Viewpoint Evaluation for Online 3D Active Object Classification

Timothy Patten[1], Michael Zillich[2], Robert Fitch[1], Markus Vincze[2] and Salah Sukkarieh[1]

***Abstract*—We present an end-to-end method for active object classification in cluttered scenes from RGB-D data. Our algorithms predict the quality of future viewpoints in the form of entropy using both class and pose. Occlusions are explicitly modelled in predicting the visible regions of objects, which modulates the corresponding discriminatory value of a given view. We implement a one-step greedy planner and demonstrate our method online using a mobile robot. We also analyse the performance of our method compared to similar strategies in simulated execution using the Willow Garage dataset. Results show that our active method usefully reduces the number of views required to accurately classify objects in clutter as compared to traditional passive perception.**

***Index Terms*—Object detection, segmentation, categorization; Semantic scene understanding; RGB-D perception**

I. Introduction

ACTIVE perception seeks to control the acquisition of sensor data in order to improve the performance of perception algorithms. Although certain perception tasks such as searching [1] and tracking [2] are routinely cast as active information maximisation problems, *object classification* is traditionally studied as a passive perception problem where data are collected during robot navigation and fed into a perception pipeline. We wish to improve the performance of object classification algorithms, particularly in cluttered and occluded environments, by taking an active approach.

Object classification is a key component of many indoor and outdoor robot systems, and helps to enable many other tasks such as grasping, manipulation, and human-robot interaction. In this work we are directly motivated by domestic scenarios such as tidying a messy room, where static objects lie tangled on the floor [3]. Object classification is necessary for identifying and grasping objects, and the robot must also navigate through clutter while performing its task. A passive perception strategy can be equivalent to choosing viewpoints randomly or acquiring data from a predefined path. We instead hope to reduce the number of viewpoints that are sufficient to classify objects and thus improve the task-level performance of the robot.

Fig. 1: Robot observing a cluttered scene of objects.

In order to develop motion planning algorithms in this case, it is necessary to predict the value of future viewpoints given the information at hand. Such utility prediction must consider many objects and occlusions in the scene and therefore must reason about partial information concerning the number of objects, their class label and pose, and potential occlusions that may limit the value of otherwise information-rich viewpoints.

In this paper we propose a set of algorithms that accurately predict viewpoint quality from RGB-D data. Our algorithms act as a perception pipeline that can be used to plan scanning sequences for classifying unknown objects in cluttered environments. We maintain a set of object hypotheses representing class and pose estimates that are incrementally updated with each new observation. Because objects are treated independently, our method can classify multiple identical objects in the scene. We assume an object database, possibly containing multiple instances of each object class, that stores object instances as point clouds constructed from multiple viewpoints. Online, we use ray tracing to predict which points of an object would be visible from a particular viewpoint given the other objects observed so far and thus compute a utility measure that indicates the discriminatory power of that viewpoint. We propose a utility function that combines these object scores into a global prediction and use this utility function to implement a one-step greedy motion planner. The underlying segmentation, classification, pose estimation, and planning algorithms in our implementation can be readily replaced or extended.

We report two sets of experimental data. The first is a simulated execution of our method compared to several variants using the Willow Garage dataset [4], which contains

Manuscript received: September 1, 2015; Revised November 13, 2015; Accepted November 22, 2015.

This paper was recommended for publication by Editor Tamim Asfour upon evaluation of the Associate Editor and Reviewers' comments.

This research is supported in part by the Australian Centre for Field Robotics; the New South Wales State Government; the Australian Research Council's Discovery Projects funding scheme (project number DP140104203); and the European Community, Seventh Framework Programme (FP7/2007-2013), under grant agreement No. 610532, SQUIRREL.

[1]T. Patten, R. Fitch and S. Sukkarieh are with the Australian Centre for Field Robotics, The University of Sydney, Australia {t.patten, r.fitch, s.sukkarieh}@acfr.usyd.edu.au

[2]M. Zillich and M. Vincze are with the Vision4Robotics Group at the Automation and Control Institute, Technical University Vienna, Austria {zillich, vincze}@acin.tuwien.ac.at

Digital Object Identifier xxxxxxxxxxxxxxx

RGB-D images of occluded objects. The second was collected by executing our method online using the mobile robot and representative setup shown in Fig. 1. Compared to passive perception, our method reduced the number of views required for classification by 36% in the Willow Garage dataset and demonstrated a similar trend in the online validation.

The contribution of this work is an end-to-end method for active object classification with multiple occluded objects from RGB-D data. Our method is task-oriented in the sense that it is designed to quickly gather information about many objects simultaneously as opposed to individually. The technical novelty lies in our approach for modulating predictions of viewpoint utility learned offline with online prediction of occlusions and a utility function for planning that jointly considers all objects and possible states. The benefit of our approach is demonstrated through experiments that show a reduction in the number of views required for scene understanding as compared to similar methods. Source code used in our experiments is available online: https://github.com/squirrel-project.

II. Related Work

Research in active perception has a long history in robotics and most recently is motivated in part by the now common availability of inexpensive RGB-D sensors. Closely related work by Wu et al. [5] focuses on a sensor-object model from RGB-D data that characterises viewpoint and visibility for predicted feature matching. Like our method, ray tracing through an occupancy grid is used to "filter" out features that will be occluded. The main difference, however, is that we do not predict which features will be seen but instead use a learnt measure of entropy to quantify the quality of a viewpoint and modulate the expected benefit by the predicted occlusions. Our work also differs in that it exploits the probabilistic output of our classifier to reason about all possible measurements with their uncertainties, rather than only assuming a single (most likely) class for each object. Lastly, we combine the predicted utility of each object, with their uncertainties, in a joint objective function such that the planner maximises the total information.

Historically, active perception has mainly been studied within the computer vision community. A good survey can be found in Chen et al. [6]. For classification, a common approach is to formulate the task as a state estimation problem and solve it with information-theoretic metrics in a sequential recognition process [7], [8], [9]. Because evaluating differential entropy or mutual information is intractable in this context due to the lack of closed form solutions, approximations relying on analytical bounds [7] or Monte Carlo evaluation [8] are often used. Recent work considers choice of classifier, in addition to viewpoint, in a sequential decision-making framework [10]. Other work explores robust active object detection [11] and active object recognition with environment interaction [12]. These examples provide end-to-end frameworks, but only consider single objects and do not account for occlusions when planning.

An alternative approach is to formulate the problem as a partially observable Markov Decision Process (POMDP). Approaches to maintain tractable solutions include employing an upper bound on the value function [13] and point-based approximation algorithms [14]. These methods are shown to improve recognition performance in terms of the number of views and the distance of the travelled path. Our method also represents object states probabilistically, but gains efficiency in a different way. We model occlusions explicitly to modulate offline viewpoint utility measurements. The methods in [13] and [14] include occlusion as part of the belief state, which increases the size of the problem.

The general problem of *passive* object classification is important in robotics and computer vision. For RGB-D data, methods exist for recognising many instances from a variety of classes [15]. Work addressing the multi-view case [16], [17] has shown that merging data from different views leads to improved results, but also highlights the importance of viewpoint selection. From a theoretical perspective, [18] provides a mathematical description of the tradeoff between recognition and planning, which serves as a justification for active viewpoint selection as we explore here.

III. Problem Formulation

The aim of the active classification problem is for a robot to determine the class and pose of static objects in an unknown environment. The robot processes point cloud observations captured with an onboard sensor and selects the next location to make an observation.

At stage k, a robot has a belief $\mathcal{B}_k$ about object states, where a belief $b_k^n \in \mathcal{B}_k = \{b_k^1, b_k^2, ..., b_k^{N_k}\}$ is maintained for each object $n \in \mathcal{N}_k = \{1, 2, ..., N_k\}$. Each belief consists of a class estimate and corresponding pose estimate. The class of an object is represented by the variable $x \in \mathcal{X} = \{x_1, x_2, ..., x_M\}$, where $\mathcal{X}$ is the known set of hand-labelled object classes of size M. Pose estimates $\mathbf{Q}_k^{n,x} \in \mathcal{Q}_k^n$ for each class define an affine transformation of a model to the environment coordinate system ϕ.

From location $\mathbf{y}_k$, the robot makes a point cloud observation $\mathcal{Z}_k$ and stores it in a global 3D occupancy grid $\mathcal{G}$. The observation is partitioned into subsets $\mathbf{z}_k^n \in \mathcal{Z}_k$ where each subset pertains to an object. The set of observations of object n up to stage k is denoted by $\mathbf{z}_{1:k}^n = \{\mathbf{z}_1^n, \mathbf{z}_2^n, ..., \mathbf{z}_k^n\}$ and the state of the object is expressed by the categorical conditional probability distribution $p_k^n(x|\mathbf{z}_{1:k}^n)$, where $\sum_{x\in\mathcal{X}} p_k^n(x|\mathbf{z}_{1:k}^n) = 1$ and $p_k^n(x|\mathbf{z}_{1:k}^n) \geq 0$. Thus the belief of object n is represented by $b_k^n = \{p_k^n(x|\mathbf{z}_{1:k}^n), \mathcal{Q}_k^n\}$.

Observing object n from a candidate location $\mathbf{y}'_{k+1} \in \mathcal{Y}_k$ is quantified by the utility value $u(\mathbf{y}'_{k+1}, b_k^n)$. Objects are assumed to be independent such that the total utility function is the summation of the utilities,

$$U(\mathbf{y}'_{k+1}, \mathcal{B}_k) = \frac{1}{N_k} \sum_{n\in\mathcal{N}_k} w_n u(\mathbf{y}'_{k+1}, b_k^n), \tag{1}$$

where w_n is a weighting factor for object n. This weighting factor will be discussed further in Sec. V.

The set of candidate viewpoints is $\mathcal{Y}_k = \mathcal{Y}_0 \backslash \mathbf{y}_{1:k}$, where $\mathcal{Y}_0$ is the initial set of available locations and $\mathbf{y}_{1:k} = \{\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_k\}$ is the history of visited locations. The next location is chosen as the one which yields the largest utility.

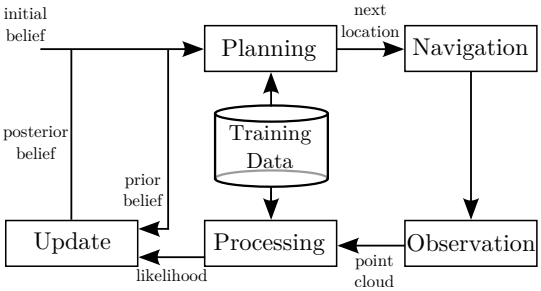


Fig. 2: Overview of the system components.

The active classification problem is defined as follows. Given observations $\mathcal{Z}_{1:k} = \{\mathcal{Z}_1, \mathcal{Z}_2, ..., \mathcal{Z}_k\}$ made from visited locations $\mathbf{y}_{1:k}$ and the current belief $\mathcal{B}_k$ at stage k, choose the next location $\mathbf{y}_{k+1}$ from the set of available locations $\mathcal{Y}_k$ by maximising the utility function

$$\mathbf{y}^*_{k+1} = \underset{\mathbf{y}'_{k+1} \in \mathcal{Y}_k}{\arg\max}\, U(\mathbf{y}'_{k+1}, \mathcal{B}_k). \tag{2}$$

IV. System Overview

An overview of our system is illustrated in Fig. 2. At each stage, a *planning* module uses the current belief and training data to determine the next location to make an observation. The target location is passed to a *navigation* module which drives the robot to the desired goal. At this location, the robot makes an *observation* of the world, which is input to a *processing* module to determine an intermediate belief. This belief is combined with the prior belief in the *update* module to generate a posterior belief that contains all the information from past observations. Finally, the process repeats by planning the next observation with the new belief.

V. Viewpoint Evaluation for Planning

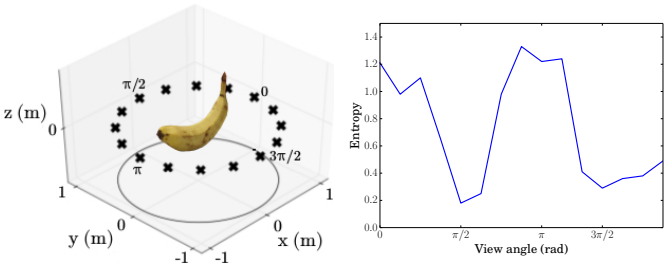
This section describes the planning and navigation modules of the system. These modules comprise methods for assessing viewpoint quality offline, predicting occlusions online, defining a global utility function and selecting (and navigating to) future viewpoints, which are integral to solve the overall objective function (2).

A. Offline Viewpoint Quality

An offline training phase determines a mapping of scalar utility values to viewpoints with respect to an object model. During training, model instances are observed from locations on a 3D view sphere. The quality of each viewpoint is determined by the Shannon entropy of the class probability distribution that is generated from classifying the acquired point cloud. (Classification is explained later in Sec. VII-B.)

An example of the variation in classification performance from different perspectives is shown by the entropy of the predicted class distributions in Fig. 3b for the observation locations around a banana in Fig. 3a. As can be seen, certain views classify the banana with a high confidence (low entropy) whereas other views do not discriminate it well from other objects in the training set.

Formally, each model in the training database is a hand-labelled instance of a class in a local coordinate system with its



(a) (b)

Fig. 3: A *banana* instance observed in training phase: (a) viewpoints and (b) entropy of the class distribution after classification for each viewpoint.

centre located at the origin. The models are viewed from $\tilde{N}_T$ locations evenly distributed around a 3D view sphere with constant radius. In this work we assume that the sensor is always oriented normal to the object centroid, which implies that each view location has a nominal orientation value. The set of view locations $\tilde{\Upsilon}^x_i = \{\tilde{\boldsymbol{\upsilon}}^x_{i,1}, \tilde{\boldsymbol{\upsilon}}^x_{i,2}, ..., \tilde{\boldsymbol{\upsilon}}^x_{i,\tilde{N}_T}\}$ for class x and instance i generates the set of point clouds $\tilde{\mathcal{Z}}^x_i = \{\tilde{\mathbf{z}}^x_{i,1}, \tilde{\mathbf{z}}^x_{i,2}, ..., \tilde{\mathbf{z}}^x_{i,\tilde{N}_T}\}$. For clarity, the notation $\tilde{\cdot}$ represents any training data. Lastly, the point clouds $\tilde{\mathbf{z}}^x_{i,j} \in \tilde{\mathcal{Z}}^x_i$ for instance i are combined and downsampled to obtain a full model point cloud $\tilde{\boldsymbol{Z}}^x_i$.

The entropy of the class distribution $\tilde{p}^x_{i,j}(x|\tilde{\mathbf{z}}^x_{i,j})$ for viewpoint $\tilde{\upsilon}^x_{i,j}$ of class x and instance i is defined by

$$\tilde{e}^x_{i,j} = \tilde{H}(X) = -\sum_{x \in \mathcal{X}}^{n} \tilde{p}^x_{i,j}(x|\tilde{\mathbf{z}}^x_{i,j}) \log(\tilde{p}^x_{i,j}(x|\tilde{\mathbf{z}}^x_{i,j})). \tag{3}$$

After all entropy values are computed for a single instance, the utility of viewpoint $\tilde{\upsilon}^x_{i,j}$ is computed by

$$\tilde{u}^x_{i,j} = 1 - \frac{\tilde{e}^x_{i,j}}{\tilde{e}^x_{i,\max}}, \tag{4}$$

where the entropy value is scaled to the interval $[0, 1]$ by dividing the entropy of the viewpoint with the maximum entropy value $\tilde{e}^x_{i,\max}$ for instance i. Subtracting the entropy from 1 then assigns a high utility to locations with low entropy and low utility to locations with high entropy.

B. Predicting Occlusions

In a cluttered environment, occlusions will have a significant impact on classification performance. This is accounted for by evaluating the expected surface area of the objects from the candidate locations. The number of visible surface points, ζ_{surf}, for instance i of class x is determined by casting a ray from each point of the full model point cloud $\tilde{\boldsymbol{Z}}^x_i$ through a local occupancy grid of the model to the candidate location $\mathbf{y}'_{k+1}$. Each visible surface point is then traced through the world occupancy grid $\mathcal{G}$ to compute the number of *unoccluded* points, ζ_{unocc}. Thus we define the *occlusion proportionality factor* as the function

$$\rho(\mathbf{y}'_{k+1}, \tilde{\boldsymbol{Z}}^x_i) = \frac{\zeta_{\text{unocc}}}{\zeta_{\text{surf}}}, \tag{5}$$

which takes as input the candidate location and a model point cloud and computes the ratio of the number of unoccluded points to the number of visible surface points.

C. Global Utility Function

The utility of observing object n from a candidate location $\mathbf{y}'_{k+1}$ is calculated by

$$u(\mathbf{y}'_{k+1}, b^n_k) = \frac{1}{M} \sum_{x \in \mathcal{X}} \Big[p^n_k(x|\mathbf{z}^n_{1:k}) \times \frac{\rho(\mathbf{y}'_{k+1}, \tilde{\boldsymbol{Z}}^x_I)}{\tilde{N}_T} \sum_{j \in \tilde{N}_T} \tilde{u}^x_{I,j} e^{-d(\mathbf{y}'_{k+1}, \boldsymbol{\tau})/\sigma} \Big], \quad (6)$$

where $d(\cdot,\cdot)$ is the Euclidean distance between two locations, $\boldsymbol{\tau} = \mathbf{Q}^{n,x}_k \tilde{\boldsymbol{\upsilon}}^x_{I,j}$ is the transformation of the viewpoint into the map frame and σ is a scalar variance. Note that the subscript I on the viewpoint $\tilde{\boldsymbol{\upsilon}}^x_{I,j}$, the utility value $\tilde{u}^x_{I,j}$, and the model point cloud $\tilde{\boldsymbol{Z}}^x_I$ represent the training instance of class x that has the strongest belief. The term $e^{-d(\mathbf{y}'_{k+1}, \tau)/\sigma}$ scales the utility contribution of viewpoint $\tilde{\boldsymbol{\upsilon}}^x_{I,j}$ by its distance from the candidate location, which accounts for both angular and distance error between $\mathbf{y}'_{k+1}$ and τ. This term makes the reasonable assumption that the utility value is continuous between different training viewpoints. More sophisticated methods (e.g., Gaussian processes) could be used to model the true relationship but this is beyond the scope of this paper.

This function computes the utility of the known viewpoints, weighted by their distance to $\mathbf{y}'_{k+1}$, and weights the contribution of each possible class by its probability. The utility values are also modulated by the occlusion proportionality factor. Intuitively, if an object is highly occluded from location $\mathbf{y}'_{k+1}$ then its utility contribution will be very small because $\zeta_{\text{unocc}} \ll \zeta_{\text{surf}}$. From such an occluded viewpoint, it is expected that the object will not be strongly recognised and the observation will not be beneficial. If an object is unoccluded then the utility contribution will approach the training utility value because $\zeta_{\text{unocc}} \approx \zeta_{\text{surf}}$. The occlusion factor is contained *within* the summation over the class distribution such that an occlusion state is computed separately for each class.

We define the object weighting factor w_n in (1) by the entropy of the object's probability distribution, i.e.

$$w_n = H(X^n) = -\sum_{x \in \mathcal{X}} p^n_k(x|\mathbf{z}^n_{1:k}) \log(p^n_k(x|\mathbf{z}^n_{1:k})). \quad (7)$$

This means that an object which the robot is unsure about (high entropy) will have more influence on the decision than an object that a robot is more sure about (low entropy).

Substituting (6) and (7) into (1) yields the utility function

$$U(\mathbf{y}'_{k+1}, \mathcal{B}_k) = \frac{1}{N_k} \sum_{n \in \mathcal{N}_k} \left[\frac{H(X^n)}{M} \times \sum_{x \in \mathcal{X}} \left[p^n_k(x|\mathbf{z}^n_{1:k}) \frac{\rho(\mathbf{y}'_{k+1}, \tilde{\boldsymbol{Z}}^x_I)}{\tilde{N}_T} \sum_{j \in \tilde{N}_T} \tilde{u}^x_{I,j} e^{-d(\mathbf{y}'_{k+1}, \tau)/\sigma} \right] \right]. \quad (8)$$

D. Viewpoint Selection and Navigation

The procedures outlined above determine a utility score for a candidate location $\mathbf{y}'_{k+1}$. Selecting the next best view involves computing the utility value for each potential viewpoint $\mathbf{y}'_{k+1} \in \mathcal{Y}$, then selecting and immediately navigating to the viewpoint which has the highest expected utility. This work assumes a given set of initial viewpoints $\mathcal{Y}_0$ with associated collision-free navigation roadmap, and removes each location $\mathbf{y}_k$ from the set as they are visited such that $\mathcal{Y}_k = \mathcal{Y}_0 \backslash \mathbf{y}_{1:k}$. At each stage, the remaining unvisited viewpoints are evaluated to search for the next best view.

VI. Fusing New Observations

In this section we present the update module of the system. This module associates observations to objects and updates class distributions and pose estimates.

A. Data Association

Each observation is stored in a 3D occupancy grid $\mathcal{G}$. The occupied space of each object n can be computed by determining the voxels that correspond to the observations $\mathbf{z}^n_{1:k-1}$. The set of voxels for each new segment is compared to the voxel set of each object in the prior belief $\mathcal{N}_{k-1}$. A segment is associated to an object if the proportion of voxels overlapping those of a prior object is greater than a given threshold. The proportion is computed as the sum of the number of overlapping voxels divided by the total number of occupied voxels for the segment. Set $\mathcal{N}_k$ is maintained by adding new objects for unassociated segments and merging associated segments with existing objects as appropriate.

This procedure can become costly because it must perform N^2V^2 voxel checks where N is the number of hypotheses and V is the number of voxels occupied by an object. We speed up the process by first comparing the bounding boxes of the object point clouds in the two different sets. Voxel matching is then only performed with the objects that pass the quick bounding box check. This greatly reduces the number of voxel checks.

Note that our association method assumes that objects are static and that the localisation and measurement errors are small. In our experiments, the errors were not large enough to cause problems, however, for situations with large error, more robust association methods could be used such as [19].

B. Class Distribution Update

The probability class distribution of object n is updated by applying Bayes' rule

$$p^n_k(x|\mathbf{z}^n_{1:k}) = \eta p^n_{k-1}(x|\mathbf{z}^n_{1:k-1}) p^n_k(\mathbf{z}^n_k|x), \quad (9)$$

where $p^n_{k-1}(x|\mathbf{z}^n_{1:k-1})$ is the prior, $p^n_k(\mathbf{z}^n_k|x)$ is the likelihood and η is a normalisation constant. The prior for a new observation segment is the combination of the probability distributions of its associated objects. The previous observations are considered independent, so the probabilities can be computed by multiplying each element and normalising.

In this work we assume a classifier that can be queried for a single observation $\mathbf{z}^n_k$ to return a probability distribution over classes $p^n_k(x|\mathbf{z}^n_k)$. From Bayes' rule, the output can be expressed as

$$p^n_k(x|\mathbf{z}^n_k) = \eta_0 p^n_k(x|\mathbf{z}^n_0) p^n_k(\mathbf{z}^n_k|x), \quad (10)$$

where η_0 is a normalisation constant different from η. For a single point cloud query, the prior contains no observations, i.e.

$\mathbf{z}_0^n = \emptyset$, and assuming a uniform prior, $p_k^n(x|\mathbf{z}_0^n) = p_k^n(x) = uniform$, (10) is equivalent to

$$p_k^n(x|\mathbf{z}_k^n) = p_k^n(\mathbf{z}_k^n|x), \tag{11}$$

meaning that the likelihood $p_k^n(\mathbf{z}_k^n|x)$ in (10) uses the output of a classifier, where observation $\mathbf{z}_k^n$ is used as its input.

C. Pose Estimate Update

The planning module requires a pose estimate for each object-class pair. In principle, the pose estimate from the latest observation could be used, but it is likely that poor estimates are made when objects are only partially observed.

We assume a pose estimator that can output a measure of estimation quality. In our implementation, this measure is provided by the alignment error of the iterative closest point (ICP) algorithm [20]. To provide a better pose estimate, $\mathbf{Q}_{k-1}^{n,x}$ (for object n and class x) is replaced if the ICP error is larger than the ICP error of the new pose estimate from the latest observation. The pose estimates of each class are considered independent, which means that each estimate can be replaced irrespective of other estimates. This procedure obtains the new set of poses $\mathcal{Q}_k^n$ for each object $n \in \mathcal{N}_k$.

VII. Point Cloud Processing

This section describes our implementation of the processing module of the system. This module can be implemented using any perception tools that output a set of partitioned point clouds with associated class probability distributions and pose estimates.

A. Segmentation

Segmentation is performed using the method in [21]. The scene is first decomposed into surface patches, fitting planes and Non-Uniform Rational B-splines (NURBS). A series of perceptual grouping principles is then run to establish pairwise probabilities of two surfaces belonging to the same object. These probabilities generate edge weights in a graph of surface patches, and a graph cut algorithm is used to optimally partition the graph to yield object hypotheses. The set of object segments is then post-processed to cull background objects (those not lying directly on the ground plane/table top). This segmentation method also handles the case of touching or stacked objects; see [21] for more details.

B. Classification

Classification is performed using [22]. The classifier is built offline by generating partial point clouds of object models in a pre-defined database. The database consists of a large number of model instances which are grouped into classes. Partial point clouds are generated from given locations on a 3D view sphere and a global feature descriptor is computed for each point cloud. Here, we use the ensemble of shape functions (ESF) descriptor, which consists of ten 64-bin histograms based on distinct shape functions (distance, angle and area distributions) [23], although any global feature descriptor could be used. The descriptors for each viewpoint and instance, along with the class label, are stored in a k-d tree with dimension 640. Online classification of a point cloud proceeds by computing the ESF descriptor and calculating the distance in the k-d tree to the closest N_c nodes. The number of nodes for each class are tallied to determine a score, which is then normalised by dividing by N_c, allowing the output to be interpreted as a probability.

Note that the object models used for training the classifier are the same as those used for computing the offline utility values. Thus, the predicted utilities are directly related to the classifier and accurately predict the belief updates.

C. Pose Estimation

The pose estimate for each class x is computed separately from the segmented point clouds $\mathbf{z}_k^n$, where a pose estimate is an affine transformation matrix of a training model point cloud into the world coordinate system ϕ. Each $\mathbf{z}_k^n$ is aligned with training point cloud $\tilde{\mathbf{Z}}_{\text{best}}^x$ corresponding to the most likely instance of class x as follows:

(i) **Downsampling:** The point clouds are downsampled using a uniform grid to speed up computation.

(ii) **Initial scaling:** The ESF descriptor is scale independent, therefore the training models can be a different size to the observed objects. $\tilde{\mathbf{Z}}_{\text{best}}^x$ is expanded or contracted such that the dimensions of its minimum bounding box are similar to the dimensions of the minimum bounding box of $\mathbf{z}_k^n$.

(iii) **Initial alignment:** The training models are viewed in a local coordinate system with the model centred at the origin. $\tilde{\mathbf{Z}}_{\text{best}}^x$ is positioned at the origin of ϕ to bring the point clouds into a common reference frame, then translated so that its centroid has the same coordinate as the centroid of $\mathbf{z}_k^n$.

(iv) **ICP and scale refinement:** The partial point cloud is aligned to the test point cloud using ICP. ICP is seeded from 4 different rotations (0, $\pi/2$, π, $3\pi/2$) on each axis and the alignment with the smallest ICP error is retained. We also make scale refinements with each iteration by performing a grid search over scale. If any new scale has a smaller error, it is maintained for the next iteration.

The modified ICP algorithm that performs scale adjustments is necessary to improve upon the initial scale guess and to take into account occlusions. As an example, a point cloud from an occluded view is likely to be smaller than if it were observed without any occlusion. The fine scale adjustment will bring the test point cloud to a size that better represents the observed point cloud.

VIII. Results

In this section we present two sets of results. First, we present quantitative experiments using the Willow Garage dataset with comparison to related methods. Second, we provide validation and demonstration with a mobile robot performing online active object classification using our method in two different setups and compare its performance with passive perception.

A. Willow Garage Dataset

1) Experimental Setup: The Willow Garage dataset consists of Kinect RGB-D sensor data of household objects on a table top. An example is shown in Fig. 4a. The dataset comprises 24 scenarios, but we rejected 5 because they contain glass objects which could not be detected with the sensor (6, 22, 23, 24) or segmentation failed (16). Each scenario contains 6 objects viewed from $11-19$ different perspectives. For each observation, the viewpoint in the map frame was extracted so that the behaviour of the robot selecting real world observations could be simulated.

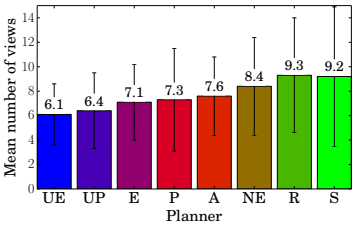
The classifier was trained on 15 classes, where each class comprised $5-20$ instances. The viewing radius was set to 2m and each instance was viewed from $\tilde{N}_T = 20$ locations. We set $\sigma = 0.5$m, which is reasonable since locations closer than 0.1m to a training viewpoint will contribute approximately 80% of its utility value, while locations further than 1m will contribute less than 15% of their utility values. The number of nearest neighbours in the k-d tree search was set to 50 and the voxel overlap threshold for data association was set to 20%.

2) Comparison Methods and Metric: We evaluated the effectiveness of our proposed utility function by comparing it to a variety of alternatives. The full set of methods were:
(UE) Our method using weighted contribution of training viewpoint utility and occlusion reasoning.
(UP) Using the training viewpoint that has the highest probability of the true class. That is, given the probability vector $\tilde{\boldsymbol{p}}^x_{i,j}(x)$, the utility of training viewpoint $\tilde{\boldsymbol{v}}^x_{i,j}(x)$ is quantified by its classification performance w.r.t. the true class, i.e. $\tilde{p}^x_{i,j}(x_{\text{true}})$. The utility values $\tilde{\mu}^x_{i,j}$ in (6) were replaced with $\tilde{p}^x_{i,j}(x_{\text{true}})$ to compute (8).
(E) Our method without occlusion reasoning, so that viewpoints are only selected by their classification ability.
(P) Variant (UP) without occlusion reasoning.
(A) Selecting viewpoints which maximise the expected surface area. The class weight w_n in (1) defined by the entropy of the object was maintained, but $u(\mathbf{y}'_{k+1}, h^n_k)$ was replaced by the number of expected surface points for object n.
(NE) Selecting viewpoints nearest to the training viewpoint with maximum utility for the most uncertain object.
(R) Passively selecting views at random.
(S) Passively selecting sequential viewpoints on perimeter.

The sequential method (S) is typical of a passive perception strategy that is driven by a navigation goal (drive around a perimeter of the scene). Here, viewpoints were selected in clockwise order. The random strategy (R) is also a useful comparison because it avoids any bias induced by the starting view of a sequential strategy.

Each simulation began from the same randomly selected viewpoint and was terminated at the "knee" of the mean probability curve, which was determined where the curve did not increase by more than 0.5% for 4 consecutive observations. The "knee" point with lowest confidence was selected as the *threshold* for the comparison. The performance of each strategy was measured by the number of views required for the mean probability curve to reach the confidence threshold. Typical values of the threshold ranged between $70-80\%$, which is a reasonable level of confidence for occluded data.

(a)



(b)

Fig. 4: Example comparison of planning strategies for one scenario in Willow Garage dataset: (a) the experimental setup and (b) the histogram of the mean number of views to reach confidence threshold.

	UE	UP	E	P	A	NE	R	S
01	**4.7 (1.4)**	5.1 (1.2)	**4.7 (1.4)**	5.1 (1.2)	5.8 (1.9)	5.5 (1.9)	5.9 (2.8)	7.7 (3.7)
02	4.8 (2.5)	8.4 (5.0)	4.8 (2.0)	6.0 (3.7)	**3.9 (1.6)**	5.0 (2.2)	5.9 (1.7)	7.8 (4.0)
03	**4.2 (1.2)**	4.6 (1.3)	4.4 (1.1)	4.4 (2.3)	6.0 (2.9)	5.5 (2.6)	6.4 (2.1)	7.6 (3.6)
04	5.8 (4.2)	**5.3 (2.6)**	7.6 (4.3)	10.2 (4.9)	7.3 (4.4)	6.3 (3.7)	6.3 (2.7)	5.9 (4.4)
05	6.0 (1.7)	5.5 (1.6)	7.5 (3.1)	6.6 (2.7)	7.5 (2.9)	**5.4 (2.3)**	5.5 (2.4)	7.8 (4.4)
07	5.3 (2.1)	**4.7 (1.2)**	6.4 (1.6)	6.0 (1.6)	5.8 (2.8)	7.1 (3.4)	5.5 (3.2)	7.2 (6.3)
08	**6.1 (2.5)**	6.4 (3.1)	7.1 (3.1)	7.3 (4.2)	7.6 (3.2)	8.4 (4.0)	9.3 (4.7)	9.2 (5.7)
09	5.1 (1.7)	5.6 (3.6)	**4.8 (2.2)**	**4.8 (1.8)**	5.5 (2.4)	5.0 (1.3)	5.6 (3.1)	10.5 (5.0)
10	**4.7 (2.1)**	6.5 (4.1)	9.1 (5.5)	7.1 (4.7)	8.0 (5.0)	8.2 (5.5)	10.7 (4.0)	10.8 (4.1)
11	**3.9 (1.3)**	**3.9 (1.9)**	4.0 (1.8)	4.0 (1.6)	4.2 (2.3)	4.5 (1.7)	4.3 (1.8)	7.8 (2.8)
12	9.1 (5.8)	8.0 (5.6)	10.2 (6.2)	10.6 (6.3)	10.0 (6.4)	**7.0 (4.6)**	9.9 (5.7)	8.6 (3.0)
13	**5.6 (1.2)**	6.5 (1.5)	7.1 (2.0)	6.5 (1.8)	7.4 (1.3)	9.5 (4.1)	9.7 (3.3)	11.6 (5.2)
14	3.5 (1.3)	**3.1 (0.7)**	3.5 (0.8)	3.6 (1.1)	3.3 (0.9)	3.3 (0.9)	5.4 (1.7)	7.6 (5.0)
15	**8.8 (3.4)**	10.3 (5.0)	10.3 (4.4)	10.3 (3.9)	10.5 (4.8)	9.8 (3.9)	9.8 (5.2)	15.4 (1.8)
17	**2.7 (0.5)**	**2.7 (0.5)**	**2.7 (0.5)**	**2.7 (0.5)**	**2.7 (0.5)**	3.4 (1.6)	4.3 (1.4)	4.5 (1.9)
18	8.4 (3.6)	**8.2 (4.2)**	7.8 (3.7)	8.6 (4.4)	9.0 (3.7)	11.6 (3.5)	8.5 (3.4)	14.9 (0.3)
19	7.7 (3.7)	8.1 (4.4)	8.3 (3.5)	8.6 (3.9)	7.6 (3.2)	9.3 (4.1)	11.8 (3.6)	**7.5 (4.3)**
20	**4.3 (1.1)**	5.5 (1.9)	4.4 (1.3)	5.4 (3.8)	4.5 (1.3)	6.4 (3.7)	11.5 (3.1)	5.7 (3.9)
21	**3.1 (1.0)**	3.7 (2.0)	4.4 (2.1)	4.1 (2.0)	4.0 (1.7)	4.2 (2.2)	4.6 (2.0)	3.6 (1.6)
mn	**5.5 (1.9)**	5.9 (2.0)	6.3 (2.3)	6.4 (2.4)	6.3 (2.2)	6.6 (2.3)	7.4 (2.6)	8.5 (3.1)

TABLE I: Number of views to reach confidence threshold in the Willow Garage dataset. Rows correspond to dataset scenarios and columns correspond to planning strategies (best planner shown in bold). The last row shows mean number of views with standard deviations in parentheses.

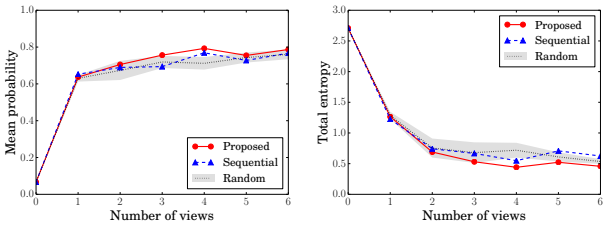
3) Results Discussion: Average results from 10 simulations (from 10 different starting locations) are shown in Fig. 4b for the setup in Fig. 4a. This example shows that all active strategies recognise the objects faster than the random and sequential strategies. The best performance is achieved by our planner (UE), with a 14% improvement compared to when occlusions are not accounted for (E). The example shows that selecting viewpoints which maximise surface area (A) tends to have worse performance than the other utility functions. All active planners which consider all objects simultaneously outperform the method that selects locations nearest to the lowest entropy viewpoint for the most uncertain object (NE).

This procedure was repeated for the other scenarios and results are presented in Tab. I. In 15 scenarios, minimising entropy (UE,E) or maximising class probability (UP,P) required the fewest views. In 10 of the scenarios, our utility function (UE) performed best. In the remaining scenarios, the best performing methods were maximising surface area (A) (2 cases), selecting viewpoints closest to the highest utility location for the most uncertain object (NE) (2 cases), and sequential views (1 case). Random viewpoint selection never achieved the best performance.

The last row in Tab. I shows mean results. Our method has the lowest mean and smallest standard deviation. In comparison to the same utility function, which uses the class probability (UP), the performance is on average 7% better.

(a) (b)

Fig. 5: Setup 1: (a) side view and (b) top view.

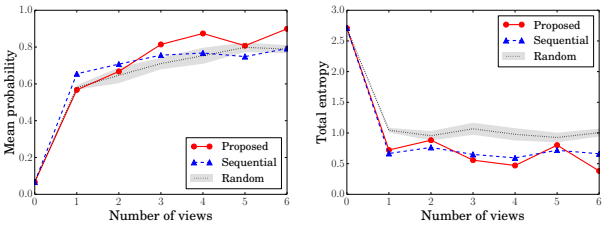


(a) (b)

Fig. 6: Comparison of active planner with random and sequential for setup 1: (a) mean of the true class probability for each object and (b) total entropy of the object class distributions.

(a) (b)

Fig. 7: Setup 2: (a) side view and (b) top view.



(a) (b)

Fig. 8: Comparison of active planner with random and sequential for setup 2: (a) mean of the true class probability for each object and (b) total entropy of the object class distributions.

Both these strategies have better performance when accounting for occlusions, for the case of views with minimum entropy the improvement is 13% while for the case of views with maximum probability the improvement is 8%.

There is little difference between the two utility functions which do not account for occlusions (E,P), and the same performance is achieved by maximising surface area (A). This shows that similar performance is achieved using either objective separately, but using them jointly is beneficial.

The worst performing active planner is (NE). This indicates that it is advantageous to consider all objects simultaneously because single observations can provide information about multiple objects and their estimates can improve together, which may lead to greater improvement overall.

As expected, all active planners outperform random and sequential. For comparison, our method recognises objects with nearly 2 and 3 fewer views, an improvement of 26% and 36% respectively. Random outperforms sequential by 1 fewer view and has a smaller variance, indicating that random is an adequate comparison for evaluating active strategies.

B. Hardware Experiments

1) Experimental Platform: Our robot is a Festo Robotino with custom-mounted ASUS XTion Pro Live RGB-D sensor, shown earlier in Fig. 1. Software is written in C++ using the point cloud library [24], octomap [25] and ROS [26].

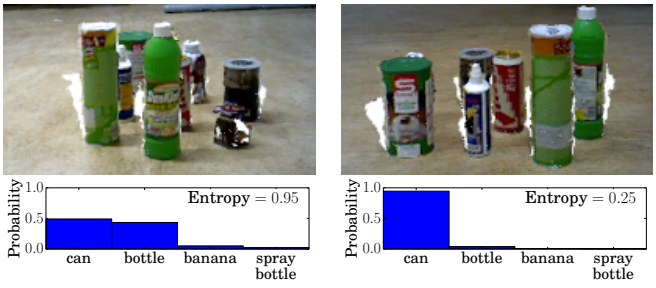
2) Experimental Setup: The classifier was trained with the same database of objects and parameters as the previous experiments. The localisation error was larger, so the voxel overlap threshold for data association was increased to 40%.

There are two setups, shown in Figs. 5 and 7, that consist of objects on the floor belonging to the classes *can*, *bottle* and *box* in a variety of shapes and sizes. The initial set of viewpoints was pre-selected as 12 evenly spaced locations on a circle around the objects with radius 2m. A simple roadmap connects each viewpoint to its neighbours for navigation.

Two hardware experiments (one for each setup) were performed with our method and terminated after 6 observations. These observations were sufficient because the class confidence did not increase significantly after this point. For comparison, we performed offline simulations where viewpoints were selected at random or sequentially (clockwise) around a half circle. Simulations used the data collected during online experiments combined with pre-collected observations from the remaining viewpoints. For each setup, 10 simulations were run in the random case and 1 in the sequential case. The initial viewpoint for both cases was set to coincide with the initial viewpoints of the online experiments, and likewise, 6 observations were made in total.

A comparison of the performance between active and passive planning was done using two metrics. The first was *mean probability*, $\bar{p_k} = \frac{1}{N_k} \sum_{n \in \mathcal{N}_k} p_k^n(x_{\mathrm{GT}})$, which is the average probability of the ground truth class x_{GT} of each object. This metric evaluates the accuracy of the classifier at the level of individual objects. The second was *total entropy*, which is the sum of the entropies of each object's class distribution. This metric evaluates the uncertainty of the classifier's estimate.

3) Results Discussion: Figures 6a and 8a indicate that the confidence of the true object identities improves as more observations are made. Similarly in Figs. 6b and 8b, the total entropy of the class probability distributions tends to decrease. For the active planner in the second setup, the total entropy shown in Fig. 8b appears to have large fluctuations. On inspection of the data, the fluctuations are explained by: 1) occlusions causing some objects to be unobserved in the beginning but then observed after the second view, resulting in an increase of the total entropy due to another object in the summation (first rise) and 2) merging object fragments into single hypotheses which momentarily increases the new entropies because of the combination of multiple and possibly different distributions (second rise). However, after all 6 observations were selected the final entropy value reaches a very low level.



(a) (b)

Fig. 9: Selected viewpoint in the online experiment of setup 2, showing the RGB-D observations, the probability values for the 4 top scoring classes and the entropy of the total distribution for the *Stiegel can* (red and white) located in the centre: (a) viewpoint 2 and (b) viewpoint 3.

In the first setup, we can see that after 3 observations the object confidence for the active planner reaches a plateau. This level is only reached by the random and sequential planners after all 6 views are observed. In the second setup, the sequential and random planners never reach the level of certainty of the active planner.

For illustrative purposes, we provide two example viewpoints and associated probability distributions for a selected object (a can) in Fig. 9. In the first viewpoint, the can is occluded and the resulting class estimate is uncertain. However, in the second viewpoint the can is unoccluded and the class estimate improves. This improvement can also be seen in Fig. 8 (views 2 and 3).

For these experiments the computation time of the planning module was measured as this is the main bottleneck of the overall system. On average, evaluating each candidate location took 7.7 ± 2.1s with our basic implementation (running on a laptop with Intel Core i5 2.7Hz and 4GB RAM) and ray tracing was the dominant operation. Fortunately, the planning stage is easily parallelisable and with an optimised implementation we believe that the system can run considerably faster.

IX. Conclusion

We have presented an end-to-end system for active object classification with RGB-D data that plans future observations to identify uncertain objects in clutter. We proposed a utility function that exploits offline classification data, combined with online occlusion reasoning, to predict the value of a future observation. Our results with a large dataset, validated with a mobile robot, show that actively selecting viewpoints usefully outperforms passively accepting data in reducing the number of views needed to confidently classify objects.

Our hope is that our framework will be of benefit in improving systems that currently rely on passive perception. However, there are several important areas of future work. One is to consider more sophisticated motion planning, accounting for continuous viewpoint locations, travel costs, and long-horizon planning that employs our framework for viewpoint prediction. We would also like to jointly and probabilistically estimate the occupied space, class and pose of objects, and to devise a planning method to improve these state estimates.

Acknowledgements

Thanks to Edith Langer for her help in operating the robot.

References

[1] S. Gan, R. Fitch, and S. Sukkarieh, "Online decentralized information gathering with spatialtemporal constraints," *Auton. Robots*, vol. 37, no. 1, pp. 1–25, 2014.
[2] Z. Xu, R. Fitch, J. Underwood, and S. Sukkarieh, "Decentralized coordinated tracking with mixed discretecontinuous decisions," *J. Field Robot.*, vol. 30, no. 5, pp. 717–740, 2013.
[3] [Online]. Available: http://www.squirrel-project.eu/
[4] [Online]. Available: http://vault.willowgarage.com/wgdata1/vol1/solutions_in_perception/Willow_Final_Test_Set/
[5] K. Wu, R. Ranasinghe, and G. Dissanayake, "Active recognition and pose estimation of household objects in clutter," in *Proc. of IEEE ICRA*, 2015.
[6] S. Chen, Y. Li, and N. M. Kwok, "Active vision in robotic systems: A survey of recent developments," *Int. J. Rob. Res.*, vol. 30, no. 11, pp. 1343–1377, 2011.
[7] M. F. Huber, T. Dencker, M. Roschani, and J. Beyerer, "Bayesian active object recognition via gaussian process regression," in *Proc. of FUSION*, 2012.
[8] J. Denzler and C. M. Brown, "Information theoretic sensor data selection for active object recognition and state estimation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 2, pp. 145–157, 2002.
[9] D. Meger, A. Gupta, and J. Little, "Viewpoint detection models for sequential embodied object category recognition," in *Proc. of IEEE ICRA*, 2010, pp. 5055–5061.
[10] C. Potthast, A. Breitenmosero, F. Sha, and G. Sukhatme, "Active multi-view object recognition and online feature selection," in *Proc. of ISRR*, 2015.
[11] I. Becerra, L. Valentin-Coronado, R. Murrieta-Cid, and J.-C. Latombe, "Appearance-based motion strategies for object detection," in *Proc. of IEEE ICRA*, 2014.
[12] B. Browatzki, V. Tikhanoff, G. Metta, H. Bulthoff, and C. Wallraven, "Active object recognition on a humanoid robot," in *Proc. of IEEE ICRA*, 2012.
[13] R. Eidenberger and J. Scharinger, "Active perception and scene modeling by planning with probabilistic 6d object poses," in *Proc. of IEEE/RSJ IROS*, 2010.
[14] N. Atanasov, B. Sankaran, J. Le Ny, G. J. Pappas, and K. Daniilidis, "Nonmyopic view planning for active object classification and pose estimation," *IEEE Trans. Robot.*, vol. 30, no. 5, pp. 1078–1090, 2014.
[15] A. Aldoma, F. Tombari, J. Prankl, A. Richtsfeld, L. Di Stefano, and M. Vincze, "Multimodal cue integration through hypotheses verification for RGB-D object recognition and 6DOF pose estimation," in *Proc. of IEEE ICRA*, 2013.
[16] T. Faulhammer, A. Aldoma, M. Zillich, and M. Vincze, "Temporal integration of feature correspondences for enhanced recognition in cluttered and dynamic environments," in *Proc. of IEEE ICRA*, 2015.
[17] Z. Xie, A. Singh, J. Uang, K. S. Narayan, and P. Abbeel, "Multimodal blending for high-accuracy instance recognition," in *Proc. of IEEE/RSJ IROS*, 2013.
[18] V. Karasev, A. Chiuso, and S. Soatto, "Controlled recognition bounds for visual learning and exploration," in *Proc. of Adv. Neural Inf. Process. Syst.*, 2012.
[19] L. L. S. Wong, L. P. Kaelbling, and T. Lozano-Prez, "Data association for semantic world modeling from partial views," *Int. J. Rob. Res.*, vol. 34, no. 7, pp. 1064–1082, 2015.
[20] P. Besl and N. D. McKay, "A method for registration of 3-D shapes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 14, no. 2, pp. 239–256, 1992.
[21] A. Richtsfeld, T. Morwald, J. Prankl, M. Zillich, and M. Vincze, "Segmentation of unknown objects in indoor environments," in *Proc. of IEEE/RSJ IROS*, 2012, pp. 4791–4796.
[22] W. Wohlkinger, A. Aldoma, R. Rusu, and M. Vincze, "3DNet: Large-scale object class recognition from CAD models," in *Proc. of IEEE ICRA*, 2012.
[23] W. Wohlkinger and M. Vincze, "Ensemble of shape functions for 3d object classification," in *Proc. of ROBIO*, 2011.
[24] R. Rusu and S. Cousins, "3D is here: Point cloud library (PCL)," in *Proc. of IEEE ICRA*, 2011.
[25] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Auton. Robots*, vol. 34, no. 3, pp. 189–206, 2013.
[26] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source robot operating system," in *Proc. of IEEE ICRA, Workshop Open Source Software*, 2009.