* per object. Once this limit is reached, the object shouldn't be checked anymore. Fig.3 shows this process. We consider that the observation succeeds if a candidate is identified as being the object and fails when it cannot decide (Unknown). It is impossible to apply an observation action on an identified object and also for those that have been observed *Max_Obs* times.

3.4 Reward and Cost function R

The cost function is given by the travelling time plus the time of the image processing. As discussed in the conclusion, optimizing the mission time and the number of identified objects is a complex multicriteria optimization problem.

4 Algorithms

The selection of an algorithm to solve the object search problem has been studied in [4]. In this paper, the authors dealt online (less than 1sec) with problems having $|C| = 8, Max\_Obs = 2, |L_c| = 15$ which is reasonably large as an object search problem. The authors conclude that *LRTDP* [3], an heuristic algorithm, is the best suited for solving the MDP for object search with a limited number of objects. The general process of the system in the object search context is shown in the Fig.4. It first gathers information about the environment, updates if needed the related MDP model, then runs a planning algorithm to select the action to perform and finally executes it. Then the process is repeated until the end of the mission, defined by having no more candidate object to identify. The system is thus able to adapt to any change in the environment or with any information gathered on candidates objects before every decision step.

5 System Implementation

The implementation has been done using OpenRTM-AIST [1] as an implementation of RT-middelware. It allows the development of software components (RTC) and provides an interface to dynamically compose and plug a set of components. The complete system is presented Fig.5. The SLAM RTC(2) maintains a freespace map of the explored area. The main control is made by the Planner RTC(7). It can request a up-to-date MDP from the Modelizer RTC(6), solve it, and send the commands to the other components. If it is a move action, the Global pathplanner RTC(5) computes a global route sent to the local Pathplanner RTC(3), which react to unforeseen objects and avoids moving obstacles. If it is an observation action (*Detect* or *Ident*) a request is sent to the Observation controller RTC(4), which will send an image processing request to the related RTC, manage the results and send update to the Modelizer. The rest of the RTCs (1) controls the hardware (Camera, Robot, URG, buffers).

The candidate detection is the first step of the object search upon which the policy is computed. If this step is not reliable the quality of the computed plan will be poor, even it is optimal. As a *Detect* function we are using a Color Histogram search.

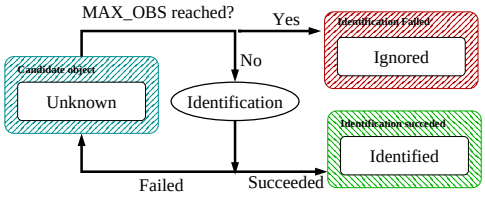


Fig. 3 Observation outcomes

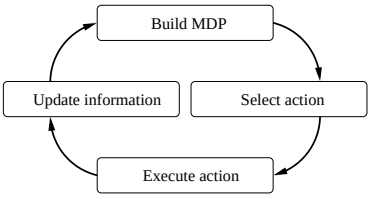


Fig. 4 Global execution loop

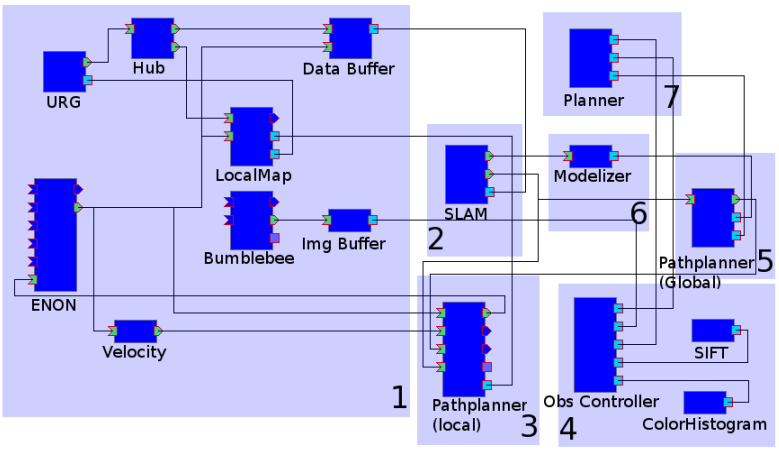
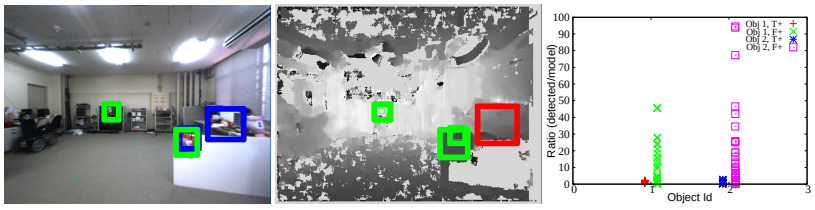


Fig. 5 Global RT components system



(a) Color detection (b) Verification (c) Surface ratio for true/false positive (T+/F+)

Fig. 6 False positive filter

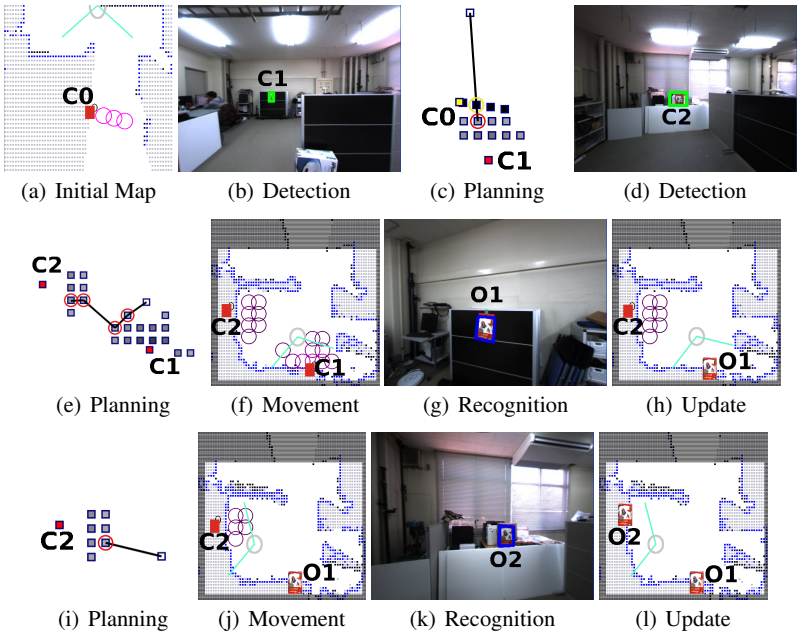
Since it is not robust, it will either return many false positive (too sensitive), or miss real candidate objects (not sensitive enough). Since we don't want to miss any real candidate, *Detect* should remain *"sensitive enough"*. Also we get the depth information from the stereo camera, we can follow the method proposed in [5]. When an object is detected, we compute the mean of its depth values. We have its world coordinate x, y, z and its model (previously given), so we can project the model at the detected location x, y, z in the image. We now have a candidate box, and a model box in the image, the algorithm can then compute the ratio between those two areas. Fig.6(c) shows for two different objects this ratio when computed with true positive (T+) and for false positive (F+). To reject incoherent objects, we keep results with a ratio in $[0.2; 4]$, where all the true positive ratio value are. Fig.6 shows a false positive elimination where one object (the right-most one) is removed while keeping the true positive[2].

[2] note that one false positive remains, it will be removed later by SIFT matching

6 Experimental Result

We use a Fujitsu ENON robot with a Point grey bumblebee stereo camera, and a Hokuyo URG laser range finder. We also embed a laptop computer connected to the lab network using a wireless connection. On the LAN we can use desktop computers with standard GPU cards (Geforce 8600GTS).

Fig.7 shows a global mission execution, within given boundaries (the brighter area in Fig.7(f)). The robot is in its starting position. It finds an unknown area from the free space map. The frontier between known an unknown area c_0 and the location from where *Detect* should be performed is computed (a). A *Detect* action is also performed from the starting position, returning one candidate object c_1 (b). With this information the MDP is computed, solved, and the next action is selected (go and detect candidate in the unknown area) (c). The action is performed by the robot by first going to the selected viewpoint. Then, since the candidate is an unknown area, the *Detect* action is performed and returned a new candidate c_2 (d). A new MDP is build and solved (e). The robot moves to the selected location (f). The candidate is really a candidate object, *Ident* is applied, returning the pose and the type of the object o_1 (g). This information is used to update the map (h). One candidate remains so the MDP is build, solved (i), the action is executed (j). The object o_2 is identified (k), and the map is updated (l). Finally, no more candidate nor unknown area remains, the mission is over.



(a) Initial Map (b) Detection (c) Planning (d) Detection

(e) Planning (f) Movement (g) Recognition (h) Update

(i) Planning (j) Movement (k) Recognition (l) Update

Fig. 7 Example of semantic mapping

7 Conclusion

We show an MDP model of the object search problem using a mobile robot. By computing a policy for candidate detection, object recognition and exploration the robot can efficiently build this map. The MDP model can be solved using *LRTDP*, an online algorithm with a good anytime behaviour. Our model is designed for our specific hardware limitation, nevertheless it can be adapted regarding to new hardware capabilities, like the presence of zooming capabilities or 3D scanner. Furthermore, the planner has been developed as a component using RT-middleware, and thus can later easily be used on different type of hardware. The current candidate detection is made at discrete time decided by the policy computed from MDP. In the future, we would like to continuously integrate candidate detection. Also this paper focused only on optimizing the mission time ; we would like to investigate how to include the number of recognized objects in the decision process leading to a multi-objective optimization problem. As we would like the robot to evolve among humans we cannot consider the environment as static, maintaining an up to date map is also a challenging perspective.

References

1. Ando, N., Suehiro, T., Kotoku, T.: A software platform for component based rt-system development: Openrtm-aist. In: SIMPAR'08, pp. 87–98 (2008)
2. Aydemir, A., Sjöö, K., Folkesson, J., Pronobis, A., Jensfelt, P.: Search in the real world: Active visual object search based on spatial relations. In: ICRA 2011. IEEE, Shanghai, China (2011)
3. Bonet, B., Geffner, H.: Labeled RTDP: Improving the convergence of real-time dynamic programming. In: E. Giunchiglia, N. Muscettola, D.S. Nau (eds.) ICAPS, pp. 12–31. AAAI (2003)
4. Boussard, M., Miura, J.: Observation planning with on-line algorithms and GPU heuristic computation. In: ICAPS-10 Workshop on Planning and Scheduling Under Uncertainty (2010)
5. Helmer, S., Lowe, D.G.: Using stereo for object recognition. In: ICRA 2010, pp. 3121–3127. IEEE (2010)
6. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision **60**(2), 91–110 (2004)
7. Masuzawa, H., Miura, J.: Observation planning for environment information summarization with deadlines. In: IROS 2010, pp. 30–36. Taipei, Taiwan (2010)
8. Meger, D., Muja, M., Helmer, S., Gupta, A., Gamroth, C., Hoffman, T., Baumann, M., Southey, T., Fazli, P., Wohlkinger, W., Viswanathan, P., Little, J.J., Lowe, D.G., Orwell, J.: Curious george: An integrated visual search platform. In: CRV, pp. 107–114. IEEE Computer Society, Washington, DC, USA (2010)
9. Puterman, M.L.: Markov Decision Processes : Discrete Stochastic Dynamic Programming. Wiley (2005)
10. Shubina, K., Tsotsos, J.K.: Visual search for an object in a 3D environment using a mobile robot. Comput. Vis. Image Underst. **114**, 535–547 (2010)
11. Sjöö, K., Gálvez-López, D., Paul, C., Jensfelt, P., Kragic, D.: Object search and localization for an indoor mobile robot. Journal of Computing and Information Technology, Special Issue on Advanced Mobile Robotics **17**(1) (2009)
12. Thrun, S., Burgard, W., Fox, D.: Probabilistic Robotics (Intelligent Robotics and Autonomous Agents). The MIT Press (2005)