Hypothesis Testing Framework for Active Object Detection

Nikolay Atanasov*, Bharath Sankaran*,
Jerome Le Ny, Thomas Koletschka, George J. Pappas, and Kostas Daniilidis

Abstract— **One of the central problems in computer vision is the detection of semantically important objects and the estimation of their pose. Most of the work in object detection has been based on single image processing and its performance is limited by occlusions and ambiguity in appearance and geometry. This paper proposes an active approach to object detection by controlling the point of view of a mobile depth camera. When an initial static detection phase identifies an object of interest, several hypotheses are made about its class and orientation. The sensor then plans a sequence of view-points, which balances the amount of energy used to move with the chance of identifying the correct hypothesis. We formulate an active M-ary hypothesis testing problem, which includes sensor mobility, and solve it using a point-based approximate POMDP algorithm. The validity of our approach is verified through simulation and experiments with real scenes captured by a kinect sensor. The results suggest a significant improvement over static object detection.**

I. Introduction

With the rapid progress of robotics research, the utility of autonomous robots is no longer restricted to controlled industrial environments. The focus has shifted to high-level interactive tasks in complex environments. The effective execution of such tasks requires the addition of semantic information to the traditional traversability representation of the environment. For example, a household robot needs to be able to detect an object of interest among those scattered on a dining table. For manipulation, it needs to estimate the object's pose accurately.

One of the central problems in computer vision, object detection and pose estimation, historically has been addressed with the assumption that the position of the sensing device is fixed [1], [2], [3]. However, occlusions, variations in lighting, and imperfect object models in realistic environments decrease the accuracy of single-view object detectors.

Active perception approaches circumvent these issues by utilizing appropriate sensing settings to gain more information about the scene. A large body of research in *sensor management* [4] presents a structured approach to controlling the degrees of freedom in sensor systems and satisfying operational constraints while achieving the task objectives. However, most of the work either assumes a simplified model for the detection process [5], [6] or avoids the problem altogether and concentrates on estimating a target's state after its detection [7], [8], [9].

This paper is a step towards bridging the gap between the research in sensor management and the recent advances in 3D object detection enabled by the advent of low cost RGB-D cameras and open-source point cloud libraries [10]. Rather than placing the burden of providing perfect results on a static detector, the sensor can move to increase the confidence in its detection. We consider the following problem. A mobile sensor has access to the models of several objects of interest. Its task is to determine which, if any, of the objects of interest are present on a cluttered table and to estimate their poses. The sensor has to balance the detection accuracy with the time spent observing the objects. The problem can be split into a static detection stage followed by a planning stage to determine a sequence of points of view, which minimize the mistakes made by the observer.

The rest of the paper is organized as follows. The next section provides an overview of related approaches to active object detection and summarizes our contribution. In section III we draw up hypotheses about the class and orientation of an unknown object and formulate the active detection problem precisely. Section IV describes our approach to static detection using a depth camera. In section V we discuss the generation of an observation model, which is used in a Bayesian framework to assign a confidence measure on the hypotheses. Section VI presents a formalism, which allows testing the hypotheses based on the sensor's observations and selecting a sequence of view-points to balance the sensing time with the decision accuracy. Implementation details are discussed in Section VII. Finally, in section VIII we present simulation results and discuss the validity of our approach.

II. Related Work

The approaches in sensor management [4], [11] can be classified according to sensor type into *mobile* (sensors have dynamic states) and *stationary* (sensors have fixed states). Also, the targets of interest might be mobile or stationary. The process of choosing sensor configurations may quantify the utility of the next configuration only (*myopic*) or may

This work was supported by the NSF-IIP-0742304, NSF-OIA-1028009, ARL MAST-CTA W911NF-08-2-0004, ARL RCTA W911NF-10-2-0016, and NSF-DGE-0966142 grants.

* - these authors contributed equally to the work

N. Atanasov and G. Pappas are with the Department of Electrical and Systems Engineering, University of Pennsylvania, Philadelphia, PA 19104, {atanasov, pappasg}@seas.upenn.edu.

B. Sankaran is with the Computational Learning and Motor Control Lab, University of Southern California, Los Angeles, CA 90089, bsankara@usc.edu

J. Le Ny is with the Department of Electrical Engineering, Ecole Polytechnique de Montreal, QC H3C-3A7, Canada, jerome.le-ny@polymtl.ca

T. Koletschka and K. Daniilidis are with the Department of Computer and Information Science, University of Pennsylvania, Philadelphia, PA 19104, thomas.koletschka@gmail.com, kostas@cis.upenn.edu

optimize over a sequence of future sensor configurations (*non-myopic*). Finally, the objective may be to identify a target and estimate its state or simply improve the state estimate of a detected target.

The earliest work in active perception can be attributed to Bajscy [12], [13]. It was focused on 3D position estimation through control of the sensor's intrinsic parameters. Pito's 1999 paper [14] addresses the next best view problem as one that maximizes information gain by increasing spatial resolution. The movement of the sensor is constrained to a circle centered around the object of interest.

The work that is closest to ours [15] uses a mobile sensor to classify stationary objects on a table and estimate their poses. Static detection is performed using SIFT matching. An object's pose distribution is represented with a Gaussian mixture. The authors use a myopic strategy to reduce the differential entropy in the pose and class distributions. This work differs from ours in that the sensor has models of all the objects so the detection is never against background. Moreover, by formulating hypotheses about an object's identity and by choosing a small discrete space for the possible sensor poses, we are able to plan non-myopically.

Velez and coworkers [16], [17] consider the problem of detecting doorways, while a mobile sensor is traveling towards a fixed goal point. The unknown state of a candidate detection is binary: "present" or "not present". Stereo disparity and plane fitting are used for pose estimation. An entropy field is computed empirically for all view-points in the workspace and is used to myopically select locations with high expected information gain. The authors assume that the static detector provides sufficiently accurate pose estimates and do not optimize them during the planning.

In our work we use a depth sensor, which validates the assumption that the position estimate of a stationary object is accurate and does not need to included in the optimization objective. However, the orientation estimates can be improved through active planning. Inspired by the work on hypothesis testing [18], we introduce a rough discretization of the space of orientations so that the hidden object state takes on several values, one for "object not present" and the rest for "object present" with a specific orientation. As a post-processing step, the rough orientation estimate is used to seed a robust alignment procedure, which provides an accurate pose estimate. In our previous work we considered a dual hypothesis problem aimed at model completion [19].

Karasev et al. [20] plan the path of a mobile sensor for visual search of an object in an otherwise known and static scene. The problem statement is different from ours but the optimization is surprisingly similar. The authors hypothesize about the pose of the object and minimize the probability of an incorrect decision. Since different object locations need to be considered, the optimization is intractable. Instead, a mathematical model of the sensing process is used to maximize the conditional entropy of the next measurement.

A lot of the work in sensor management assumes a fixed sensor position, which simplifies the problem considerably because the trade-off between minimizing movement energy and maximizing view-point informativeness is avoided [8], [21]. Often, the action selection process is myopic. In contrast, we consider a mobile sensor, include the detection process in the optimization, and use non-myopic planning. Golovin and Krause [22] showed that myopic planning for an adaptively submodular objective function is merely by a constant factor worse than the optimal strategy. Unfortunately, the objective in our formulation is not adaptively submodular and even with a fixed sensor state, a myopic strategy can perform arbitrarily worse than the optimal policy [18].

The contributions of this paper are two-fold. Firstly, we introduce the idea of implicit pose estimation in 3D object detection by utilizing a vocabulary tree-based partial view matching. In addition to detecting the object's class this approach allows us to retrieve a course pose estimate. Moreover, relying on partial views helps in scenarios in which the object of interest is either partially occluded or in contact with another object. Secondly, we introduce a formal hypothesis testing framework to improve upon the static detection results by moving the sensor to more informative view-points. Our non-myopic planning approach weights the benefit of gaining more certainty about the correct hypothesis against the physical cost of moving the sensor.

III. Problem Formulation

Let the table surface be represented by a bounded set $\mathcal{T} \subset \mathbb{R}^2$. Let B_0 be the possibly infinite set of all object classes that may appear on the table. We assume that each object class has a single model associated with it and use the words model and class interchangeably. Instances are drawn from B_0 at random and are placed uniformly on the table surface. For simplicity we assume that the objects have a random yaw and no pitch or roll so that their poses are in $SE(2)$. The extension of our framework to the $SE(3)$ case is immediate.

Consider a mobile depth sensor, whose position and orientation at time t are $x_t = (x_t^p, x_t^r) \in SE(3)$. Let Ω represent the state of the *static* environment, which includes factors such as scene geometry, lighting, occlusion, etc. At time t, the depth sensor can obtain a point cloud $\mathcal{Q}_t \subset \mathbb{R}^3$ from the scene which is visible from x_t according to $\mathcal{Q}_t = \phi(x_t, \Omega)$. The first task of the sensor is to split $\mathcal{Q}_t$ into separate surfaces (*segmentation*) and associate them with either new or previously observed objects (*data association*). These procedures are not the focus of our paper but we mention briefly how we perform them in Subsection VII-A. We assume that they estimate the object positions accurately.

The sensor has access to a database of size $L_1 < \infty$ of object models $B_1 \subset B_0$ and a subset $B_2 = \{C_1, \ldots, C_{L_2}\} \subseteq B_1$ of them are designated as objects of interest. The task of the sensor is to detect all objects from B_2, which are present on the table and to estimate their pose as *quickly* as possible. Note that the detection is both against known objects from B_1 and unknown background from $B_0 \setminus B_1$.

We are interested in choosing a sequence of view-points for the mobile sensor, which has an optimal trade-off between energy used to move and number of incorrect classifications. Doing this with respect to all objects simultaneously results in a complex joint optimization problem. Instead, we treat the objects independently and process them sequentially, which simplifies the task to choosing a sequence of sensor poses with respect to a single object.

Further, we restrict the motion of the sensor to a sphere $S^2(\rho)$ of radius ρ, centered at the location of the object. The sensor's orientation is fixed so that it points at the centroid of the object. We denote this space of sensor poses by $V(\rho)$ and refer to it as a *viewsphere*. A sensor pose $x \in V(\rho)$ is called a *viewpoint*. As a result, we only need to plan for a sequence of viewpoints. At a high-level planning stage we assume that we can work with a fully actuated model of the sensor dynamics so that for any two poses $x^1, x^2 \in V(\rho)$, there exists a control $u^{1,2} \in U$, which takes the sensor from x^1 to x^2. At time t the sensor can either move and make one more observation or decide on the class and orientation of the unknown object and retire.

As mentioned in Section II, the space of object orientations is discretized sparsely as $\Theta = \{r_1, \ldots, r_N\} \subset SO(2)$. Thus, the hidden variables in the single object optimization are the object class $Y \in B_2$ and the object orientation $R \in \Theta$. The sensor needs to decide between $M = L_2N + 1$ hypotheses:

H_0 : the object does not belong to B_2,
H_i : the object is of class $cl(H_i) := C_{mod(i,L_2)} \in B_2$ with orientation $or(H_i) := r_{(i - cl(H_i))/L_2} \in \Theta$ for $i = 1, \ldots, L_2N$

In order to measure how well the task has been carried out we introduce the following costs:

$$\Lambda_{ij} = \text{ cost for deciding on } H_i\text{, when } H_j \text{ is correct}$$
$$= \begin{cases} d(or(H_i), or(H_j)) & cl(H_i) = cl(H_j) \in B_2 \\ \Lambda_{fn} & cl(H_i) \neq cl(H_j) \in B_2 \\ \Lambda_{fp} & cl(H_i) \in B_2, cl(H_j) \in B_1 \setminus B_2 \\ 0 & \text{if } i = j \end{cases}$$

$$c(x^1, x^2) = d_{SE(3)}(x^1, x^2) + c_0 = \text{ cost of moving from } x^1 \text{ to } x^2 \text{ on the sphere and taking another observation}$$

where $d_{SE(3)}(\cdot, \cdot)$ is a metric on $SE(3)$, $c_0 > 0$ is a fixed measurement cost, Λ_{fp} and Λ_{fn} are costs for making false positive and false negative mistakes respectively, and $d(\cdot, \cdot)$ is a cost for an incorrect orientation estimate, when the class is correct.

Problem 1 (Active Object Detection): Given an object with random class $Y \in B_0$ and orientation $R \in SO(2)$ on the table $\mathcal{T} \ltimes SO(2)$, let $j^*(Y, R)$ denote the index of the hypothesis with the same class and closest orientation. The objective of the mobile sensor is to find a stopping time τ, a sequence of viewpoints $x_0, \ldots, x_{\tau-1}$, and a decision rule $\delta(\mathcal{Q}_{1:\tau}) \in \{0, \ldots, M-1\}$ which minimize the cost:

$$\mathbb{E}_{\mathcal{Q}_{1:\tau}, Y, R}\left\{\sum_{t=0}^{\tau-1} c(x_t, x_{t-1}) + \Lambda_{\delta(\mathcal{Q}_{1:\tau}), j^*(Y,R)}\right\}. \quad (1)$$

Remark 1: The first term in (1) captures the energy spent moving, while the second term is a weighted probability of making a mistake. To see this suppose that all decision costs are equal, i.e. $D := d(or(H_i), or(H_j)) = \Lambda_{fn} = \Lambda_{fp}, \forall i, j$. Then:

$$\mathbb{E}_{\substack{\mathcal{Q}_{1:\tau},\\ Y,R}}\left\{\Lambda_{\delta(\mathcal{Q}_{1:\tau}), j^*(Y,R)}\right\} = \mathbb{E}_{\substack{\mathcal{Q}_{1:\tau},\\ Y,R}}\left\{D\mathbb{1}_{\delta(\mathcal{Q}_{1:\tau}) \neq j^*(Y,R)}\right\}$$
$$= D\mathbb{P}(\delta \neq j^*),$$

which is the probability that decision $\delta(\mathcal{Q}_{1:\tau})$ is incorrect.

Our approach to solving the active object detection problem consists of two stages. First, we use a vocabulary tree to perform static detection in 3D. Since the detection scores are affected by sensor noise and occlusions we don't use them directly. Instead, we use a probabilistic framework to maintain the hypotheses about the detection outcome. In the second stage, we use non-myopic planing to select better viewpoints for the static detector and to update the probabilities of the hypotheses.

IV. Static Object Detection

This section details our static recognition procedure. We use a modified 3D version of a vocabulary tree [23], which is trained using the models in B_1. A training database is generated by extracting a set of templates for each model $\mathcal{M} \in B_1$. To represent random clutter we add composite models to B_1, each of which consist of several common tabletop objects such as a collection of bottles, bowls, vases, etc. A viewsphere $V(\rho)$ is centered around $\mathcal{M}$ and is discretized into a set of viewpoints $V_G(\rho) = \{v_1(\rho), \ldots, v_G(\rho)\} \subset V(\rho)$. A simulated depth sensor is used to extract a pointcloud template from every viewpoint. Thus, our training database is the set $\mathcal{D} = \{\mathcal{P}_{g,l} \mid g = 1, \ldots, G, l = 1, \ldots, L_1\}$ of templates. Features, which describe the local surface curvature are extracted for each template as described below and are used to train a vocabulary tree. Given a query pointcloud at test time, we extract a set of features and use the vocabulary tree to find the template from $\mathcal{D}$, whose features match those of the query the closest.

A. Feature Extraction

First, it is necessary to identify a set of keypoints for each template $\mathcal{P} \in \mathcal{D}$, at which to compute local surface descriptors. Most 3D features are some variation of surface normal estimation, which makes them very sensitive to noise. As a result, using a unique keypoint estimator is prone to errors. To avoid this, we extract a set of keypoints $\mathcal{K}_{\mathcal{P}}$ by sampling the pointcloud $\mathcal{P}$ uniformly. Computing the keypoints over the entire surface accounts for global appearance and reduces noise sensitivity.

Next, neighboring points within a fixed radius of every keypoint are used to compute Fast Point Feature Histograms [24]. The same number of local features is computed at every keypoint since the radius of the support region is fixed. The features are then filtered using a pass-through filter to eliminate badly conditioned ones, which gives the final set $\{f\}_{kp}$ associated with $kp \in \mathcal{K}_{\mathcal{P}}$. The keypoint extraction is shown in Fig 1.

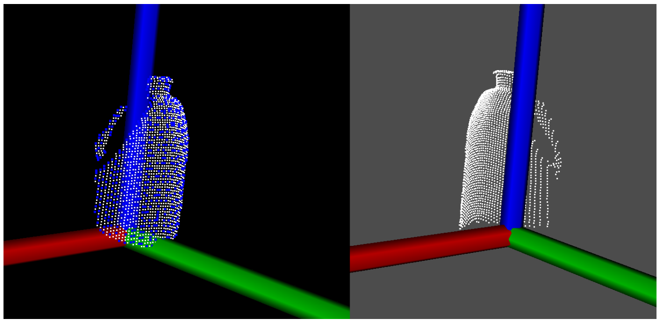


Fig. 1. A query surface (white) with keypoints (blue) extracted using uniform sampling is shown on the left. The best surface returned by the vocabulary tree is shown on the right.

B. Vocabulary Tree Training

The sets of features $\bigcup_{kp \in \mathcal{K}_{\mathcal{P}}} \{f\}_{kp}$ associated with each template $\mathcal{P} \in \mathcal{D}$ are used to train the vocabulary tree. Instead of performing unsupervised clustering on the features, we associate cluster centers with a feature from each of the L_1 models in B_1. After the first set of nodes in the tree is specified, the rest of the cluster centers are computed via hierarchical k-means clustering. During the tree construction, the feature relevance at node i is determined by weighting it with weight w_i based on entropy:

$$w_i = \ln\left(\frac{\eta}{\eta_i}\right),$$

where η is the total number of documents (GL_1) in the tree and η_i is the number of documents which have a descriptor vector passing through node i. The weights are used in the retrieval phase to weight the database descriptors while calculating the relevance score.

C. Vocabulary Tree Performance

Given a query pointcloud $\mathcal{Q}$ at test time, we compute keypoints and extract features using the procedure from IV-A. The features are quantized using the words of the vocabulary tree to create a document descriptor vector q. Descriptor vectors $d_{\mathcal{P}}$ are computed in a similar manner for all templates $\mathcal{P} \in \mathcal{D}$. The query descriptor is propagated down the tree by comparing it with the k cluster centers and choosing the closest one through a nearest neighbor search. The descriptor vectors from the tree are ranked according to a relevance score $s(q, d_{\mathcal{P}})$, which is the normalized difference between the query and a database vector:

$$s(q, d_{\mathcal{P}}) = \left\| \frac{d_{\mathcal{P}}}{\|d_{\mathcal{P}}\|} - \frac{q}{\|q\|} \right\|$$

The document $d_{\mathcal{P}}$ with the lowest relevance score indicates the best matching template from the database.

The performance of the static detector was evaluated by using the templates $\mathcal{D}$ as queries to construct a confusion matrix (See Fig. 2). If the retrieved template matches the model of the query it is considered correct regardless of the viewpoint.

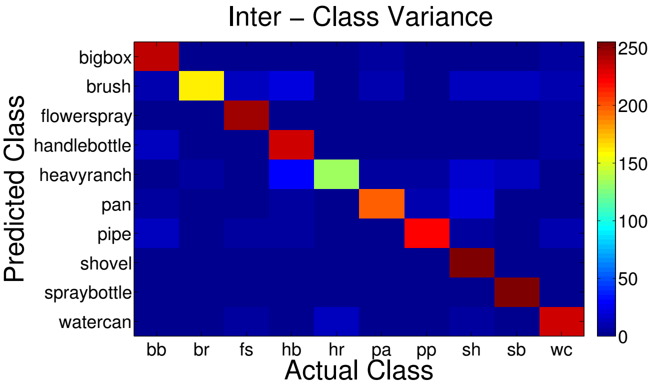


Fig. 2. Confusion matrix for all classes in the vocabulary tree. A class is formed from all views associated with an object.

V. Observation Model

We would like to use a Bayesian framework to maintain probabilities for the object hypotheses. This requires statistics about the operation of the sensor for different object classes, orientations, and viewpoints. Instead of using the segmented pointcloud $\mathcal{Q}_t$ as the observation of the sensor, we take the output of the vocabulary tree. As a result, we deal with the space of possible vocabulary tree outputs rather than the space of all possible pointclouds. Moreover, this includes the operation of the vision algorithm in the sensor statistics.

Given a query pointcloud $\mathcal{Q}_t$ suppose that the vocabulary tree returns template $\mathcal{P}_{g,l}$ as the top match. Assume that the models in the training database are indexed so that those from B_2 have a lower l index than those from $B_1 \setminus B_2$. We take the linear index of the closest match $\mathcal{P}_{g,l}$ as the observation if the match is an object of interest. Otherwise, we record only the model index l, ignoring the viewpoint g:

$$Z_t = \begin{cases} (l-1)G + g, & \text{if } l \le L_2 \\ L_2 G + (l - L_2), & \text{if } l > L_2, \forall g \in \{1, \ldots, G\}. \end{cases}$$

This makes the observation space one dimensional. Given a sensor pose $x \in V(\rho)$ and an object hypothesis H_i, we need to approximate the data likelihood of Z_t:

$$h_i^x(z) := \mathbb{P}(Z_t = z \mid x, H_i)$$

The function h is called the *observation model* of the static detector. It can be obtained off-line since it only depends on the characteristics of the sensor and the vision algorithm. Ideally, the sensing and detection processes should be abstracted to obtain a closed-form representation of h but this is a daunting task for a depth sensor and a vocabulary tree. Instead, we learn a histogram approximating h using the training dataset B_1.

The viewsphere is discretized into a set of viewpoints $\mathcal{X}(\rho) \subset V(\rho)$, which will be used in the planning phase. It need *not* be the same as the set $V_G(\rho)$ used to train the vocabulary tree. We generated 50 random environments from the models in B_1 for each of the 7 hypotheses and used a simulated depth sensor to obtain scores from the vocabulary tree for a set $\mathcal{X}(\rho)$ of 42 viewpoints.

VI. Active Hypothesis Testing

In this section we provide a dynamic programming formulation for the single object optimization problem in (1). As mentioned earlier, we restrict the possible sensor poses to a discrete set $\mathcal{X}(\rho)$ of viewpoints on the viewsphere $V(\rho)$ centered at the unknown object. The state at time t consists of the sensor pose $x_t \in \mathcal{X}(\rho)$ and the *information state*, summarized by the sufficient statistic consisting of the probabilities for each hypothesis:

$$p_i(t) = \mathbb{P}(H_i \mid Z_1 = z_1, \ldots, Z_t = z_t, x_{0:t}) \in [0, 1],$$

where $z_{1:t}$ are the observations from the vocabulary tree. Suppose that at time t the state is $(x_t, p(t))$ and the sensor decides to continue observing by moving to a new viewpoint $x_{t+1} \in \mathcal{X}(\rho)$. The new observation z_{t+1} is used to update the probabilities of the hypotheses according to Bayes' rule:

$$\begin{aligned} p(t+1) &= T(p(t), x_{t+1}, z_{t+1}), \text{ with } i\text{th component:}\\ p_i(t+1) &= \mathbb{P}(H_i \mid z_{1:(t+1)}, x_{0:(t+1)})\\ &= \frac{\mathbb{P}(Z_{t+1} = z_{t+1} \mid x_{t+1}, H_i)\mathbb{P}(H_i \mid z_{1:t}, x_{0:t})}{\mathbb{P}(Z_{t+1} = z_{t+1} \mid x_{t+1})}\\ &= \frac{h_i^{x_{t+1}}(z_{t+1})p_i(t)}{\sum_{j=0}^{M-1} h_j^{x_{t+1}}(z_{t+1})p_j(t)}, \quad \forall i = 0, \ldots, M-1 \end{aligned}$$

using the assumption of independence of successive observations. Supposing that τ is fixed for a moment, the terminal cost of the dynamic program can be derived after the latest observation z_τ has been incorporated in the posterior:

$$\begin{aligned} J_\tau(x_\tau, p(\tau)) &= \min_{\delta \in \{0,\ldots,M-1\}} \mathbb{E}_{Y,R}\Lambda_{\delta, j^*(Y,R)}\\ &= \min_{\delta \in \{0,\ldots,M-1\}} \sum_{j=0}^{M-1} \Lambda_{\delta,j} p_j(\tau) \end{aligned}$$

The intermediate stage costs for $t = 0, \ldots, (\tau - 1)$ are:

$$\begin{aligned} J_t(x_t, p(t)) = \min_{v \in \mathcal{X}(\rho)} \Big\{ & c(x_t, v) +\\ & \mathbb{E}_{Z_{t+1}} J_{t+1}(v, T(p(t), v, Z_{t+1})) \Big\} \end{aligned}$$

Letting τ be random again and t go to infinity we get the following infinite-horizon dynamic programming equation:

$$\begin{aligned} J(x, p) = \min \Big\{ & \min_{\delta \in \{0,\ldots,M-1\}} \sum_{j=0}^{M-1} \Lambda_{\delta j} p_j, \qquad (2)\\ & \min_{v \in \mathcal{X}(\rho)} c(x, v) + \mathbb{E}_Z\{J(v, T(p, v, Z))\} \Big\}, \end{aligned}$$

which is well-posed by Propositions 9.8 and 9.10 in [25]. Equation (2) gives an intuition about the relationship between the cost functions $c(\cdot,\cdot)$, Λ_{ij} and the stopping time τ. If at time t, the expected cost of making a mistake given by $\min_{\delta \in \{0,\ldots,M-1\}} \sum_{j=0}^{M-1} \Lambda_{\delta j} p_j(t)$ is smaller than the cost of taking one more measurement, the sensor stops and chooses the minimizing hypothesis; otherwise it continues measuring.

We resort to numerical approximation techniques, which work well when the state space of the problem is sufficiently small. The decision rule $\delta(\cdot)$ can be replaced by a set of sink states $A = \{a_0, \ldots, a_{M-1}\}$ such that if the sensor goes to state a_i, it decides on hypothesis H_i and remains there for the rest of time. Then, for $s_1, s_2 \in \mathcal{X}(\rho) \cup A$ the cost of movement and the state transition function become:

$$c'(s_1, p, s_2) = \begin{cases} c(s_1, s_2), & s_1, s_2 \in \mathcal{X}(\rho)\\ \sum_{j=0}^{M-1} p_j \Lambda_{s_2,j}, & s_1 \in \mathcal{X}(\rho), s_2 \in A\\ 0, & s_1 = s_2 \in A\\ \infty, & \text{otherwise} \end{cases}$$

$$T'(p(t), s_{t+1}, z_{t+1}) = \begin{cases} T(p(t), s_{t+1}, z_{t+1}), & s_{t+1} \in \mathcal{X}(\rho)\\ p(t), & s_{t+1} \in A \end{cases}$$

We can rewrite (2) into the usual Bellman optimality equation for a POMDP:

$$J(s, p) = \min_{s' \in \mathcal{X}(\rho) \cup A} \Big\{ c'(s, p, s') + \mathbb{E}_Z\{J(s', T'(p, s', Z)\} \Big\}$$

We use a point-based POMDP algorithm [26], [27], which approximates optimally reachable belief spaces in order to solve the problem efficiently and obtain an approximate stationary policy $\hat{\mu} : \mathcal{X}(\rho) \cup A \times [0, 1]^M \to \mathcal{X}(\rho) \cup A$.

VII. Implementation Details

The previous sections developed a procedure for making a decision about the class and orientation of a single object. In this section we present the details of using this procedure to process all objects on the table as quickly as possible.

A. Segmentation and data association

The points $\mathcal{Q}_t$ received from the scene are clustered according to Euclidean distance by using a Kd-tree. An occupancy grid representing the 2D table surface is maintained in order to associate the clustered surfaces with new or previously seen objects. The centroid of a newly obtained surface is projected to the table and compared with the occupied cells. If the new centroid is close enough to an existing object, the surface is associated with that object and the cell is indexed by the existing object ID. Otherwise, a new object with a unique ID is instantiated.

B. Coupling between objects

The optimization in Problem 1 is with respect to a single object but while executing it, the sensor can obtain surfaces from other objects within its field of view. We have the sensor turn towards the centroid and update the hypotheses' probabilities of every visible object. The turning is required because the observation model was trained only for a sensor facing the centroid of the object. Removing this assumption requires more training data and complicates the observation model approximation significantly. The energy used for these turns is not included in the optimization in (1).

The scores obtained from the vocabulary tree are not affected significantly by scaling. This allows us to vary the radius ρ of the viewsphere in order to ease the sensor movement and to update hypotheses for other objects within the field of view. The radius is set to 1 meter by default but if the next viewpoint is not reachable, its can be adapted to accommodate obstacles and the sensor dynamics. Algorithm 1 summarizes the complete hypothesis testing framework.

Algorithm 1 Active Object Detection

```
1: **Input**: Initial sensor pose $x_0 = (x_0^p, x_0^r) \in SE(3)$, object models of
   interest $B_2$, vector of priors $p(0) \in [0,1]^M$ for the $M$ hypotheses
2: **Output**: Decisions $\delta \in \{0, \dots, M-1\}$ for every object on the table
3: Priority queue $pq \leftarrow \emptyset$
4: Current object ID $k^* \leftarrow$ unassigned
5: **for** $t = 0$ to $\infty$ **do**
6:     Obtain a point cloud: $\mathcal{Q}_t \leftarrow \phi(x_t, \Omega)$
7:     Cluster $\mathcal{Q}_t$ and update the table occupancy grid
8:     **for** every undecided object $k$ seen in $\mathcal{Q}_t$ **do**
9:         Rotate the sensor so that $x_t^r$ faces the centroid of $k$
10:        Get viewsphere radius: $\rho \leftarrow \|x_t^p - centroid(k)\|$
11:        Get closest viewpoint: $v^k \leftarrow \arg\min_{v \in \mathcal{X}^k(\rho)} \|x_t^p - v\|$
12:        Obtain a point cloud: $\mathcal{Q}^k \leftarrow \phi(x_t, \Omega)$
13:        Get vocabulary tree score $z^k$ using $\mathcal{Q}^k$
14:        Update probabilities for object $k$: $p^k \leftarrow T(p^k, v^k, z^k)$
15:        **if** $k \notin pq$ **then**
16:            Insert $k$ in $pq$ according to probability $k \in B_2$, i.e. $1 - p_0^k$
17:    **if** $k^*$ is unassigned **then**
18:        **if** $pq$ is not empty **then**
19:            $k^* \leftarrow pq.pop()$
20:        **else**                  ▷ All objects seen so far have been processed.
21:            **if** whole table explored **then**
22:                **break**
23:            **else**
24:                Move sensor to an unexplored area and start over
25:    $x_{t+1} \leftarrow \hat{\mu}(v^{k^*}, p^{k^*})$
26:    **if** $x_{t+1} = a_i \in A$ **then**
27:        $\delta^{k^*} \leftarrow i$, $k^* \leftarrow$ unassigned, Go to line 18
28:    Move sensor to $x_{t+1}$
```

VIII. Performance Evaluation

Object models, constructed using the kinect fusion algorithm from PCL, were used to construct B_0. The performance of the static and the active detectors were compared on synthetic scenes obtained from B_0. The results are summarized in Subsection VIII-A. In Subsection VIII-B, the active framework was evaluated on several real scenes from a lab environment captured with a kinect sensor.

We used a subset of 10 models from B_0 and 2 clutter models to define B_1. Templates were extracted from them and were used to train the vocabulary tree. A single object of interest was used: $B_2 = \{\text{Handlebottle}\}$. The space of object yaws was discretized into 6 bins to formulate hypotheses about the detections:

$$H_0 = \text{The object is } \textit{not} \text{ a Handlebottle}$$
$$H_i = \text{The object is a Handlebottle with yaw } (i-1)60 \text{ deg}$$
$$\text{for } i = 1, \dots, 6$$

A. Static versus active object detection

Fourteen synthetic scenes with 5 objects each were constructed from the models in B_0. The objects were chosen so that there were 10 instances of each hypothesis. The active detection algorithm was used with a simulated depth sensor to make decisions. Twenty five repetitions with different starting sensor poses were carried out for very object. The score from the vocabulary tree after the first observation was recorded as the decision of the static detector. The results are summarized in Fig. 3. The first seven rows of the confusion matrix show the decisions of the static object detector, while the results from the active framework are shown in the last seven rows. There is a marked improvement in the results when the active detection module is used.

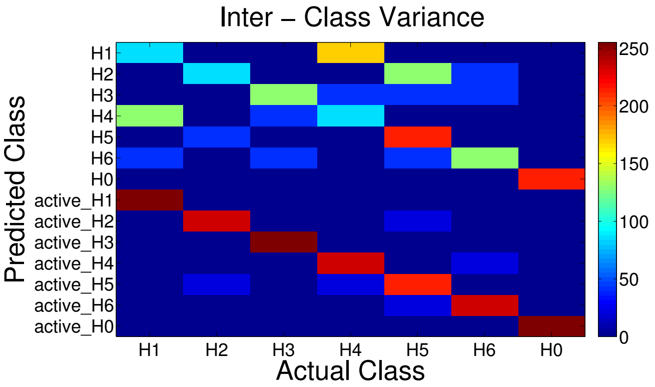


Fig. 3. Confusion matrix comparing the performance of the static and the active object detectors. The columns show the true hypotheses associated with each detection. The first 7 rows present the decisions made by the static detector. The last 7 rows show the decisions made by the active detector

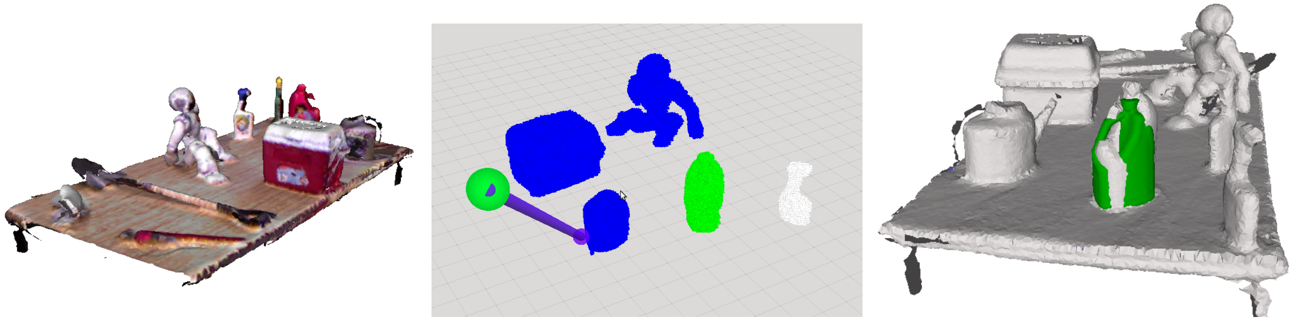
B. Experiments with real scenes

Several real scenes were captured from our lab using a kinect sensor and the fusion algorithm from PCL (See Fig. 4(a)). With B_2 and B_1 the same as before, the sensor's task was to detect any Handlebottles on the table and estimate their orientation. The opreration of the active detection algorithm is exemplified in Fig. 4(b). Since the object orientations in a real scene are not discretized a refinement step is needed if the algorithm detects an object of interest, i.e. decides on $H_i, i > 0$. Then, the last observed pointcloud is aligned with the database template, which corresponds to H_i using an iterative closest point algorithm. Thus, the final decision includes both a class and a continuous pose estimate.

The algorithm colors the pointcloud clusters based on its current understanding of them. Objects, which have been seen but not processed yet are colored red. The object, which is currently under evaluation is colored yellow. Once the system makes a decision about an object, it is colored green if it is of interest, i.e. in B_2, and blue otherwise. Fig. 4(c) shows a detected Handlebottle in the scene. The active framework chose hypothesis H_3, which corresponds to a yaw of 120°. The model associated with H_3 is overlaid on the scene *before* the orientation refinement. As you can see the two objects are already very close and the alignment procedure will produce a good *continuous* orientation estimate.

IX. Conclusion

We consider the problem of detection and pose estimation of semantically important objects versus background using a depth camera. Regardless of the quality of an object detector, the results from static recognition are affected adversely



(a) Real scene captured using kinect fusion (b) Our algorithm's interpretation of the scene (c) The object of interest is detected

Fig. 4. The active detection framework is applied to a real scene. An object is colored green if the algorithm decides it is of interest (in B_2) and blue otherwise. Hypothesis 3 (Handlebottle with yaw 120°) was chosen for the green object in Fig. 4(b). The model associated with H_3 is overlaid on the scene in Fig. 4(c) *before* the final orientation refinement. See the attached video or `http://www.seas.upenn.edu/~atanasov/ICRA2013_ActivePerception.mp4` for more details.

by occlusions, lighting, and scene variations. To address these issues we formulate hypotheses about the class and orientation of an unknown object and propose a soft detection strategy, in which the sensor moves to increase its confidence in the correct hypothesis. A non-myopic planning framework is used to balance the amount of energy spent for sensor motion with the benefit of decreasing the probability of incorrect decision. We demonstrated through simulation that our active detection framework outperforms static detection significantly. Additionally, we tested the algorithm on several real scenes and obtained very promising results.

In future, work we plan to carry out additional real-world experiments in order to reproduce the analysis performed in simulation. On the theoretical side, we would like to investigate the effect of introducing sensor dynamics in the active M-ary hypothesis testing problem in order to obtain a suboptimal policy with performance guarantees.

References

[1] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[2] S. Belongie, J. Malik, and J. Puzicha, "Shape matching and object recognition using shape contexts," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 4, pp. 509–522, 2002.

[3] T.-J. Fan, G. G. Medioni, and R. Nevatia, "Recognizing 3-d objects using surface descriptions," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 11, no. 11, pp. 1140–1157, 1989.

[4] A. O. Hero III and D. Cochran, "Sensor management: Past, present, and future," *Sensors Journal, IEEE*, vol. 11, no. 12, pp. 3064–3075, December 2011.

[5] M. T. J. Spaan, "Cooperative active perception using POMDPs," in *AAAI 2008 Workshop on Advancements in POMDP Solvers*, 2008.

[6] K. L. Jenkins, "Fast adaptive sensor management for feature-based classification," Ph.D. dissertation, Boston University, 2010.

[7] C. Kreucher, K. Kastella, and A. O. Hero III, "Sensor Management Using Relevance Feedback Learning," in *IEEE Transactions on Signal Processing*, 2003.

[8] E. Sommerlade and I. Reid, "Information theoretic active scene exploration," in *Proc. IEEE Computer Vision and Pattern Recognition (CVPR)*, May 2008.

[9] L. Mihaylova, T. Lefebvre, H. Bruyninckx, and J. D. Schutter, "Active robotic sensing as decision making with statistical methods," 2003.

[10] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011.

[11] M. Huber, "Probabilistic framework for sensor management," Ph.D. dissertation, Universität Karlsruhe (TH), 2009.

[12] R. Bajcsy, "Active perception," *Proceedings of the IEEE*, vol. 76, no. 8, pp. 966–1005, aug 1988.

[13] E. Krotkov and R. Bajcsy, "Active vision for reliable ranging: Cooperating focus, stereo, and vergence," *International Journal of Computer Vision*, vol. 11, no. 2, pp. 187–203, 1993.

[14] R. Pito, "A solution to the next best view problem for automated surface acquisition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 21, no. 10, pp. 1016–1030, 1999.

[15] R. Eidenberger and J. Scharinger, "Active perception and scene modeling by planning with probabilistic 6d object poses," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, October 2010, pp. 1036–1043.

[16] J. Velez, G. Hemann, A. S. Huang, I. Posner, and N. Roy, "Active exploration for robust object detection," in *Int. Joint Conf. on Artificial Intelligence (IJCAI)*, Barcelona, Spain, Jul. 2011.

[17] ——, "Planning to perceive: Exploiting mobility for robust object detection," in *Int. Conf. on Automated Planning and Scheduling (ICAPS)*, Freiburg, Germany, Jun. 2011.

[18] M. Naghshavar and T. Javidi, "Active sequential hypothesis testing," 2012. [Online]. Available: arXiv:1203.4626

[19] B. Sankaran, "Sequential hypothesis testing for next best view estimation," Master's thesis, Deapartment of Computer and Information Science, University of Pennsylvania, May 2012.

[20] V. Karasev, A. Chiuso, and S. Soatto, "Controlled recognition bounds for visual learning and exploration," in *Advances in Neural Information Processing Systems*, December 2012.

[21] S. Ekvall, P. Jensfelt, and D. Kragic, "Integrating active mobile robot object recognition and slam in natural environments," in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, October 2006, pp. 5792 –5797.

[22] D. Golovin and A. Krause, "Adaptive submodularity: Theory and applications in active learning and stochastic optimization," *Journal of Artificial Intelligence Research (JAIR)*, vol. 42, pp. 427–486, 2011.

[23] D. Nistér and H. Stewénius, "Scalable recognition with a vocabulary tree," in *CVPR (2)*, 2006, pp. 2161–2168.

[24] R. B. Rusu, "Semantic 3d object maps for everyday manipulation in human living environments," phd, Tecnische Universitatet Muenchen, Munich, Germany, 10/2009 2009.

[25] D. P. Bertsekas and S. E. Shreve, *Stochastic Optimal Control: the discrete-time case*. Athena Scientific, 2007.

[26] H. Kurniawati, D. Hsu, and W. S. Lee, "SARSOP: Efficient Point-Based POMDP Planning by Approximating Optimally Reachable Belief Spaces," *Proc. Robotics: Science and Systems*, 2008.

[27] S. C. W. Ong, S. W. Png, D. Hsu, and W. S. Lee, "POMDPs for Robotic Tasks with Mixed Observability," *Proc. Robotics: Science and Systems*, 2009.