Planning One-Eye-in-Arm Robot for Object Localization

Srinivasa Venkatesh
Department of Computer Sciences
Florida Institute of Technology
Melbourne, Florida 32901
svenkatesh2010@my.fit.edu

Marius Silaghi
Department of Computer Sciences
Florida Institute of Technology
Melbourne, Florida 32901
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 known precisely where to find the object that it has to manipulate. However, errors and unexpected events may place the target object in unexpected positions and locations.

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 an initial search of the object in the environment of the robot. 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 circle of one meter, centered in its base.

2. Related Work

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 [7].

Contour based representations have a long history in object recognition and computer vision. Even though previous approaches had some success, it is clear that finding contours exactly belonging to the shape of an object is a hard problem. This insight has given rise to an emphasis on local texture descriptors [8], the dominant approach today. These appearance based descriptors summarize local texture information in the form of histograms of gradients [5], shape context [1], and geometric blur [2]. While prominent edges are roughly encoded, exact shape location has been replaced by a representation of texture. Curvature of contours and junctions provide crucial shape information.

Object Recognition. Object recognition techniques have been the focus of a large amount of literature because of their direct application to real-world problems. The problem of object recognition in the presence of high uncertainty led to the development of systems that perform recognition using a sequence of observations from different points of view. The performance of an object recognition system depends on two major components: an inference component which fuses the evidence accumulated from successive observations, and an observation selection component which chooses the parameter settings for the next observation. In this work, probabilistic inference is employed as it is a particularly effective method for evidence fusion.

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

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

(where 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 [6].

The processing and understanding objects by robots is a process designed to produce information based on visual systems and software. In our experiment we use the Open Source Computer Vision Library (OpenCV) for detection and understanding the objects captured by the camera of the R12 robot.

Bayesian Networks. The Bayesian network is a knowledge representation technique for modeling uncertainty. A Bayesian network is a directed acyclic graph, where each node is associated with a condition probability table (CPT). The nodes in a Bayesian network represent random variables in a domain, and the arcs between nodes represent the conditional dependency relationships among the variables. The CPT of a node gives the conditional probability distribution of its variable given evidence about the value of variables in the parent nodes.

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). For each $(x, y) \in E$ say that there is 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.

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 when other variables (the evidence variables) are observed. This process of computing the posterior distribution of variables given evidence is called 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 [4].

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.

The problem can be formalized as a contingency search problem (S, A, M, G) over a set of beliefs:

- 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 occurence probabilities p_k)

- A: a set of actions (e.g., move arm, capture image to analyze an area), each of them having a certain cost (e.g., time, energy). The sum of all actions 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)

The set of transition probabilities M is approximated in our technique to assume that $M_{i,j}^{a_t}$ is maximal for raising p_k to 1 whenever the action a_t consists of analyzing σ_k.

Given the whole search space has been explored, the algorithm finds the location with the highest probability for matching the target object. It declares the object to be $FOUND$ when the probability surpases a high threshold.

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 symmetry of the angles between the lens and field of view and the angle between the lens and the focal point as seen in Figure 1.

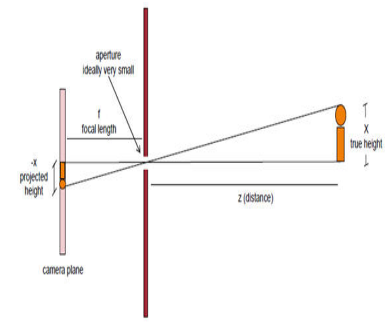

Figure 1. Focal point relationship.

The angle was determined by photographing an object of known dimensions at a known vertical distance away as seen in Figure 2.

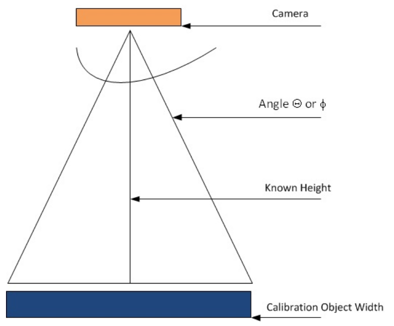

Figure 2. Angle Calibration.

4. Planning Technique

We use the following variables 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

Object's Color match score. 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. With this method, initially one needs to determine which colors to find and how to separate object from background colors.

Various linear or non-linear filtering operations are available for 2D images. It means 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.

In this case we transform the RGB image to HSV (Hue Saturation Value) image format. We use a filter function to specify lower and upper bound parameters for the color threshold and they are scalars. 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 verctor machines (SVMs). In our experiments the threshold value filter applies for the Green color.

Object's Shape match score. To segment the shape, we convert the color image to GRAY. Then we find the contour using the algorithm in [9].

In the contour template match algorithm, we load the created contour image file and the template contour image file. We use the OpenCV 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, and the score of the match is assumed to be the percentage of the matching.

Object's Texture match score. In this method, we have a set of template images for the object. There are five template points/images which are considered:

1. Bottom right corner of the object
2. Bottom left corner of the object
3. Top right corner of the object
4. Top left corner of the object
5. Top view of the object

We compare a template image against overlapped image regions using the square differences method (CV_TM_SQDIFF in OpenCV). The *min* and *max* are calculated over the sub-array, and then this sub-array is normalized. The global minimum and maximum are used to determine how close the object matching is. The template-based score of the match based on the comparison of the templates is evaluated as the ratio between range and an upper bound.

If the object detection score in an area is lower than the minimum threshold value then the area status will be updated as *Rejected*. If the object detection score is greater than or equal to the minimum threshold value and is less than or equal to the minimum good value, then the object status will be updated as *Unlikely*. If the object detection score value is greater than or equal to the Maximum Good Value and is less than the Maximum Threshold Value, then the object status will be updated as *Likely*.

If the object detection score value is greater than or equal to the Minimum Good Value and is less than or equal to Maximum Good Value, then the object status will be updated as *Found* as it scores high score compared to other scenarios.

These scores are used to infer a posterior probability of the occurence of the object in an area, using the Bayesian Network described later. The Figure 3 shows the Search splitting algorithm model used to explore the search space.

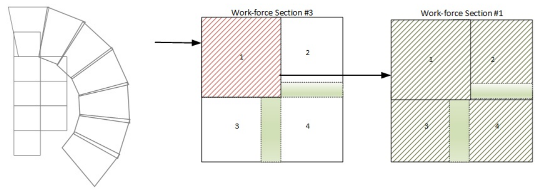

Figure 3. Search splitting algorithm model.

Algorithms used. 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 splits the work-space received as parameter into sub-areas, hierarchically, and passes them on to Algorithm 2. This 2^{nd}

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

```
function FindObjectBySearch(SearchSpace)
    (status, location, probability) ← FindObject($SearchSpace$);
    if (status = FOUND) OR (status = REJECT)) then
        return (status, location);
    end
    add($RegionsQueue$, ($SearchSpace, probability$));
    forever do
        if empty(RegionsQueue) then break;
        $first\_region \leftarrow extractHead(RegionsQueue)$;
        $(crt\_region, rest\_region) \leftarrow split(first\_region)$;
        (status, location, probability) ← FindObject($crt\_region$);
        if (status = FOUND) OR (status = REJECT) then
            return (status, location);
        end
        if scanDepth(crt_region) == MAXDEPTH then
            updateBestFound(probability,location)
        else
            add($RegionsQueue$, ($crt\_region, probability$));
        end
        if (not empty(rest_region)) then
            (status, location, probability) ← FindObject($rest\_region$);
            if (status = FOUND) OR (status = REJECT) then
                return (status, location);
            end
            if scanDepth(rest_region) == MAXDEPTH then
                updateBestFound(probability,location);
            else
                add($RegionsQueue$, ($rest\_region, probability$));
            end
        end
    end
    return (UNKNOWN, bestFound);
```

Algorithm 2: Find the object in a given current space

```
function FindObject(crt_space)
    Locate Robot camera to capture $crt\_space$;
    Capture image;
    Calculate probability value and assign them ;
    $ColorProbVal$ ← Get probability value;
    $ShapeProbVal$ ← Get probability value;
    $TextureProbVal$ ← Get probability value;
    $ImgProbValue = BN(ColorProbVal, ShapeProbVal, TextureProbVal)$;

    if $current\_Scan\_Depth == MAXDEPTH$ then
        return (UNKNOWN, ImgProbValue);
    end
    if ($ImgProbValue \geq MaxProbValue$) then
        return ($FOUND, location, ImgProbValue$);
    end
    if ($ImgProbValue < MinProbValue$) then
        return ($REJECT, location, ImgProbValue$);
    end
    return ($LIKELY, location, ImgProbValue$);
```

algorithm captures the image and calculates the probability of the occurrence of the target object based on Bayesian Network inference. It decides whether the object is present, absent, 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 would check 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 found needed, it adds the studied area and the 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 continuously 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 breaks it into `current-region` and `rest-region` by using the `split` function. First the `current-region` is analyzed by Algorithm 2 to extract the status of the detection, the location of the object and its probability value. If more search is found 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 add the current region and the probability to `Regions` priority queue.

4.1 Bayesian Network

Figure 4 shows the Bayesian Network Model used in our experiments. We use the Equation 1 to calculate the probability for the given image.

$$p(O \mid CST) = \frac{p(C \mid O)\, p(S \mid O)\, p(T \mid O)\, p(O)}{p(CST)} \quad (1)$$

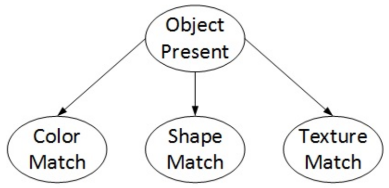

Figure 4. Bayesian Network Model.

$$p(CST) = \sum_{o} p(C \mid o)\, p(S \mid o)\, p(T \mid o)\, p(o) \qquad (2)$$

Figure 5 shows the flow diagram for the object detection process.

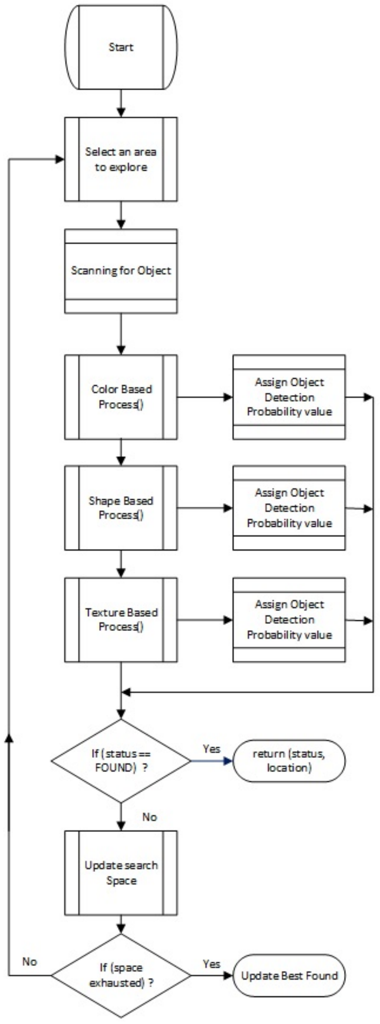

Figure 5. Flow diagram for the object detection process.

4.2 Bayesian Network Design

We build a Bayesian Network modeling 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 techniques to find the object.

4.3 Bayesian Network Training

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

Table 1 shows an example of the CPT for Color as it is populated during the training algorithm (prior to its normalization). We perform supervised training, namely where each image is tagged as either containing or not containing the target object. Originally all entries are 0. For each image analyzed, we obtain the classification for the current detector (e.g., Color), and we increment the entry in the corresponding table. After all training images are analyzed, the table is normalized such that the sum of the numbers on each row is 1.

Table 1. Training phase of CPT for Color.

Object	C_Rejected	C_Unlikely	C_Likely	C_Found
F	394	5	1	0
T	2	3	7	538

5. Experiments

A Bayesian Network was trained using 1000 images taken of the search-space from different positions and with different environment luminosities.

We run 100 experiments using the aforementioned algorithm. The object was detected in 99 experiments with an average of 6 snapshots for a maximum detection error of the position of approximately 1 cm.

The Table 2 shows how the detection quality varies with the detection thresholds.

Table 2. Object detection results table.

Reject	Unlikely	Likely	Found	Total	% Found
1	2	5	992	1000	99.20

6. Conclusion and Future Work

We addressed the problem of locating a known object within a bounded distance from a robotic arm equipped with one camera in its arm. A planning technique is proposed based on heuristics for reducing the number of snapshot-capturing steps. This heuristic is based on a low-to-high resolution search, where the environment is first scanned at low resolution before exploring with increased resolution the areas where the target object is located with the highest probability.

The original search space is segmented into a set of areas based on the maximum view range of the camera. After a low resolution image is taken of an area, a probability of occurrence is assigned to it. Further, a binary search is used to explore the areas with the highest probability of occurrence for the target object, until a threshold of detection is passed, or the search space is exhausted.

The contingency plan searches an initial location of the target object using a detector that fuses the decision of three sensors: color, shape, and texture. 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 experiments with a ST12 robot using a ST-MC33 camera, the target object (a green box) was detected in average after less than 6 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] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

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