Information Theory to Optimize Plans for Intelligent Robotic Arm Search of Known Objects

Roger Ballard **Timothy Atkinson** **Taher Patanwala**
Zubin Kadva **Zemeng Wang** **Chimiao Wang**
Zongqiao Liu **Rajaa Rahil** **Marius C. Silaghi**

School of Computing
Florida Institute of Technology

ABSTRACT

We propose an approach based on information theory and probabilistic models to plan optimized search processes of known objects by intelligent eye in hand robotic arms. Searching and reaching for a known object (a tool) in one's office is an operation that humans perform frequently in their daily activities. Intelligent robotic arms also encounter this problem in the various applications in which they are expected to serve.

The problem suffers from uncertainties coming from the lack of information about the position of the object, from noisy sensors, imperfect models of the target object, imperfect models of the environment, and from approximations in computations. The use of information theoretic and probabilistic models helps us to mitigate at least a few of these challenges, approaching optimality for this important task.

Author Keywords

Eye-in-arm, Search, Information Theory

INTRODUCTION

Robotic arms are traditionally used as automates that follow predefined trajectories, but recently they are combined with sensors to provide more intelligent functions such as abilities to open doors and grasp unknown objects. Here we address a seemingly more mundane problem of locating a known object in a partially known and bounded environment. In our problem we assume that the robotic arm has a single camera positioned in the arm. The problem is in fact challenging if we consider the need to optimize the number of movements, speed of localization, and certainty of result [5]. The lack of stereo vision can be compensated by taking pictures from multiple positions of the end effector, an additional challenge that adds up to the aforementioned problems.

Among motivating applications we list manufacturing lines, tool search by robotic arms on autonomous vehicles or in space, and medical robotic arms trying to find the right place to perform a blood drawing.

One of the commonly used approaches to path planning is based on search in the configuration space, where the arm is avoiding collision with elements in the environment, as well as with itself. In this work the problem of path planning including collision with other objects and among its own segments, is assumed to be solved in a different module, not discussed here. In our own experiments, path planning is performed in the robot driver. Here we are concerned about the planning of a sequence of movements that maximizes the certainty of the localization of the searched object in a minimal number of image captures.

The environment, with potential positions of the object, can be segmented in partly overlapping areas, each of them being a possible view from the camera in the hand of the robot. The search proceeds by selecting the views to acquire and analyze next. The order is given by a heuristic for maximizing the information about the location of the searched object. The information concerning what is known is maintained in a belief map. With each picture, the belief changes and the next picture point of view is planned such as to increase this information. After covering some of the background and related work in the next section, we continue by describing formally the problem and the proposed heuristics. We end with discussions and conclusions.

BACKGROUND

Planning problems have been addressed by robotics research for multiple decades. An important evolution of this research area consisted in the adoption of probabilistic models to represent in a scientific way the uncertainty existing in most real problems.

The source of uncertainly is constituted jointly by ignorance (e.g., concerning exact position of objects, luminosity and shape) and by the high computational complexity of known algorithms to access and process data. The ignorance is manifested not only in the lack of data but also in the incomplete modeling of physical phenomena, or in the approximations selected for modeling them.

Several other approaches had been proposed to address uncertainty, including default logic, fudge factors and fuzzy logic, but the community has largely concentrated on probabilistic approaches, which are accepted as being better scientifically founded among alternatives.

Probabilistic models generally use "statements" as ontological commitments (nature of reality) and probabilities as "epistemological" commitments (possible states of knowledge),

interpretable as "degrees of belief" or as "frequency", potentially describing objective properties of the world. The basic objects/statements are represented using random variables. States of the world correspond to assignments of values to these random variables.

The use of probabilistic models does not automatically reduce errors from uncertainty in reasoning except in as much as the probabilistic models do address that particular uncertainty. For example, most probabilistic models still make significant approximations concerning the actual relations (or absence of relation) between facts. Another common approximation is in discretizing time and space, and studies have addressed the convergence of these approximation towards their continuous counterparts [1]. Probabilities can be modeled and learned with technologies such as artificial neural networks and Bayesian networks.

Bayesian Nets

One of the most influential techniques for creating probabilistic models of phenomena is the Bayesian Network. The Bayesian Networks are graphical probabilistic models where statements (random variables) are depicted with nodes and conditional dependence relations between these concepts are displayed with directed arcs. The strength of Bayesian Networks come from the fact that not all dependence relations have to be depicted, since some of them can be inferred from others. In general, a random variable does not need to be linked to a second variable if they are independent given variables on already specified paths between them. The illustration in Figure 1 shows a simplified belief network for detecting known objects based on signals from camera interpreted as shape, color, and texture [9], in the presence of various orientations and lighting conditions:

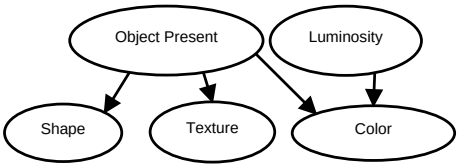
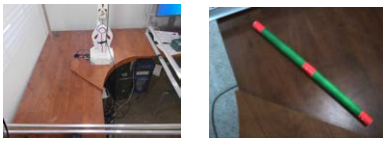


Figure 1. Sketch Belief Network showing potential conditional dependence assumptions between variables involved in the detection of an Object, without showing conditional probability tables.

For planning problems in environments with uncertainty in sensors or actions outcome, an alternative to continuous replanning is to build *contingent plans* or *policies*. A policy is a mapping from each belief state of the agent into a plan to be executed in that state.

Addressing the robotic eye-in-arm search problems with Partially Observable Markov Decision Processes (POMDPs) has been first suggested in [8]. A POMDP $(\Sigma, A, T, R, \Omega, O, \gamma)$ is described by a set Σ of states, a set A of actions, a set T of conditional transition probabilities between states (given performed actions), a reward function $R : \Sigma \times A \rightarrow \mathbb{R}$, a set Ω of possible observations, a set O of conditional observation probabilities and a discount factor γ. Several algorithms were proposed for efficiently solving POMDPs, such as value iteration, policy iteration, point-based value iteration [3, 2, 4]. With POMDPs the goal is to maximize the expected utility, defined as the scalar (dot) product between belief (vector of probabilities for each state) and the utility of the corresponding states.



a b

Figure 2. Sample Search Problem: (a) Space and (b) Object

Information Theory and Decision Trees

In this work we propose to address the problem using a different approach, namely information theoretical decision tree learning (ID3) [6]. Decision trees are a technology largely used for learning and classifying one concept. A decision tree consists of a tree data structure where each node represents a question and each branch is a possible answer to that question. Leaves stand for answers to the classification problem represented by the tree.

It is assumed that a user classifying a sample traverses the tree from its root towards one leaf, at each node asking the corresponding question represented by it, and once its answer is obtained, following on the corresponding branch. The leaf reached in this way tells the classification result for the given sample.

ID3

The question of building decision trees that minimize the number of questions needed for classifying one sample has been heavily researched in the past. It has to be noticed that the number of possible decision trees is exponential and their enumeration is generally out of question. A heuristic commonly proposed for building such trees given available training data is based on information theory. The idea is to ask a question that reduces the expected entropy of the data in a training sample.

Algorithm 1: Pseudocode for Object Search

```
1 procedure ObjectSearch() do
2     b = Init_Belief();
3     for ( ; ; ) do
4         if Object identified and reached, or space exhausted
            then
5             return(b);
6         q = Best_Next_Viewpoint(b);
7         capture_image(q);
8         Update_Belief(b,q);
```

THE ROBOTIC ARM EYE IN HAND SEARCH PROBLEM

A robotic arm controls an area within which it searches for a known object. For example, the area in our experiments is depicted in Figure 2.a and the searched object is shown in Figure 2.b. The algorithm used in [9] for this problem has similarities with the approach proposed here and a pseudocode for it that is reusable here is given in Algorithm 1.

The problem is formally represented as a lattice $x_{i,j}$ of possible positions for the object (assumed to lay on a flat table). The object can occupy a set of several neighboring elements of the lattice, function of its position and orientation. The object may also not be present in the environment. The ratio of the lattice is given by the desired resolution of the detection. Experiments in this work use a pixel-level representation while some of the previous approaches were aiming at resolutions as rough as 10 cm [9, 7].

Originally the probability distribution of the position of the object is uniform across the lattice. The lattice can be analyzed from any point out of a set of points of observation, each of them covering a different quadrilateral's intersection with a subset in the lattice (function of the orientation, height, and location of the camera). At each step, one of the possible points of observation is selected such as to maximize the expected amount of information gathered about the position of the object. This solution is related to ID3.

We assume that, given uncertainties, capturing the object from immediate positions (1cm) corresponds to fully localizing it. Given that binary search is known to efficiently scan an organized data set, it can be expected that an efficient search will start by taking remote snapshots from higher altitude and converge towards close-up captures.

Note that if the camera would allow for zooming, then the same snapshot could theoretically be taken from multiple positions. However, our camera does not have zooming capabilities, and we will not simulate them by magnifying image areas, as this is assumed to yield lower quality detection.

SOLUTION BASED ON ID3

We note that the problem can be modeled as a learning/classification problem where we are trying to detect the position of the searched object by asking repeated questions. Each snapshot taking can be seen as such a question. A relevant algorithm that tries to minimize the number of asked questions is ID3, where each step minimizes the entropy.

Adapted ID3

The idea is to measure the entropy of each belief distribution $b_t(s)$ representing the knowledge at a given time t about the possible exhaustive and mutually exclusive states s using Shannon's formula $H(b_t) = \sum_s -b_t(s) \log_2 b_t(s)$.

The informational gain expected from a given question q at time t can be computed as the difference between the entropy before asking the question and the expected entropy after asking the question. The belief at time t is denoted b_t and each of the possible beliefs obtainable after learning the answer i if question q is posed at this moment, is denoted $b_{t+1}^{q,i}$. Assuming each answer i is expected to question q with a probability $p(b_t, q, i)$, as it can be computed from the belief b_t, the expected information gain from question q is:

$$G_{q,t} = H(b_t) - \sum_i p(b_t, q, i) H(b_{t+1}^{q,i})$$

.

Therefore, the question that the ID3 heuristic recommends to be asked at moment t is:

$$q(t) = \underset{q}{\operatorname{argmax}} \left(H(b_t) - \sum_i p(b_t, q, i) H(b_{t+1}^{q,i}) \right)$$

.

In common learning problems, the classification outcome (belief b_t) is given by some set of samples in a training set that are matching questions asked so far, and is represented via counts of *positive* and *negative* training samples. Such a decision tree is first built once with large training sets of samples and later reused to classify new incoming samples.

In this research, we do not pre-compute and store such trees due to their sheer size given the number of possible questions. Rather, we recompute the tree on the fly by predicting $p(b_t, q, i)$ using b_t.

The aforementioned general procedure being proposed is summarized in the generic pseudocode of Algorithm 2.

Algorithm 2: Generic Pseudocode for Information Theoretical Search

```
1  procedure Init_Belief() do
2      b = lattice representing the whole search space, each
         node being the probability of the object center (or its
         parts) being in the corresponding area;
3      return b;
4  procedure Best_Next_Viewpoint(b) do
5      V = set of N views sampled with chosen distribution
         across possible views;
6      B = evaluate(V,b); // e.g., set of views with expected
         belief maps and information gain;
7      v = best of B;
8      for (;;) do
9          if (timeout) then
10             return (v,B(v),H(B(v)));
11         V = resample N views based on B;
12         B = evaluate(V,b); // e.g., based on expected
             information gain;
13         v = best of B;
```

Belief Updates

Each snapshot taking function updates the belief. The effect of the analysis of the snapshot capture at position and orientation (p, o) which observes the set of states (subset of the lattice with possible positions) given by a function $View(p, o)$ is a belief function update $\omega(j)$, with the properties that

$$\omega(j) : View(p, o) \rightarrow [0, 1]$$

and that

$$\sum_{j \in View(p,o)} \omega(j) \leq 1$$

.

If the belief before the snapshot is b', the belief update is a function u:

$$b = u(b', \omega)$$

One of the possible ways to integrate the new observation is incremental update:

$$b_\gamma(j) = \begin{cases} \gamma\omega(j) + (1-\gamma)b'(j) & \text{if } j \in View(p,o) \\ \frac{1-\sum_{j\in View(p,o)}(b_\gamma(j))}{\sum_{j\notin View(p,o)} b'(j)} * b'(j) & \text{otherwise} \end{cases} \quad (1)$$

In the above approach, a learning factor γ is used for updating the belief about recently observed areas, modeling the classification error probability, and this factor can decrease in time as the snapshots are taken zooming closer to the search area. The belief about areas not observed in this round is updated by normalization to make the total belief integrate to 1, which is done by multiplying it with the factor $\frac{1-\sum_{j\in View(p,o)}(b_\gamma(j))}{\sum_{j\notin View(p,o)} b'(j)}$.

Aggregated History

An alternative approach is to gather all the features from images captures so far and to use a single classifier (e.g. ANN) to generate a posterior probability for each location. A third approach is to keep the classification given by the snapshot taken from the position from which the area has the largest projection in the image. The approach taken in one of the experiments is to multiply the new object detection probability with the previous belief and with an observation probability modeling the uncertainty of the image analysis.

The next question

In a given state of the search, we can evaluate the next question to ask by enumerating a subset of the candidate questions (positions and orientations). For each candidate question, there are two possible outcomes:
- object found with probability limited by the zoom level (size of the corresponding area in snapshot)
- object not found

The probability of each of these outcomes happening is based on the current probability mass assigned to the corresponding region. Therefore the utility of each question can be computed as the expected utility along the aforementioned branches.

Stochastic Search of Questions

The set of positions and orientations considered for the next snapshot is explored using a stochastic search method. A set of P capture positions are distributed throughout the workspace. For example, one can use a density that is higher in the height of the last snapshot than at other heights. At each position, a number of O orientations towards the searched surface are considered.

The information gain is estimated for each of these points. During a set of k rounds repeating the above procedure, the positions are redistributed into 3D cubic cells centered at previous points, proportionally with the information gain found at that position. The size of the cells halves with each round. The orientations at each new point are similarly sampled in rectangles centered in previous directions at the previous point, and distributed proportionally with the corresponding information gains.

Computational Considerations

A trade-off exists between the number of positions P and orientations O to consider at each round and the number of states s (surface resolution) where belief of object presence is evaluated. Potential ways to alleviate this complexity is to start with a lower resolution and to increase it with each snapshot being taken, as a function of the height of the snapshot, such that the maximum resolution is obtained when the height is at the minimum accepted value.

Look-ahead

Just as with decision trees, the algorithm needs not be limited to evaluating only the next question at a time. Instead, just as with decision trees, game playing, and POMDP solvers, policies consisting in sequences of multiple actions can be evaluated before deciding the next questions. As with general look-ahead techniques, cheap evaluations/predictions at a given search state can make a good trade-off with deeper look-ahead steps (see Algorithm 3).

Algorithm 3: Look-ahead in evaluation

```
1  procedure evaluate(V:view, b: belief) do
2      for q in V do
3          for i as outcome of snapshot q do
4              b'_{q,i} = update(b,q,i);
5              if (not deepest level) then
6                  (q', b'_{q,i}, H') = Best_Next_Viewpoint(b'_{q,i});
7          H=aggregate_expected_gain(q) over all i;
8      return maximum (q,b,H);
```

Algorithm 4: Pseudocode for Greedy Hierarchical Search

```
1  procedure Init_Belief() do
2      b = an empty tree with root representing whole search
         space and an empty ordered queue with the leaves of
         the tree;
3      return b;
4  procedure Best_Next_Viewpoint(b) do
5      for (;;) do
6          v = most likely node in b;
7          if (v is known and not minimal) then
8              split v in overlapping views;
9              insert splits in b;
10         return v;
```

DISCUSSION

A solution that was not based on a lattice but on a dynamically expanding belief tree enforcing a hierarchical search

was proposed in [9] and is described in Algorithm 4. Its main drawbacks with respect to the lattice approach used here and in the POMDP model [7] is that views location cannot be flexibly optimized in case objects are detected in border areas, or are so well identified that multiple levels of splitting can be jumped at once.

View Ordering Heuristics

The ID3 approach is an alternative to the common approach to action planning which is based on decision theory associating a reward with various states. There the state has to describe knowledge [7]. Alternative heuristics to the ID3 are possible, such as where each action (snapshot taking) is expected to add extra information, and the reward is defined as:

- the mode of the belief function $b(s)$,
- the variance of the obtained belief function $b(s)$.

There can also be alternative ways of sampling the search space for candidate next questions. Instead of the described version of stochastic search, one can use:

- a kind of beam search where predefined transitions are tried and only the N most promising ones are extended further.
- simulated annealing transitions from each out of the set of N positions.

IMPLEMENTATIONS AND EXPERIMENTS

Two implementations are detailed together with their evaluation: baseline and optimized search.

Baseline

The baseline algorithm makes the assumption that in a blind search (with no extra information), all optimal algorithms become equal and it is only after a detection that the search heuristic makes a difference. It further posits that the location of the object may be gleaned from the image directly.

Algorithm Initialization

The Bayesian Network model of the object and its colors described in Figure 3 is first trained from a set of sample images. A sequence of viewing angles is built where each next view is selected to maximize marginal information gain assuming lack of object detection with previous views.

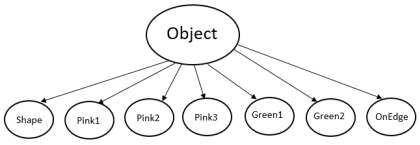


Figure 3. Baseline belief network.

Image processing

After each snapshot is taken, this method calculates both where the object could be located in the image and the probability that the object is indeed present.

First the image is scanned for all red and green pixels based on the Hue and Saturation from HSV. These pixels are set in a red and green binary image. Contours are found and their size is estimated based on their minimal enclosing rectangle. Rectangles with too small of an area for the respective color are pruned. One more sensor is added, an `'onEdge'` detection, which evaluates the statement, "Is this blob on the edge of the captured picture?". It is calculated by projecting the corners of the rectangle back onto the image and seeing if any of them falls off the image. We call each rotated rectangle with properties like color and `onEdge`, blobs.

An attempt is then made to pair multiple blobs together forming Red-Green-Red chains or Green-Green chains referred to as *detections*. In order for red to combine to a green, the two colors must be next to each other and match orientation (orientation can be flipped by 180 degrees as detections cannot tell front from back). Green-green is allowed based on the assumption that there was an undetected red. Unlike red-green pairings, green-green pairings must have a space between them. Not only must the orientation of the green blobs match each other, but the space is treated as an invisible rectangle whose orientation must also match. The one exception to matching orientation is when an object is `'onEdge'`, in which case we ignore that object's orientation (for example detecting the edge of a square will look like a triangle in the image, and the resulting blob will have an orientation radically off from where it really is at). If a matching only occurs due to the `'onEdge'` flag, then that object is considered to not have the proper `'shape'`.

This could theoretically lead to a detection of Green-Green-Green or Red-Green-Red-Green-Red-Green (or longer)– these detections are broken down so that each detection has no more than two greens and no more than three reds following the rules noted above. Each of these detections then have their location estimated based on the camera and its position.

Next, if the robot's position is not optimal for estimating the object's location; a new image needs to be taken from a better position. In order to place the new image in the queue, the algorithm calculates the object occurrence probability based on the Bayesian network.

The Bayesian network responds to the blobs that were trained together. If there is only one green, then it is `green1`, two greens will result in `green1` and `green2`. The first red detected will be `red1`, the second will be `red2` and the third `red3`. The one exception is if detection links the blobs with the pattern green-red-green and instead the evidence will be `green1`, `red2`, `green2` as the `red2` represents the middle red of the tube.

Object Localization

To simplify the detection of the object location, the robot's camera is oriented along the robot's waist such that an x offset in the image can be directly attributed to a shift of the waist in the robot's configuration space.

The procedure is shown in Algorithm 5. This algorithm scores matches by their probability and can deal with objects not located on the table. However, it has a several weaknesses. Its biggest problem is that its initial search success depends on serendipity. Due to the involved computational

Algorithm 5: Detailed Baseline

```
1  procedure Init_Belief() do
2      b = a set of potential view angles with initial weight of
         .5;
3      img = null;
4  procedure Best_Next_Viewpoint(b) do
5      while (b ≠ ∅) do
6          c = first item in queue send c to the robot;
7          if (img ≠ null) then
8              blobs = processImage(img);
9              dets = processBlobs(blobs);
10             score all detections;
11             add all detections to b based on object
                 occurrence probability;
12         return image from last command c to robot;
```

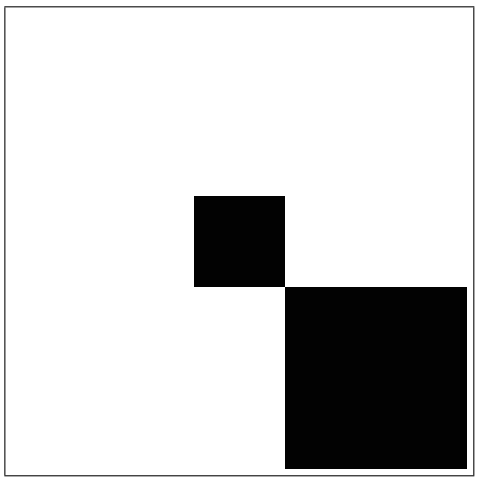


Figure 4. Initial belief state, where black regions have 0 probability and white regions have the maximum probability over the distribution.

complexity, the selected initial sequence may not be optimal. For example, in a first version, it took 48 moves to search the space in a worse case where the object is in the very last location and there are no purposeful distraction objects (obviously every false match object will take an extra move). A slight improvement to the used approximation, and the search could be reduced to 23 moves for a worse case scenario.

Optimized Implementation

In this approach, significantly more effort was put into the planning portion of the algorithm than the computer vision portion of the algorithm. In order to simplify the computer vision task, the vision problem was reduced from "find the given object on the table" to "find a pink blob on the table." By finding the location of the three pink blobs on the object, the exact location and orientation of the object can be determined.

Representing Belief States

For this problem, the $1m \times 1m$ table being searched was discretized into $1cm \times 1cm$ regions, resulting in a 100×100 region grid. A belief state is simply a probability distribution of the location of the target pink blob over this grid. An example initial belief state is shown in Figure 4

Evaluating Belief States

In order to be able to evaluate potential actions for planning, we need to be able to evaluate the utility of their resulting belief states. We experimentally compare two methods for evaluating a belief state: entropy and weighted standard deviation.

For the entropy measure, the entropy of a belief state is defined to be the entropy of the probability distribution:

$$H(B) = \sum_{n=1}^{|B|} -B[n] \log_2 B[n] \tag{2}$$

Where $|B|$ is the number of regions in the belief state, and $B[n]$ is the belief that the target is in region n. From a decision theoretic perspective it can be considered that the utility of the belief state is simply taken to be the negative of its entropy:

$$U(B) = -H(B) \tag{3}$$

For the weighted standard deviation measure, the weighted standard deviation of a distribution with locations associated with the domain of the distribution is defined as follows:

$$center(B) = \sum_{n=1}^{|B|} B(n) loc[n] \tag{4}$$

$$variance(B) = \sum_{n=1}^{|B|} d(loc[n], center(B))^2 \tag{5}$$

$$stdev(B) = \sqrt{variance(B)} \tag{6}$$

Where $|B|$ is the number of regions in the belief state, B(n) is the belief that the target is in region n, loc[n] is the location of region n, and d is the distance metric used. The utility of the belief state is simply taken to be the negative of its weighted standard deviation:

$$U(B) = -stdev(B) \tag{7}$$

The motivation for the weighted standard deviation measure is the fact that the entropy measure does not take into account the spatial information of the belief state. Due to this limitation, using the entropy measure (without look-ahead) can lead to plans that leave unexplored regions interleaved with explored regions, potentially leading to a greater overall number of steps required to find the target.

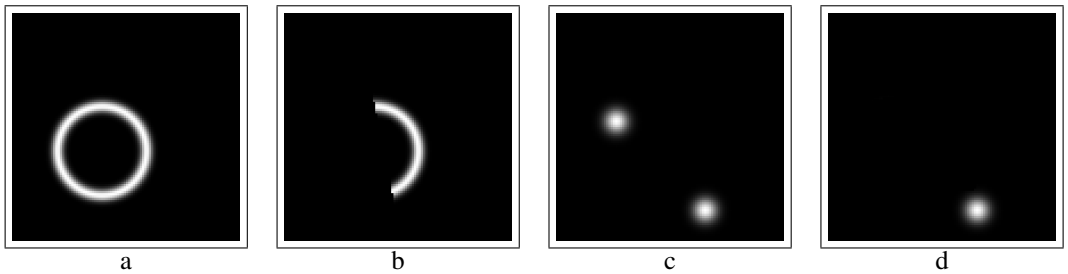
Updating Beliefs

The current belief state can be maintained at all times and can be updated using Bayesian inference. Given a prior belief state B and a piece of evidence E, the following two rules are sufficient to produce a new belief state B', such that:

$$\forall n \in \{1..N\} : B'(n) = B(n|E) \tag{8}$$

Algorithm 6: Pseudocode for Action Space Exploration

```
mostPromising = a new empty list of regions;
insert a region representing t*he whole configuration space into mostPromising;
for (int depth = 0; depth < search_depth; depth++) do
    for (int dim =0; dim < dimensionality; dim++) do
        newMostPromising = a new empty list of regions;
        split each box into mostPromising into three sub-boxes, splitting on the dimension dim;
        insert each sub-box into newMostPromising;
        remove any invalid configurations from newMostPromising;
        //Some configuration space coordinates are invalid in the workspace;
        filter newMostPromising to the beam_width boxes with the highest utility;
        replace mostPromising with newMostPromising;
return best element of mostPromising;
```



a b c d

Figure 5. a) Initial belief state for finding the second point. This takes the form of a ring centered on the first point b) The effect of using already-taken photographs to update the belief state for the second point before even beginning the search c) Initial belief state for finding the third point. This takes the form of two radial Gaussians based on the found location of the first two points d) The effect of using already-taken photographs to update the belief state for the third point before even beginning the search

Position 1 Position 2 Position 3

Figure 6. Location of the target object in three position.

- **Target not Detected in Image:** If the target is not detected in the image, then the update rule is given as follows:

$$B'[n] = \begin{cases} \alpha * B[n] * false\_neg\_prob & \text{if } visible(n, c) \\ \alpha * B[n] & \text{otherwise} \end{cases} \quad (9)$$

  Where c is the configuration of the robot arm that the image was taken from, and $visible(n, c)$ is true iff region n is visible in configuration c. A normalization factor, α, is also employed.

- **Target Detected in Image:** If the target is detected in the image, then the update rule is given as follows:

$$\begin{aligned} B'[n] \quad = \quad & \alpha * B[n] * \\ & (((1 - false\_positive\_prob) \\ & \quad * kernel(d(loc[n], detect\_loc))) \\ & + false\_positive\_prob) \end{aligned} \quad (10)$$

  Where $detect\_loc$ is the location of the detection mapped to the table, and $kernel$ is the kernel function representing the uncertainty in the camera. (In this implementation, the kernel was a Gaussian distribution with σ proportional to the distance of the camera from the target and the distance in the image of the target from the center of the image, to model increasing uncertainty at further distances and increasing uncertainty around the edges of the image.)

Evaluating Actions

The expected utility of an action (an image taken in a certain configuration) can be naturally defined in terms of the utility of the potential belief states it could result in:

$$U(a, B) = \sum_{loc=1}^{|B|} B(loc)U(\textit{resultant\_belief}(B, a, loc)) \quad (11)$$

Where a is the action taken, loc is the true location of the target, B is the prior belief state, and *resultant_belief* is defined according to the update rules in the Updating Beliefs section.

However, this direct calculation is prohibitively expensive. Instead, we estimate the utility of an action by sampling several locations from the prior belief distribution and averaging the resultant utilities for each sample.

Exploring the Action Space

We can treat our configuration space as an n-dimensional box, and explore the space by repeatedly dividing the space into sub-boxes. We define the utility of a box in the configuration space to be the maximum utility of any point inside that box. Making the observation that the utility function is locally smooth in the configuration space, we can approximate the utility of a box by evaluating the utility of the center of that box. As we are subdiving, we can take the most promising boxes at any depth (those with the highest approximated utility) and expand them further. We can repeat this process until we have subdivided the boxes into several suitably small volumes, thus selectively exploring the search space. This is outlined more formally in Algorithm 6.

Finding a Point

We can find a single point by repeatedly determining the best next action to take, taking it, and updating our belief with the evidence we get. In order to determine when to stop and declare the point found, a threshold value on the utility of our belief state is used. For the entropy measure, this threshold is 5 bits. For the weighted standard deviation measure, this threshold is 2.5cm. In general, taking pictures with successful detection at 1cm distance from the object important features can also be considered an acceptable termination condition.

Finding the Object

Given the above algorithm for finding a single point, we can apply this algorithm to find the object's position and orientation. For the first point, we initialize the belief state uniformly randomly, except for certain invalid regions, such as not on the table or inside the base of the robot, as shown in Figure 4.[1] For the second point, since we know the radius of the object and the position of the first point, we initialize the belief state to be a ring of that radius around the first found point, as shown in Figure 5.a. For the third point, since we know the radius of the object and the position of the first two points, we initialize the belief state to be two Gaussian distributions, as shown in Figure 5.c. After initializing the belief states for the second and third points, we can update our belief by factoring in the evidence already gathered. The effect of using this technique is shown in Figures 5.b and 5.d.

EXPERIMENTAL RESULTS

The three detailed algorithm instances (baseline, optimized with standard deviation heuristic, and optimized with entropy heuristic) were run with the target object in three locations. The results are in Figures 7, 8, and 9. The two optimized heuristics do significantly improve over the baseline version.

[1]Although some of the points may be in these invalid regions, it is guaranteed that not all of the points will be in these regions.

CONCLUSIONS

In this work we have proposed a new framework and algorithms for planning robotic Eye-in-Arm search of known objects. Our framework is inspired from the artificial intelligence theory of learning based on decision trees. As such, each snapshot-taking operation in the search process corresponds to a question being asked in the decision tree, while the possible positions of the searched object correspond to the possible samples classified by the decision tree. As a further parallel to decision trees, we test search heuristics based on entropy, as inspired from the information theoretical ID3 learning algorithm.

The main difference between our search framework and decision tree learning is that we build the relevant tree branches dynamically for each classification, rather than pre-building it once for all (except for the version keeping one likely branch in the baseline technique). This is due to the size of the tree which would be difficult to store. Another difference is in the fact that we do not use training data but rather compute expected outcomes based on current robot beliefs, represented by a probability distribution of the object location. Considered questions (i.e., snapshot positions) are sampled out of the total set of possible questions using stochastic search or dichotomous beam search. Actions can be selected based on comparing candidate plans composed of sequences of multiple look-ahead steps and evaluating the expected beliefs. This is a process also reminiscent of POMDP policy evaluations.

Experimental evaluations are performed using an ST-12 robotic arm with a camera mounted in its end-effector, and three algorithm versions. A baseline version with a simple search whose intelligence mainly lies in the computer vision part is compared against two heuristics for the robotic arm next-question selection: namely based on entropy, and based on standard deviation. For the described experiments, the two intelligent search processes performed comparatively well, and much better than the technique focusing on computer vision. The intelligent search heuristics are expected to excel in different problem sub-domains, and the identification of these domains is planed for future work, as well as the exploration of several mentioned optimizations.

REFERENCES

1. Pratik Chaudhari, Sertac Karaman, David Hsu, and Emilio Frazzoli. 2013. Sampling-based algorithms for continuous-time POMDPs. In *2013 American Control Conference*. IEEE, 4604–4610.
2. H. Kurniawati, David Hsu, and Wee Sun Lee. 2008. SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces. *Robotics: Science and Systems* 2008 (2008).
3. Joelle Pineau, Geoff Gordon, and Sebastian Thrun. 2003. *Point-Based Value Iteration: An Anytime Algorithm for POMDPs*. Technical Report. Proc. of IJCAI, Mexico.
4. Pascal Poupart, Kee-Eung Kim, and Dongho Kim. 2011. Closing the gap: Improved bounds on optimal POMDP solutions. In *ICAPS*.

Test	StdDev			Entropy			Baseline
	Find 1st	Find 2nd	Find 3rd	Find 1st	Find 2nd	Find 3rd	
1	2	1	1	4	1	1	28
2	4	0	1	3	1	2	
3	2	1	1	3	1	1	
4	2	1	1	3	1	1	
5	2	1	1	3	1	1	
6	5	1	1	3	1	1	
7	2	1	1	3	1	1	
8	4	0	1	4	1	1	
9	2	1	1	2	1	1	
10	4	1	0	4	1	1	
Average	2.9	0.8	0.9	3.2	1	1.1	28
Cumulative	4.6			5.3			

Figure 7. Position 1

Test	StdDev			Entropy			Baseline
	Find 1st	Find 2nd	Find 3rd	Find 1st	Find 2nd	Find 3rd	
1	9	1	1	2	1	2	49
2	3	3	1	4	2	1	
3	10	1	1	6	1	1	
4	7	2	8	4	2	1	
5	5	2	1	7	0	1	
6	7	2	6	4	1	1	
7	7	3	2	5	1	1	
8	6	2	2	7	2	2	
9	4	2	2	3	2	1	
10	8	1	2	6	1	1	
Average	6.6	1.9	2.6	4.8	1.3	1.2	49
Cumulative	11.1			7.3			

Figure 8. Position 2

Test	StdDev			Entropy			Baseline
	Find 1st	Find 2nd	Find 3rd	Find 1st	Find 2nd	Find 3rd	
1	5	1	1	2	1	4	9
2	4	1	1	4	1	1	
3	4	1	1	3	1	1	
4	4	1	1	3	1	2	
5	4	1	1	3	1	1	
6	5	1	4	3	8	3	
7	4	1	1	4	1	1	
8	5	1	0	2	1	1	
9	4	1	1	4	1	1	
10	4	1	0	2	1	1	
Average	4.3	1	1.1	3	1.7	1.6	9
Cumulative	6.4			6.3			

Figure 9. Position 3

5. Jan Rosell, Raúl Suárez, Carlos Rosales, and Alexander Pérez. 2011. Autonomous motion planning of a hand-arm robotic system based on captured human-like hand postures. *Autonomous Robots* 31, 1 (2011), 87.
6. Stuart Russell and Peter Norvig. 2013. *Artificial Intelligence — A Modern Approach*.
7. Marius C. Silaghi and Jixing Zheng. 2017. POMDPs for Robotic Arm Search and Reach to Known Objects. In *ArXiv: 1704.07942*.
8. Srinivasa Venkatesh and Marius C. Silaghi. 2015. Planning One Eye-in-Arm Robot for Object Localization. In *Proceedings of FCRAR 2015*.
9. Srinivasa Venkatesh and Marius C. Silaghi. 2016. Optimizing Eye-in-Arm Robot for Localization of Known Objects. In *Proceedings of FCRAR 2016*.