Optimizing eye-in-arm robot localization of known objects

Srinivasa Venkatesh, Marius C. Silaghi

Florida Institute of Technology
Computer Science
150 W University Blvd
Melbourne, FL 32901

svenkatesh2010@my.fit.edu, msilaghi@fit.edu

ABSTRACT

Locating known objects is an important task for robots. When a robot has a single camera located in its arm, the robot can use it to get pictures from multiple points of view. These pictures can be used for locating in 3D a desired object, when the object is found on the floor within a bounded distance from the robot.

Here we propose a technique for planning a sequence of arm positions to be used for capturing the camera snapshots that can locate the object with given precision. Heuristics are used to reduce the number of steps (i.e., camera snapshot taking operations) performed by the robot following the obtained contingency plan.

Keywords
Object Localization, Eye-in-hand

1 INTRODUCTION

In common industrial applications the robotic hand is supposed to know precisely where to find the object that it has to manipulate. However, errors and unexpected events may place the target object in unexpected locations and orientations.

A stereo vision system surveying the whole operational environment can locate the target object and let the robot update its working plan accordingly. However, such a setup can encounter problems if the view of the target object is obstructed by some features of the environment.

Furthermore, when the robotic hand is placed on a mobile platform, the environment of the robot may be too large to be efficiently covered by an external stereo vision system.

It has been therefore considered relevant to address the problem of locating target objects using cameras found in the arm of the robot. Such a setup gives the robot significant flexibility in searching for objects in complex environments.

While one can place a stereo vision system in the arm of the robot, a single camera can also be sufficient, since the mobility of the arm enables the robot to dynamically construct its stereo vision with images taken from multiple points of view.

Compared to a stereo vision system located in the arm, the limited precision of the arm movement, combined with other errors in stabilizing the robot support, may lead to higher errors in the 3D location inferred from two given images captured using a single camera. However, the robotic arm can take an unlimited number of pictures from a multitude of points of view, compensating for these errors at the expense of a set of extra movements.

Besides the problem of computing the exact location of the object from stereo vision, a planning technique needs to be designed for exploring the environment of the robot in search of cues for the occurrence of the object. We propose to identify the object using a combination of features, namely using a Bayesian Network to fuse separate detectors based on color, shape and texture of the object.

In this work we assume that the object is placed within a bounded distance from the trunk of the robotic arm. We report experiments with a ST12 robotic arm equipped with a Sentech ST-MC33 camera in its hand and which

looks for a green box within a square of 1.14 meters, centered in its base.

2 RELATED WORK

Our work is at the intersection of research in planning, search, vision, and Bayesian sensor fusion. Planning is the process of thinking about and organizing the activities required to achieve the goal. It involves creation and maintenance of a plan. Search involves finding an item with specified properties among a collection of items. Vision perception is the ability to interpret the surrounding environment by processing information contained in visible sight.

Search The most widely known form of best-first search is called A^* search. The A^* algorithm evaluates nodes by combining $g(n)$, the cost to reach the node, and $h(n)$, the estimated cost to get from the node to the goal:

$$f(n) = g(n) + h(n) \quad (1)$$

Since $g(n)$ gives the path cost from the start node to node n, and $h(n)$ is the estimated cost of the cheapest path from n to the goal, we have:
$f(n) = \textit{estimated cost of the cheapest solution through } n.$
Thus, if we are trying to find the cheapest solution, a reasonable thing to try first is the node with the lowest value of $g(n) + h(n)$. Provided that the heuristic function $h(n)$ satisfies certain conditions, A^* search is both complete and guaranteed to find a solution with optimal direct path cost.

Greedy Best First search tries to expand the node that is closest to the goal, on the grounds that this is likely to lead to a solution quickly. Thus, it evaluates nodes by using just the heuristic function; that is, $f(n) = h(n)$.

Bayesian Recognition Consider a set of objects $o_i, i \in \{1, ..., n\}$ and a camera facing an object whose class and pose are to be identified. Let the camera measurement be parameterized by a feature vector d, which depends on the identity o_i of the object, its pose θ and the viewing position v. Under uncertainty, this relationship can be represented through a probability density function

$$p(d \mid o_i, \theta, v), i = 1, ..., n; \theta \in \mathbb{S}^2, v \in V$$

(where $\mathbb{S}^2$ is the surface of a unit sphere and V denotes the set of possible camera viewpoints) whose parameters are assumed to be learned or modeled off-line through some training procedure [12].

Bayesian Networks The Bayesian Network represents the dependencies among variables. Bayesian Networks can represent essentially any full joint probability distribution and in many cases can do so very concisely. A Bayesian Network is a directed graph in which each node is annotated with quantitative probability information. The Bayesian network formalism allows for efficient representation and reasoning with uncertain knowledge [16].

A directed graph G can be defined as an ordered pair that consists of a finite set V of nodes and an irreflexive adjacency relation E on V. The graph G is denoted as (V, E). Each $(x, y) \in E$ represents an arc (directed edge) from node x to node y. In the graph, this is denoted by an arrow from x to y, and x and y are called the start point and the end point of the arrow respectively. We also say that node x and node y are adjacent or x and y are neighbors of each other. x is also called a parent of y and y is called a child of x. By using the concepts of parent and child recursively, we can also define the concept of ancestor and descendant. We also call a node that does not have any parent, a root node. By irreflexive adjacency relation we mean that for any $x \in V$, $(x, x) \notin E$, i.e., an arc cannot have a node as both its start point and end point.

A Bayesian network is a directed graph in which:

1. Each node corresponds to a random variable, which may be discrete or continuous.
2. A set of directed links or arrows connects pairs of nodes. If there is an arrow from node X to node Y, X is said to be a parent of Y. The graph has no directed cycles, and hence is a directed acyclic graph (DAG).
3. Each node X_i is associated with a conditional probability distribution $P(X_i | Parents(X_j))$ that quantifies the effect of the parents on the node.

The topology of the network (i.e., the set of nodes and links), specifies the conditional independence relationships that hold in the domain. The intuitive meaning of an arrow from X to Y is that X has a direct influence on Y shielding Y from the influence of indirect ancestors of X. As a heuristic one uses causes as parents of effects. It is usually easy for a domain expert to decide what direct influences exist in the domain, based on cause-effect relations. Once the topology of the Bayesian network is laid out, one needs to be able to specify a conditional probability distribution for each variable, given its parents [16].

We use a Bayesian network to determine the probability of detecting the object. The Bayesian network

can be used to model a world and to answer probabilistic queries about the random variables that describe this world. For example, the network can be used to update the knowledge about the state of a subset of variables (query variables) when other variables (the evidence variables) are observed. This process computes the posterior distribution of query variables given evidence by probabilistic inference. The posterior helps choose values for a subset of variables to minimize some expected loss function, for instance the probability of decision error [6].

Object Recognition Gradient based features are included in our analysis because they can be used to detect local changes in color, texture, and brightness. Here, we use the computational architecture of gradient features in [13].

The emphasis on local texture descriptors [17], is the dominant approach today. The appearance based descriptors summarize local texture information in the form of histograms of gradients [8], shape context [1], and geometric blur [2]. While prominent edges are roughly encoded, exact shape location has been replaced by a representation of texture.

Various linear or non-linear filtering operations are available for 2D images such that for each pixel location (x, y) in the source image, its neighborhood is considered and used to compute the response. In case of a linear filter, this is a weighted sum of pixel values. In case of morphological operations, the filter exploits the minimum or maximum values.

The computed response is stored in the destination image at the same location (x, y). The output image is of the same size as the input image. Normally, the functions support multi-channel arrays, in which case every channel is processed independently. Therefore, the output image will also have the same number of channels as the input one [3].

Robotic Arm Vision A perception driven object recognition process was implemented in [4] that allows a humanoid robot to recognize objects by actively resolving ambiguities. It also demonstrates in simulation and in real life experiments that by employing additional viewpoints, objects can be identified faster. It generates plans in the joint angle space of the robot, which resulted in speeding up the recognition process [4].

A strategy for optimal action selection with the purpose of recognizing objects based on a database of predefined images can be used to reduce uncertainty and ambiguity [9]. An extension to stereo vision is described by [18].

Bayesian approaches have been used for actively selecting camera parameters in order to recognize a given object from a finite set of object classes. A Gaussian process regression is applied to learn the likelihood of image features in [10] given the object classes and camera parameters. The object recognition task was treated as a Bayesian state estimation problem.

Decomposition into regions can be used in a similar way to the local interest points extracted from gray-level images, but to capture shape rather than texture [11]. The new kind of shape feature based on annular regions was proposed for recognition in presence of occlusion and clutter. For detecting generic objects in static images, the active basis model (ABM) [5] utilizes, a grey-value local power spectrum to find a common template and deformable templates from a set of training images and to detect an object in unknown images by template matching using color based features.

The local contour descriptors can complement texture descriptors for object recognition [17]. Object contours are a strong representation of shape, whereas texture-based representations summarize contours to avoid matching them to an object exactly. In [17] a local contour representation complements texture features by encoding junction information and curvature. The representation discretizes contour orientation at interest points and records the intensity of the contour at each angle as feature elements. A hierarchical system building an increasingly complex and invariant feature representation is described in [19].

Active object recognition is the technique to reduce the uncertainty of single view recognition, by planning sequences of views, actively obtaining these views, and integrating multiple recognition results [7]. Understanding recognition as a sequential decision problem challenges the visual agent to select discriminating information sources. Bayesian sensor fusion can be used in disambiguate initial object hypotheses [15]. Instance based learning is used for training module parameters. Using a parameterized appearance based model [14], the fusion of successive probabilistic interpretations integrates information about the spatial structure of the object model. The view planner favors the object hypotheses which are consistent with learned observations trajectories in feature space. To enable real-time control, an instance based classifier is outlined to derive a decision policy directly from the stream of action sequences induced by the Bayesian planner.

In Bayesian interpretation with uncertain and noisy environments, classification on the basis of a crisp mapping from observations y to symbols o_i is replaced

by a sensor model [15]. Given the object o_i under visual parameter φ, the likelihood of obtaining feature vector y is denoted by $p(y \mid o_i, \varphi)$. The likelihood is estimated from a set of sample images with fixed o_i and φ_j, capturing the inaccuracies in the parameter φ_j such as moderate light variations or segmentation errors. From the learned likelihoods one obtains via Bayesian inversions [15]:

$$P(o_i, \varphi \mid y) = p(y \mid o_i, \varphi_j) P(\varphi_j \mid o_i) \frac{P(o_i)}{p(y)} \quad (2)$$

and a posterior estimate with respect to the object hypotheses o_i is given by $P(o_i \mid y) = \sum_j P(o_i, \varphi_j \mid y)$. Note that a corresponding estimate for pose φ_j is determined by

$$P(\varphi_j \mid o_i, y) = \frac{P(o_i, \varphi_j \mid y)}{P(o_i \mid y)} \quad (3)$$

3 PROBLEM FORMALIZATION

The problem we address is the use of a robotic arm equipped with a camera in its hand to determine the position of an object (green box) of known dimensions. This object can be placed at any location and with any orientation within the robot work-space.

Based on its environment, i.e., planning in a partially observable space, our problem falls in the category of problems best modeled as `partially observable Markov decision processes` (POMDP). However, unlike most POMDPs, in our problem there is no uncertainty in the transition, as the robotic arm is considered to be controlled in a reliable manner and we assume that the rest of the environment is static.

In a partially observable environment, every percept (sensor reading) helps narrow down the set of possible states the agent and environment might be in, thus making it more probable for the agent to achieve its goals. So the solution to a problem is not a sequence of actions but a `contingency plan` (also known as a strategy) that specifies what to do depending on what sensor values are obtained [16].

A sequential decision problem for a fully observable, stochastic environment with a Markovian transition model and additive rewards is called a `Markov decision process`, or `MDP`. An MDP is formalized as a set S of states (with an initial state s_o), a set A of actions for each state, a transition model M specifying the probability of transition between any two states s_1 and s_2 given a taken action a $P(s_2|s_1, a)$, and a reward function K_s associating each state with an utility of visiting it for a unit of time [16].

MDP Model It is common to address a POMDP by reducing it to an MDP, namely where the states are redefined to describe points in the space of possible beliefs [16]. If we define the state of our process as a tuple, including both the position of the robot and a current belief map about the position of the searched object, then the problem obtained is an MDP defined as follows:

- S: a set of states $S = \{s_i\}_i$ (a state $s_i =< r_i, \{\sigma_k, p_k\}_k >$ is an arm position r_i and a belief map associating a set of areas σ_k covering the search space, with object occurrence probabilities p_k)
- A: a set of actions of type Try(X), consisting of moving the arm to a pose X and capturing an image to analyze the seen area. Each such action has a certain cost (e.g., time, energy). The sum of all actions cost over the whole plan should be minimized.
- M: a set of transitions probabilities $\{M_{i,j}^{a_t}\}_{i,j,k}$ (from each belief map s_i to each new belief map s_j, given an action a_t).
- G: a goal state (belief map with a FOUND state or all REJECT/UNKNOWN states)

Since the number of possible actions is large, the obtained MDP model is complex and its exact solving is left for future work. However, we use it as a source of inspiration for the following heuristic solver.

Initially, we did consider the A^* search model, but the A^* algorithm considers the cost to reach the node. In our problem, we already reached the node. To avoid the re-optimization, for this thesis we decided to explore a solution based on modeling the problem using the General Search Framework [16].

General Search Problem While this is a planning problem, one heuristic that we use here is to represent it as a search problem and to extract the plan as the physical execution of an efficient general search algorithm for this problem. The problem can thus be represented as a general search problem with parameters (S, A, M, G):

- S: a set of states $S = \{s_i\}_i$ (a state $s_i =< r_i, \{\sigma_k, p_k\}_k >$ is an arm position r_i and a belief map associating a set of areas σ_k covering the search space, with object occurrence probabilities p_k)

- A: a set of actions of type Try(X), consisting of moving the arm to a pose X and capturing an image to analyze the seen area. Each such action has a certain cost (e.g., time, energy). The sum of all actions cost over the whole plan should be minimized.

The solution (plan) in this case consists in the steps of the search algorithm used to solve this search problem. We propose to use the Greedy Best First Search [16]. Greedy Best First Search is an instance of the `A search algorithm`. This technique is appropriate since it does not try to optimize the cost of already achieved operations, which cannot be recovered once the robot has spent time and energy performing them. The Greedy Best First Search maintains a frontier of the explored search space, extending at each step the most promising one (the state believed to be closer to the solution).

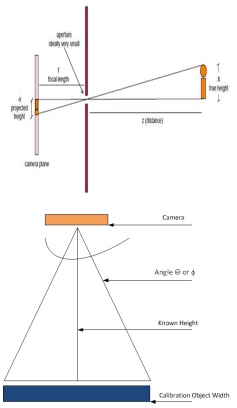


Figure 1: (a) Focal point relationship. (b) Angle calibration

Viewpoint Selection The object recognition problem can be defined as that of finding the viewpoint selection strategy that minimizes the number of observations required to perform recognition of an unknown object with a particular level of confidence. This strategy is dependent on the relationship between camera observations, object class, object pose and camera parameters (i.e. viewing position).

In order to determine an object's position in space from the image captured by camera, the focal point of the camera must be known. If the focal point of the camera is not known in advance, it can be experimentally determined. In our case this was done by taking advantage of the relation of the angle between the lens and field of view and the angle between the lens and the focal point as seen in Figure 1.a. The angle was determined by photographing an object of known dimensions at a known vertical distance away as seen in Figure 1.b.

4 PLANNING TECHNIQUE

In the current work we experiment with a planning technique based on performing the Greedy Best First Search algorithm in a belief space. This technique evaluates the quality of each state (its belief state), based on analyzing the image captured by the camera at that particular location.

Analyzing the image and the association of a posterior probability for the occurrence of the object at that location is achieved via Bayesian sensor fusion. The sensors involved in this process are in fact algorithms using different types of visual features.

4.1 Visual Features

We use the following sensors (outputs of detection algorithms) as random variables in the Bayesian fusion employed to detect the object in the work-space.

1. Object's Color match score
2. Object's Shape match score
3. Object's Texture match score

Now we introduce the features used by each of these sensors.

4.1.1 Color Feature

To detect and segment an object from an image one can use its color. The colors in the object and the background should have a significant difference in order to provide information for the segmentation. For this sensor, initially one needs to determine which colors to find and how to separate the object from the background colors.

In this case we transform the image from the Red-Green-Blue (RGB) color space to the Hue Saturation Value (HSV) color space. We use a filter function to specify scalar lower and upper bound parameters for the color threshold. Color thresholds can be learned automatically by comparing images of the known target object with images of the background, and can be trained using support vector machines (SVMs). In our experiments the threshold value filter applies for the green

color, as detected by the OpenCV SVM functionality in function `cvLatentSvmDetectObjects` [3].

4.1.2 Shape Feature

Another important family of features describe the shape of the object. To segment the shape, we convert the color image to gray. Then we find the contour using the technique proposed in [20]. For this purpose, we use the OpenCV `SurfFeatureDetector.detect` keypoint detector. Keypoints are computed both for the template and for the candidate images. The Hamming distance between the keypoints of a candidate and a template is evaluated using the technique `BFMatcher.match` from OpenCV, and we use the percentage of the matching as the output of this sensor.

4.1.3 Texture Feature

The texture match sensor employs a set of template images for the object. Five template points-of-view/images are provided, and are shown in a later section. We compare a template image against overlapped image regions using the square differences method (CV_TM_SQDIFF in OpenCV's `matchTemplate` function). The *minimum* and *maximum* element values and their positions are calculated using the `minMaxLoc` function. The extremes are searched across the whole array in the specified region.

4.1.4 Image segmentation and feature extraction

The object in the robot work-space occupies only some area of the image obtained from the robot camera. To separate the background area from the object we train a Gaussian Mixture Model (GMM). GMMs are commonly used for background removal tasks. They are specified by K normal distributions $N_k(\mu_k, \sum_k)$ with mean μ_k and convariance $\sum_k$ as well as a weight w_k [4].

Figure 2 shows the Segmentation Process: A background model is trained on the area and applied to the image. Low values in the resulting probability map indicate the presence of an object.

Figure 2: Image segmentation and feature extraction.

The probability P_{BGR} of an image pixel I being considered as background is computed by evaluating the weighted probability density function (PDF) of the multivariate distribution defined by [4]:

$$P_{BGR} = \sum_{k=1}^{K} \frac{w_k}{\sqrt{|\ 2\Pi \sum_k\ |}} \cdot e^{\frac{1}{2}(I-\mu)^T \sum_k^{-1}(I-\mu)} \quad (4)$$

We consider all image locations with background probability smaller than a fixed threshold as being occupied by the object. Finally morphological operations are applied to remove clutter and artifacts.

5 PLANNING THE SEARCH

We detail the planning process for the search of the known object in the environment of the robot. The original search space where the object is believed to be found is first split into sub-regions as described in Section 5.1. Further, a plan is followed to explore some of these regions and, if needed, regions are further split into smaller regions according to heuristics. The presence of the object in a given area is evaluated using the sensor fusion technique detailed is Section 5.3.

5.1 Search Space Splitting

The robot recursively divides its current work-space into 2D regions. The robot scans each region and processes the pictures for detection. A fragment of a sample top-level division into regions is shown in a schematic top-view of the environment in Figure 3.

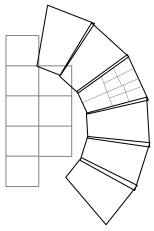


Figure 3: Fragment of a top view for the robot work-space.

When the output of the sensors is processed for a region using the Bayesian sensor fusion, we obtain a posterior probability of the occurrence of the object in the region. In each step, the region with the highest current posterior probability of containing the object is further split into sub-regions with overlapping margins, as shown in Figure 4.a.

The current region is further divided into a number of sections and the obtained sections are added to the

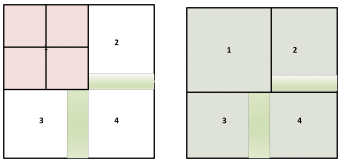


Figure 4: (a) Robot's workspace frontier extending. (b) Robot's worskspace frontier after extension.

frontier with the score of the match obtained for them from the image of the parent area. An illustration of the implicit hierarchy in the scan process is illustrated in Figure 5. The space is fragmented up to a certain depth ($MAXDEPTH$), after which the camera would come too close to the object for a reliable analysis.

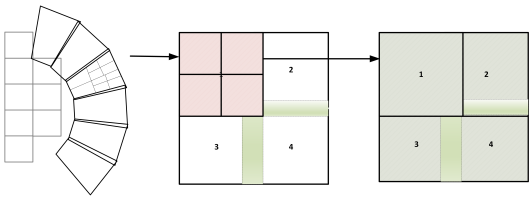


Figure 5: Robot's work-space scan diagram.

Figure 6.a shows the starting point of scanning the work-space, corresponding to the first top region in Figure 5. Once the robot decides that it has found the matching object, it determines the x, y and z coordinates of its corners. Figure 6.b shows a view of the object placed in the work-space.

Figure 6: (a) Scanning start position. (b) Scanning in the work-space.

5.2 Object Finding Algorithms

To find the object in a given work-space of the robot, we have followed the logic and procedure which are described in Algorithms 1 and 2. Algorithm 1 expands the search frontier by splitting the work-space received as parameter into sub-areas, hierarchically, capturing and passing the image of the most promising one to Algorithm 2. This

Algorithm 1: Find the object by scanning the Robot's work-space

```
1  function FindObjectBySearch(SearchSpace)
2      add to RegionsQueue result of
       InitialSplit(SearchSpace) with object presence
       probability given by prior_probability / #splits;
3      forever do
4          if empty(RegionsQueue) then break;
           crt_region ← extractHead(RegionsQueue);
5          Position robot camera to capture crt_region;
6          Capture image;
7          (status, location, probability) ←
           FindObject(image);
8          if (status = FOUND) OR (status = REJECT)
           then
9              return (status, location);
10         end
11         if scanDepth(crt_region) == MAXDEPTH
           then
12             updateBestFound(probability,location)
13         else
14             add(RegionsQueue, split(crt_region, image));
15         end
16     end
17     return (bestFound, location);
```

2^{nd} algorithm calculates the probability of the occurrence of the target object based on Bayesian network inference. It decides whether the object is present in the region, absent in the region, or if more search is needed, based on the obtained probability value. The conclusion is returned to Algorithm 1 in a tuple containing a status, location and probability value for the image. Based on the data which is received, Algorithm 1 decides whether the object is found and if the condition is met, it would return the status and location of the object to the user. Otherwise, if more search is needed, it splits the region and adds its fragments tagged with the corresponding probability of the occurrence value into the `Regions priority queue`. A priority queue is a data structure which helps extracting efficiently the area with the highest probability value on each query.

The techniques loops while the `Regions` queue is not empty. It retrieves the most promising region (with the highest probability of occurrence for the object) from the queue and analyzes it. First the `current-region` is analyzed by Algorithm 2 to extract the status of the detection, the location of the object and its probability

Algorithm 2: Find the object in a given image

```
function FindObject(image)
    Calculate probability value and assign them;
    ColorMatchVal ← Get color match score;
    ShapeMatchVal ← Get shape match score;
    TextureMatchVal ← Get texture match score;
    ImgProbValue =
    BN(ColorMatchVal, ShapeMatchVal, TextureMatchVal);
    location ← FindObjectDistance(image);
    if current_Scan_Depth == MAXDEPTH then
        return (UNKNOWN, ImgProbValue);
    end
    if (ImgProbValue≥MaxProbValue) then
        return (FOUND, location, ImgProbValue);
    end
    if (ImgProbValue<MinProbValue) then
        return (REJECT, location, ImgProbValue);
    end
    return (LIKELY, location, ImgProbValue);
```

Algorithm 3: Find the object distance or location for a given image

```
function FindObjectDistance(crt_image)
    Load crt_image;
    if empty(crt_image) then
        image_not_valid;
        crt_image ← setDistance(0);
        break;
    end
    Convert the image to gray-scale, blur it, and detect
    edges.;
    gray_image ←
    applyColorConversion_BGR_TO_GRAY;
    blur_image ← applyGaussianBlur(gray_image);
    canny_image ← applyCannyEdges(blur_image);
    contour_image ← findContours(canny_image);
    applyContourApproximation(contour_image);
    keypoint_match ←
    applyKeyPointDetectAndMatch;
    AreaRectangle ← CalcArea(contour_image);
    calibrate_focal_len ← loadCalibrateImage;
    distance = (width * focal length) / image pixels;
    return (distance);
```

value. If more search is needed, one checks whether the scan depth reached the maximum depth value. If it is reached, then the obtained probability and location is used to update the current best found hypothesis for the object location (if the occurrence probability is higher then previously found ones). If the scan depth has not reached the maximum depth, then Algorithm 1 would split it and add its fragments and their corresponding probabilities to the `Regions` priority queue. The `split` function used by the above algorithm breaks the region in 4 areas and tags each of them with their corresponding posterior probability obtained from the corresponding part of the image.

Algorithm 3 shows the step by step procedure to calculate the object's distance or location. The current image will be loaded and converted from BGR color space to GRAY color space. This allows for applying a Gaussian Blur filter to take any extra noise in the image. After filtering the image, apply the Canny Edge technique to detect the edges of the object. We extract its contour, and using contour approximation technique we remove unrelated contours. Calculate the keypoints to detect and match the object. Then calculate the area of the object.

In this algorithm, to determine the distance from the camera, we are going to utilize the technique called triangle similarity. For example, if we have a object with a known width W. We then place this object some distance D from the camera. We take a picture of our object using the camera and then measure the apparent width in pixels P. This allows us to derive the perceived focal length F of the camera

$$Focal\_Length = \frac{(Pixels * Distance)}{Width}$$

When we move the camera both closer and farther away from the object, we can apply the triangle similarity to determine the distance of the object to the camera by using the following technique.

$$Distance = \frac{(Width * Focal\_Length)}{Pixels}$$

Triangle Similarity Two geometrical objects are called similar if they both have the same shape, or one has the same shape as the mirror image of the other. More precisely, one can be obtained from the other by uniformly scaling (enlarging or shrinking), possibly with additional translation, rotation and reflection. This means that either object can be re-scaled, re-positioned, and reflected, so as to coincide precisely with the other object. If two objects are similar, each is congruent to the result of a particular uniform scaling of the other. A modern and novel perspective of similarity is to consider geometrical objects similar if one appears congruent to the other when zoomed in or out at some level.

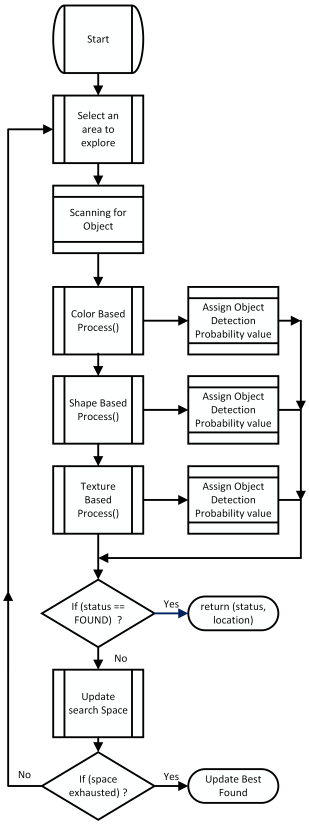


Figure 7: Flow diagram for the object detection process.

5.3 Bayesian Network

The Bayesian network formalism was invented to allow efficient representation of, and rigorous reasoning with uncertain knowledge. The approach allows for learning from experience, and it combines the best of classical AI and neural nets [16]. The formal definition for Bayesian Networks was detailed in Section 2. The combination of the topology and the conditional distributions suffices to specify (implicitly) the full joint distribution for all the variables specified in our example.

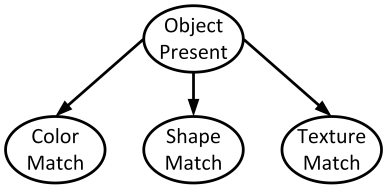


Figure 8: Bayesian Network Model.

Figure 8 shows the Bayesian Network Model used in our experiments. We use the Equation 5 to calculate the probability for the given image, where O stands for `Object Present`, C stands for `Color Match`, S stands for `Shape Match` and T stands for `Texture Match`.

$$p(O \mid CST) = \frac{p(C \mid O)\, p(S \mid O)\, p(T \mid O)\, p(O)}{p(CST)} \quad (5)$$

$$p(CST) = \sum_{o} p(C \mid o)\, p(S \mid o)\, p(T \mid o)\, p(o) \quad (6)$$

5.4 Bayesian Network Design

The Bayesian Network was built to model the relation between the real presence of an object in an image and the quality of the hypothesised matches proposed by various techniques/sensors (color match, shape match, texture match). In our experiments we use three different sensors to detect the object.

5.5 Bayesian Network Training

We train the Bayesian Network, inferring the posteriors in its conditional probability tables from experiments. When we process an image with our three different techniques (e.g., Color, Shape, and Texture), each of them computes a matching score for the object. The Bayesian Network is trained, helping to merge these scores into a probability of the match.

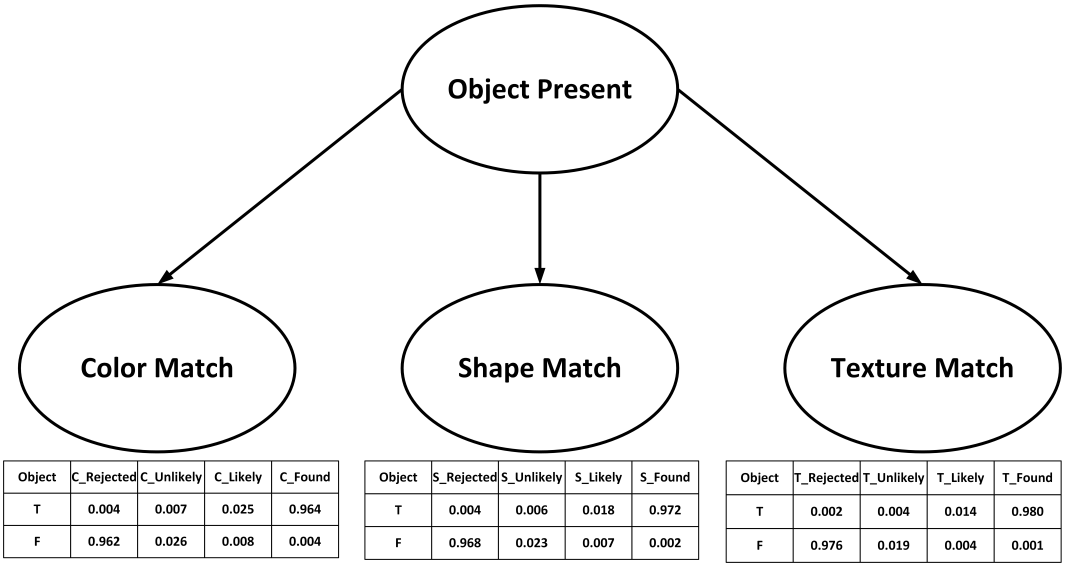
We perform supervised training, namely where each image is tagged as either containing or not containing the target object. Originally all entries in the CPT are 0. For each image analyzed, we obtain the classification for the current detectors, and we increment the entry in the corresponding table entry. After all training images are analyzed, each row is normalized by dividing each entry by the sum of the two entries in the row.

6 EXPERIMENTS

In our experiments, a Bayesian Network was trained using images taken of the search-space from 6 different positions and with 3 different environment luminosity. We ran 100 experiments using the aforementioned algorithm.

6.1 Scanning and Detection Experiments

This section contains the experiments with more sensor images. During the training phase of algorithm, a Bayesian Network has been used to train the table and generate Conditional Probability Table (CPT) for each



Object	C_Rejected	C_Unlikely	C_Likely	C_Found
T	0.004	0.007	0.025	0.964
F	0.962	0.026	0.008	0.004

Object	S_Rejected	S_Unlikely	S_Likely	S_Found
T	0.004	0.006	0.018	0.972
F	0.968	0.023	0.007	0.002

Object	T_Rejected	T_Unlikely	T_Likely	T_Found
T	0.002	0.004	0.014	0.980
F	0.976	0.019	0.004	0.001

Figure 9: Bayesian Network Diagram with CPTs. The CPT/prior of the root is given in the problem formulation.

sensor. The following tables are generated based on the experimental results.

The conditional distributions in Figure 9 are shown as a conditional probability table (CPT). This form of table can be used for discrete variables: other representations, including those suitable for continuous variables. Each row in a CPT contains the conditional probability of each node value for a conditioning case. A conditioning case is just a possible combination of values for the parent nodes. Each row will be normalized and they must sum to 1, because the entire represent an exhaustive set of cases for the variable. In general, a CPT table for a Boolean variable with k Boolean parents contains two independently specifiable rows of probabilities.

Table 1: Training phase of CPT for Color matching.

Object	$C_{Rejected}$	$C_{Unlikely}$	C_{Likely}	C_{Found}
T	12	21	74	2893
F	2886	78	23	13

6.2 Receiver Operating Characteristic (ROC) Curve for Object Scanning

A Receiver Operating Characteristic (ROC) curve is a graphical plot that illustrates the performance of a binary classifier system as its discrimination threshold is varied.

Table 2: Training phase of CPT for Shape matching.

Object	$S_{Rejected}$	$S_{Unlikely}$	S_{Likely}	S_{Found}
T	11	18	54	2917
F	2904	71	19	6

Table 3: Training phase of CPT for Texture matching.

Object	$T_{Rejected}$	$T_{Unlikely}$	T_{Likely}	T_{Found}
T	7	13	42	2938
F	2929	55	12	4

The curve is created by plotting false positives rate against false negatives rate at various threshold settings. We conducted 100 experiments to evaluate the Receiver Operating Characteristic (ROC). The Figure 10 shows the obtained False Positives vs. False Negatives ROC curve.

Case Study: A case study has been conducted on the experiments to find out how many images are required to detect the object in a given work-space. Object detection experiments have been done with two scenarios. First scenario is where the object was present in the given work-space. In this case, we did process and analyze 15 images

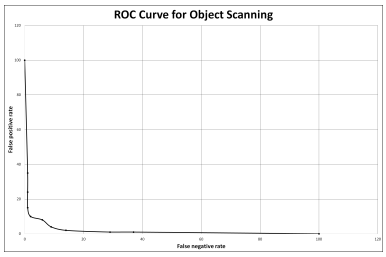


Figure 10: ROC Curve for Object Scanning.

to detect the object. But this could vary based on where the object is located in that work-space. The worst case scenario would be having to process a total of 21 images. The second scenario is where the object was not present in the given work-space. In this case the robot captures all the regions images, and processes and analyzes them to find the object. As we know that the object cannot be detected in those images, the algorithm processes and analyzes a maximum of 6 images. Therefore the total number of analyzed images is 21. Table 4 shows both scenarios along with the number of images each scenario analyzed. The error of the object location is approximately 1 cm.

Table 4: Case study on how many images required to detect object.

Scan	No. of Images
Object Present	15
Object Not Present	6

7 CONCLUSION AND FUTURE WORK

We addressed the problem of locating a known object within a predefined search space reachable from a robotic arm equipped with one camera on its arm. A planning technique is proposed based on heuristics for reducing the number of snapshot-capturing steps. This planning is based on running the greedy best first search algorithm over a formulation of the problem based on the general search problem framework. The heuristic employed by the search algorithm is based on the posterior probability of the object occurrence as generated by a Bayesian network fusing the output of three different sensors.

The original search space is segmented into a set of areas based on the maximum view range of the camera. After a global image is taken of an area, a probability of occurrence is assigned to it. Based on the planner, the most promising areas are further split until the object is found or is declared absent.

Once a first location of the object is hypothesized, the 3D position in refined using a sequence of snapshots captured on a path that the arm is following in approaching the target object. In our case study for the ST12 robot using a ST-MC33 camera, the target object (a green box) was detected in 21 snapshots.

REFERENCES

[1] S. Belongie, J. Malik, and J. Puzicha. Matching shapes. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 1, pages 454–461 vol.1, 2001.

[2] A. Berg and J. Malik. Geometric blur for template matching. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–607–I–614 vol.1, 2001.

[3] G. Bradski. The opencv library. *Dr. Dobb's Journal of Software Tools*, 25(11):120–126, 2000.

[4] B. Browatzki, V. Tikhanoff, G. Metta, H. Bulthoff, and C. Wallraven. Active object recognition on a humanoid robot. In *IEEE International Conference on Robotics and Automation (ICRA), 2012*, pages 2021–2028, 2012.

[5] T. Bui and K.-S. Hong. Supervised learning of a color-based active basis model for object recognition. In *Knowledge and Systems Engineering (KSE), 2010 Second International Conference on*, pages 69–74, 2010.

[6] J. Cheng, D. A. Bell, and W. Liu. Learning bayesian networks from data: An efficient approach based on information theory. Technical report, University of Alberta, 1998.

[7] S. Y. Chung and H. P. Huang. Simultaneous topological map prediction and moving object trajectory prediction in unknown environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2008. IROS 2008*, pages 1594–1599, Sept 2008.

[8] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893 vol. 1, June 2005.

[9] K. Deguchi and H. Ohtsu. An information theoretic approach for active and effective object recognitions. In *18th International Conference on Pattern Recognition, 2006. ICPR 2006*, volume 2, pages 622–622, 2006.

[10] M. Huber, T. Dencker, M. Roschani, and J. Beyerer. Bayesian active object recognition via gaussian process regression. In *15th International Conference on Information Fusion (FUSION), 2012*, pages 1718–1725, 2012.

[11] F. Jurie and C. Schmid. Scale-invariant shape features for recognition of object categories. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004*, volume 2, pages II–90–II–96 Vol.2, 2004.

[12] C. Laporte, R. Brooks, and T. Arbel. A fast discriminant approach to active object recognition and pose estimation. In *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, volume 3, pages 91–94 Vol.3, 2004.

[13] D. Martin, C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(5):530–549, May 2004.

[14] H. Murase and S. K. Nayar. Visual learning and recognition of 3-d objects from appearance. *Int. J. Comput. Vision*, 14(1):5–24, Jan. 1995.

[15] L. Paletta, M. Prantl, and A. Pinz. Learning temporal context in active object recognition using bayesian analysis. In *Proceedings of the 15th International Conference on Pattern Recognition*, volume 1, pages 695–699 vol.1, 2000.

[16] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall series in artificial intelligence. Prentice Hall, 2010.

[17] J. Schlecht and B. Ommer. Contour-based object detection. In *Proc. BMVC*, pages 50.1–50.9, 2011. http://dx.doi.org/10.5244/C.25.50.

[18] A. Selinger and R. Nelson. Appearance-based object recognition using multiple views. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2001. CVPR 2001.*, volume 1, pages I–905–I–911 vol.1, 2001.

[19] T. Serre, L. Wolf, S. Bileschi, M. Riesenhuber, and T. Poggio. Robust object recognition with cortex-like mechanisms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(3):411–426, March 2007.

[20] S. Suzuki and K. Abe. Topological structural analysis of digitized binary images by border following. *CVGIP*, 30(1):32–46, 1985.