Robotics and Autonomous Systems 84 (2016) 31–47

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot

Active multi-view object recognition: A unifying view on online feature selection and view planning

Christian Potthast *, Andreas Breitenmoser, Fei Sha, Gaurav S. Sukhatme

University of Southern California, Department of Computer Science, Los Angeles, CA 90089, USA

HIGHLIGHTS

- A unified approach to feature and viewpoint selection for multi-view object recognition is proposed.
- Online feature selection reduces the dimensionality and with that the computation time.
- View planning offers performance advantages whenever multiple views are required due to ambiguous situations or occlusions.
- Increased recognition accuracy and reduced computation cost are realized by an information-theoretic action selection framework.

ARTICLE INFO

Article history:
Received 26 May 2015
Received in revised form
14 January 2016
Accepted 27 June 2016
Available online 4 July 2016

Keywords:
Viewpoint selection
Feature selection
Multi-view object recognition
Information-theory
Active perception
Object change detection

ABSTRACT

Many robots are limited in their operating capabilities, both computational and energy-wise. A strong desire exists to keep computation cost and energy consumption to a minimum when executing tasks like object recognition with a mobile robot. Adaptive action selection is a paradigm, offering great flexibility in trading off the cost of acquiring information against making robust and reliable inference under uncertainty. In this paper, we study active multi-view object recognition and describe an information-theoretic framework that combines and unifies two common techniques: online feature selection for reducing computational costs and view planning for resolving ambiguities and occlusions. Our algorithm adaptively chooses between the two strategies of either selecting only the features that are most informative to the recognition, or moving to a new viewpoint that optimally reduces the expected uncertainty on the identity of the object. This two step process allows us to keep overall computation cost minimal but simultaneously increase recognition accuracy. Extensive empirical studies on a large RGB-D dataset, and with two different feature sets, have validated the effectiveness of the proposed framework. Our experiments show that dynamic feature selection alone reduces the computation time at runtime 2.5–6 times and, when combining it with viewpoint selection, we significantly increase the recognition accuracy on average by 8%–18% absolute compared to systems that do not use these two strategies. By establishing a link between active object recognition and change detection, we were further able to use our framework for the follow-up task of actively detecting object change. Furthermore, we have successfully demonstrated the framework's applicability to a low-powered quadcopter platform with limited operating time.

1. Introduction

Despite decades of research, object recognition from visual percepts is still an unsolved and challenging problem. In robotics especially, where many tasks build on, or directly involve, the recognition of objects, the need has grown for creating reliable and fast classification systems. Conventional approaches for object recognition are tour de force, exclusively relying in large part on complex statistical models for classification and heavy engineering of computationally-intensive features [1,2]. All too often, systems are assumed to be unlimited and their processing capabilities are neglected. This stands in stark contrast to many of today's mobile robots, which often have limited power and need to be economical with resources for computing and reasoning—being able to move and execute given tasks is their first priority!

Active object recognition [3,4] is an appealing paradigm to overcome the challenges in recognition of objects. Here, a mobile robot configures and positions its sensors to inspect a target object

* Corresponding author.
E-mail addresses: potthast@usc.edu (C. Potthast), andreas.breitenmoser@usc.edu (A. Breitenmoser), feisha@usc.edu (F. Sha), gaurav@usc.edu (G.S. Sukhatme).

http://dx.doi.org/10.1016/j.robot.2016.06.013
0921-8890/

and recognize it as a previously learnt object. The advantage of active object recognition lies in the ability of reacting to unforeseen objects or scenes by exploratory actions. This allows to seek actively for the information that is the most useful in performing inference under uncertainty.

Adaptive action selection offers the flexibility that is required to potentially re-use the statistical modeling power of conventional approaches even under the severe (computing) constraints present on mobile robots. The general idea is to control the sensors such that only data with high information content is collected. By running inference recursively based on previously acquired information (which represents the robot's current knowledge, i.e., its belief state about the identity of the object), the collection of data can be *adapted dynamically*. Therefore, computation time can potentially be reduced, because we only process useful information and have the ability to improve our belief with each additional action.

In previous work, there are two common actions that have been explored in order to increase the recognition accuracy: first, feature selection [5], which computes a subset of informative features; second, viewpoint selection or view planning [6,3], which controls where and how many additional observations to take. However, previous approaches have not looked at the potential of reducing computation time due to feature selection. Moreover, existing work does not exploit the full potential of combining these two strategies and applying feature and viewpoint selection in concert.

The main idea of our work hinges on the following intuition: *Is it possible to reduce overall costs for recognition by relying on feature subsets gathered from multiple views*, in lieu of the full feature set, potentially taken from a single view, using a complex statistical model for classification? More concretely, in this paper, we present an information-theoretic multi-view object recognition framework that unifies the two strategies of *feature selection* and *viewpoint selection*. We aim at exploiting local information by selecting additional features at the current viewpoint and further exploring the object by selecting new viewpoints around it. The described framework optimizes the viewpoint selection by reducing ambiguities and selecting features dynamically, such that only useful and informative features are extracted and incorporated in the recognition process. Unlike standard feature selection, the two strategies are symbiotic in our algorithm. Feature selection is an iterative process, thus at any given time, the selected feature depends on the trajectory of the past viewpoints while simultaneously affecting the future viewpoint(s).

We perform extensive evaluations on a large RGB-D camera dataset, which is composed of single object scenes. To evaluate our framework, we conduct experiments with two different feature sets, one set consisting of *simple features* designed by us, and one set of more complex *kernel descriptors* proposed by [1]. The use of the simple features shows that, in combination with feature selection and view planning, our framework can compete with state-of-the-art single-view classification systems. In contrast, using the kernel descriptors shows that our framework is capable of dealing with more advanced features as well.

We further show the advantages of combining intelligent view planning and online feature selection. Online feature selection reduces the computation time fivefold for the simple features and 2.5-fold for the kernel features at runtime. Furthermore, in combination with view planning, we are able to increase the recognition accuracy significantly by resolving object ambiguities. In the case of the simple features, we achieve an increase in accuracies by 8%–15%, while in case of the kernel descriptors the increase is about 8%. Overall, our results show pose estimates within $\pm 45^\circ$ of the ground truth and object recognition accuracies of over 90% when using the simple features in rather challenging object recognition experiments. Using the kernel descriptors, we improve the object recognition accuracy to over 98%. As proof-of-concept, we eventually implemented and tested the proposed approach on a custom-built quadcopter robot. The quadcopter is equipped with an RGB-D camera and collects multiple views by flying around a target object, which allows to recognize the object.

Additionally, we show how our active multi-view recognition system can be used for *object change detection*. Given a prior map of the environment, the task is to determine whether objects in the environment have changed. This can either mean that the object has changed completely (i.e., it belongs to a different object class), or the object has changed its orientation only (i.e., it belongs to the same class but its rotation differs in yaw). Our results show that on average our framework can detect such changes in the environment with an accuracy of larger than 90%.

The remainder of the article is organized as follows. Section 2 presents related work on active object recognition and pose estimation, including relevant approaches in feature selection and viewpoint planning. The problem formulation and proposed solution of our active multi-view object recognition approach is detailed in Section 3. Section 4 introduces the unified selection of actions, which is applied to both feature selection in Section 5 and viewpoint selection in Section 6. The object recognition experiments and results are discussed in Section 7, and a conclusion is provided in Section 8.

2. Related work

Prior works on active perception go back as far as the seminal paper [7]. "Active" can either refer to adjustments of the sensor parameters themselves, e.g., adaptively changing the focal length of a camera, as in [7,4], or mean that the entire sensor moves together with the underlying platform. The latter is particularly common in robotics, where robots plan new observation locations and viewing directions actively [8,9,3]. This is known as robotic view planning. [6] presents a comparison of several representative approaches to view planning from literature. View planning can be formulated as a purely geometric problem [10,11] but as well relates to robotic exploration, where information-theoretic strategies have become state-of-the-art solutions [12,13]. The selection of optimal viewpoints for a robot can be useful in various applications, such as for the inspection of underwater structures [14], for object search in indoor environments [15,16] or detecting objects on a table top [3]. Moreover, methods for view planning and active object recognition provide the basis for operations like object sorting [17] and manipulation [13,18], acquisition of object representations [19] or change detection (we will show in the end of Section 7 how our object recognition framework naturally extends to perform object change detection).

The common goal of object recognition approaches is to recognize objects and estimate their poses [4,8,15,20,21]. [4] presents an active object recognition framework that, similar to ours, relies on a sequential decision process and mutual information as information-theoretic measure for the selection of actions. [8] selects multiple viewpoints by using the Jeffrey's divergence instead of maximizing the mutual information. [15] combines active object recognition with Simultaneous Localization and Mapping (SLAM) techniques to estimate object positions and link semantic with spatial information. [20] extends a single-view to a multi-view object recognition method for increased efficiency by applying multi-step optimization and extracting SIFT features from camera images. [21] performs object recognition and object detection by using different classifiers and several shape and visual features.

To cope with objects that appear ambiguous in single views, [22–24,20] propose multi-view object recognition frameworks

that use cameras and image databases, whereas [3,16,25,21] recognize objects by extracting features from laser scanners and RGB-D depth cameras. [25] applies the online boosting approach from [26] to point clouds in the robotic domain for feature selection. Feature selection can support the object recognition task by reducing the computation time and confusions through suboptimal feature sets [27,5]. A standard alternative to boosting is the selection of features by mutual information [28,5]. [28] tries to select the most relevant features by maximizing the dependency of a feature on the target object (i.e., maximizing the mutual information), while minimizing the redundancy between the features. [29] evaluates the feature selection methods from [28] when applied to the RGB-D dataset of [21]. Ensemble methods [30] offer further alternatives of how to combine features; [31] employs ensemble methods for 3D face recognition and [32] compares the concatenation of all available features to ensembles with stacking. [33] proposes a unifying view on active recognition and feature selection, however, focuses on feature interdependencies and the selection of features given a concrete context, e.g., a specific target object to detect. Their method, as well as the other feature selection methods for object recognition mentioned above, do not explicitly consider viewpoint selection, whereas our object recognition framework applies both feature and viewpoint selection in concert.

Although many approaches for active view planning as well as online feature selection exist, to the best of our knowledge, the advantages of both approaches have never been leveraged in combination. This paper builds on a prior conference paper [34]. The journal version provides a more detailed problem formulation, demonstrates broader applicability to more complex feature sets and object change detection, and includes more extensive evaluation of the unified feature and viewpoint selection framework.

3. Problem formulation

We look at solving the following problem. Given a static scene and an object in question, we want to recognize the object and, at the same time, estimate its pose relative to the coordinate system of the observing sensor, while keeping the computation cost to a minimum. The recognition process matches an observation of the target object with the objects stored in a database and computes the belief states of all the objects. Due to noise, ambiguity and occlusions the recognition process does not always lead to certain matchings, which in turn results in multiple likely object candidates. To this end, our process seeks to select a minimal sequence of actions which reduces the computation time but simultaneously maximizes the acquired information about the target object. In other words, the actions are chosen in such a way as to reduce the uncertainty in the current belief state estimates for the object candidates.

3.1. System overview

The general system diagram of our active recognition framework is presented in Fig. 1. It consists of the four modules: *object observation*, *feature selection*, *recognition state estimation*, and *viewpoint selection*.

In each time step t, we select a next action from the set of feasible actions, which are: (1) stopping the recognition process, (2) staying at the current viewpoint and computing a next feature f_{t+1} (feature selection), or (3) moving the sensor to a new viewpoint by applying the motion action a_t (viewpoint selection). Concretely, the action a_t defines the relative motion of the sensor to move from the current position to the new observation position. We only allow sensor motions to a restricted set of next observation positions. The discretization of the viewpoints reduces the complexity in computation when compared to a continuous representation of viewpoints. When computing a new feature f_{t+1} at the current viewpoint, we set $a_t = \emptyset$, defining a *zero motion action*.

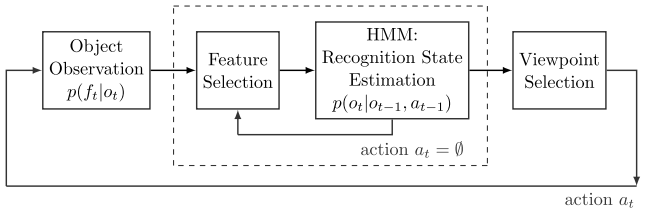

Fig. 1. System overview of the active multi-view object recognition framework. After each observation, the HMM updates the recognition state, upon which a new feature (inner loop) or a new viewpoint (outer loop) is selected. The feature type that is selected first at any new location is pre-defined. The iterative processes terminate when the stopping criteria are met.

During object observation, the sensor captures the raw data, such as color images and 3D point clouds, which is used to compute the features. The sensor measurements at a new viewpoint are parametrized by the feature vector $\boldsymbol{f} = [f^1, \ldots, f^N]$, which includes N different feature types. Given that the nth feature in $\boldsymbol{f}$ has been selected at time t, we extract feature $f_t = f^n$ from the measurement data. The observation uncertainty is captured in our observation model, defined as the conditional probability density function $p(f_t|o_t)$, with recognition state $o_t = \{c_t, \theta_t\}$. Here, the feature f_t is conditioned on the object class c_t and orientation angle θ_t of the observed object. In the recognition state estimation, we use a *Bayesian framework* to infer about the current c_t and θ_t from the measurements. The parameters are assumed to be obtained from a physical object model and to be trained offline.

At each observation position, we start off with extracting one first feature from our total feature set specified by vector $\boldsymbol{f}$. We then have the choice of local exploitation, i.e., extracting another feature, or global exploration, i.e., moving the sensor to a new observation position. The premise is to extract as much information as possible at the current viewpoint since a motion action is inherently more costly. Only after having exploited all the features that yield useful information, we want to move to a new location. However, as moving the sensor physically is costly, finally, we move to the next viewpoint only if the gain of information (in expectation) significantly reduces the uncertainty in the object belief states. Put in other words, a next motion action is applied if and only if the recognition accuracy is expected to improve by a significant amount.

Before we extract and add a new feature, we compute the utility of all remaining features in our arsenal and select the one with the highest utility. We only allow to add one feature at a time, and once a feature is added it cannot be removed from the set anymore. This way, integrating a new feature simply becomes a sequential update step without the need for re-computing the posterior of the entire feature set. We continue until either a stopping criterion has been reached or all N features are observed at a given viewpoint, forcing us to either terminate or move to a new viewpoint. The overall idea is that, at each viewpoint, we have only extracted useful features and left out features that yield no additional information. Moreover, not extracting a feature saves time, which helps to reduce the total time needed to recognize the object.

3.2. Sequential Bayesian recognition

With every new observation of a feature f_{t+1}, we want to efficiently integrate the new observation with results from prior observations, and thus update our current belief about the object class and orientation. We formulate the recognition and pose estimation over the object classes and object poses as a Bayesian

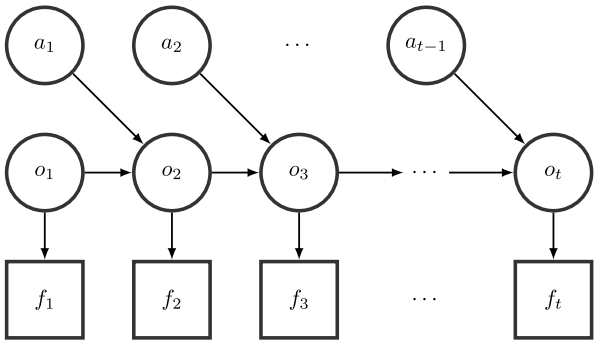

Fig. 2. HMM for sequential Bayesian recognition. Given a sequence of observed features $f_{1:t}$ and executed actions $a_{1:t-1}$, we reason about the hidden recognition state of an object at time t, $o_t = \{c_t, \theta_t\}$, including its object class $c_t = c^k$ and object pose $\theta_t = \theta^q$.

network in form of a *Hidden Markov Model* (HMM). A Bayesian formulation has distinct advantages, with the main advantage being that Bayesian methods model all sources of uncertainty in form of random variables. Furthermore, HMMs have efficient formulations to sequentially update the posterior distribution, making it ideal for integrating new observations, such as in our case in the form of a new feature f_{t+1}. Compared to a naive Bayes update, which is often used in existing work [4,23,8,22], the HMM formulation additionally gives us the possibility to explicitly formulate the *state transitions*.

The joint probability distribution of the HMM, as shown in Fig. 2, is defined as

$$\begin{aligned} p(o_{1:T}, f_{1:T}, a_{1:T-1}) &= p(o_{1:T})\, p(a_{1:T-1}|o_{1:T})\, p(f_{1:T}|o_{1:T}, a_{1:T-1}) \\ &= \left[p(o_1) \prod_{t=2}^{T} p(o_t|o_{t-1}, a_{t-1})\right] \\ &\quad \times \left[\prod_{t=1}^{T} p(f_t|o_t)\right], \end{aligned} \tag{1}$$

and the corresponding log probability distribution, $\log(p(o_{1:T}, f_{1:T}, a_{1:T-1}))$, equals

$$\log(p(o_1)) + \sum_{t=2}^{T} \log(p(o_t|o_{t-1}, a_{t-1})) + \sum_{t=1}^{T} \log(p(f_t|o_t)). \tag{2}$$

The observation sequence is $S = \{o_1, \ldots, o_t, a_1, \ldots, a_{t-1}\}$, the hidden state of our HMM is defined as recognition state o_t with $o_{1:T} = \{o_1, \ldots, o_t, \ldots, o_T\}$, the observations are given as observed features f_t with $f_{1:T} = \{f_1, \ldots, f_t, \ldots, f_T\}$ and the motion actions between viewpoints are given as a_t with $a_{1:T-1} = \{a_1, \ldots, a_t, \ldots, a_{T-1}\}$. The state of the hidden process satisfies the Markov property, that is, given the value of o_{t-1}, the current state o_t is independent of all the states prior to $t-1$.

The *observation model* $p(f_t|o_t)$ represents the likelihood that feature f_t is generated by recognition state o_t. The transition probability is defined as $p(o_t|o_{t-1}, a_{t-1})$, which means that the transition to the next recognition state is dependent on the previous recognition state (i.e., the previous object class and orientation) as well as the latest motion action that has been applied. The posterior marginals $p(o_t|f_{1:T}, a_{1:T-1})$ of all hidden state variables, given a sequence of observations, can be efficiently computed using the *forward–backward algorithm*. The algorithm computes a smoothed estimate given all evidence in two passes, with the first pass going forward in time and the second pass going backwards in time:

$$\begin{aligned} p(o_t|f_{1:T}, a_{1:T-1}) &= p(o_t|f_{1:t}, f_{t+1:T}, a_{1:t-1}, a_{t:T-1}) \\ &\propto p(f_{t+1:T}, a_{t:T-1}|o_t)\, p(o_t|f_{1:t}, a_{1:t-1}). \end{aligned} \tag{3}$$

In the first pass, we compute the forward probabilities, which are the probabilities for all $t \in \{1, \ldots, T\}$ ending up in any particular state given the first t observations in the sequence $p(o_t|f_{1:t}, a_{1:t-1})$. In the second step, we compute the probabilities $p(f_{t+1:T}, a_{t:T-1}|o_t)$ of making the remaining observations given any starting point t.

The recursion of the forward–backward algorithm can be broken up into a forward and backward part. The forward recursion is defined as:

$$\begin{aligned} \alpha(o_T) &\equiv p(f_{1:T}, o_{1:T}, a_{1:T-1}), \quad \text{and} \\ \alpha(o_T) &= p(f_T|o_T) \sum_{t=2}^{T} \alpha(o_{t-1})\, p(o_t|o_{t-1}, a_{t-1}). \end{aligned} \tag{4}$$

$\alpha(o_T)$ is the joint probability of observing $f_{1:T}$ and being in state o_T. We can similarly formulate the recursive backward part:

$$\begin{aligned} \beta(o_T) &\equiv p(f_{T+1:1}, o_{T+1:1}, a_{T:1}), \quad \text{and} \\ \beta(o_T) &= \sum_{t=1}^{T} \beta(o_{t+1}) p(f_{t+1}|o_{t+1})\, p(o_{t+1}|o_t, a_t). \end{aligned} \tag{5}$$

$\beta(o_T)$ is the conditional probability of future observations $f_{T+1:1}$, under the assumption of being in state o_T. The posterior probability can then be obtained from

$$p(o_t|f_{1:T}, a_{1:T-1}) = \sum_{t=1}^{T} \alpha(o_t)\, \beta(o_t). \tag{6}$$

Before we can make use of the forward–backward algorithm in practice, there is another important issue to be addressed. In the recursion of the forward and backward parts, we note that at each step the new value is obtained from the previous value by multiplication. These probabilities are often far less than unity and the values can go to zero exponentially fast. There are two approaches to deal with this numerical problem. One option is to use re-scaled versions of $\alpha(o_t)$ and $\beta(o_t)$ whose values remain close to unity. However, since our observation probability is in log-space, $\log(p(f_t|o_t))$, this is impractical and oftentimes not even possible. The second option is to work in log-space as well and evaluate the likelihood functions by taking the logarithms. However, because of "$\log(\sum(x)) \neq \sum(\log(x))$", we cannot easily do that. Instead, we must use the "log-sum-exp" trick, which eventually leads to the recursive formulation of $\alpha(o_t)$ and $\beta(o_t)$ in log-space:

$$\begin{aligned} \log(\alpha(o_T)) &= \log(p(f_T|o_T)) + \log \sum_{t=2}^{T} \exp(\log(\alpha(o_{t-1})) \\ &\quad + \log(p(o_t|o_{t-1}, a_{t-1}))), \\ \log(\beta(o_T)) &= \log \sum_{t=1}^{T} \exp\big[\log(\beta(o_{t+1})) \\ &\quad + \log(p(f_{t+1}|o_{t+1})) + \log(p(o_{t+1}|o_t, a_t))\big]. \end{aligned} \tag{7}$$

The posterior probability is then given in log-space as

$$\log(p(o_t|f_{1:T}, a_{1:T-1})) = \sum_{t=1}^{T} \log(\alpha(o_t)) + \log(\beta(o_t)). \tag{8}$$

3.3. Observation model

For recognition, we learn a function $g : \mathbb{R}^{|\boldsymbol{f}|} \to \mathbb{N}$ that maps the extracted object features $\boldsymbol{f}$ to a unique object identifier $i \in \{1, \ldots, K\,Q\}$, each pointing to a distinct object model $o^i = \{c^k, \theta^q\}$, with $k \in \{1, \ldots, K\}$ and $q \in \{1, \ldots, Q\}$. K denotes the total number of object classes c^k and Q defines the number of discretized object poses θ^q per object class. The function is learned

in a supervised fashion, with all possible features $\boldsymbol{f} = [f^1, \ldots, f^N]$ and the unique identifier i available as the value pair $(\boldsymbol{f}, i)$ at model training phase. While we assume that all features of an object are observed at training phase, we do not make this assumption during prediction, where not all components of $\boldsymbol{f}$ are observed due to feature selection.

To deal with the missing data problem at prediction time, we use a Gaussian generative model of the form

$$\log(P(f|o^i)) = \sum_{m=1}^{M} \log(\mathcal{N}(f|\mu_{o^i,m}, \sigma_{o^i,m})). \tag{9}$$

M is the number of independent normal distributions to model a feature f. Each feature f is represented as a feature histogram with M bins, and the mean $\mu_{o^i,m}$ and variance $\sigma_{o^i,m}$ for each bin m are computed over the training data of an object with recognition state o^i. We use M independent normal distributions since a multivariate normal distribution, with full covariance matrix, would become high-dimensional for high-dimensional features. High-dimensional models are in general poorly separable as is well understood under the *curse of dimensionality*. Defining the problem as an independent Gaussian generative classifier has the advantage of handling missing features by marginalizing them out at prediction time, i.e., instead of inferring about the missing features, we simply perform inference on a subset of the full feature vector $\boldsymbol{f}$. We can now compute the log-likelihood $\log(p(\boldsymbol{f}|o^i))$ of generating the model o^i from feature vector $\boldsymbol{f}$ as

$$\log(P(\boldsymbol{f}|o^i, \psi)) = \sum_{n=1}^{N} \psi^n \sum_{m=1}^{M} \log(\mathcal{N}(f^n|\mu_{o^i,m}, \sigma_{o^i,m})), \tag{10}$$

with ψ^n denoting a scaling factor to scale the different feature distributions. The feature distribution in (10) allows us to integrate out the marginals very easily. In practice, marginalization of a feature is simply done by setting the scaling factor to zero for all the features we do not want to include in the inference. With that, the log-likelihood $\log(P(\boldsymbol{f}|o^i, \psi))$ is only computed over a subset of all possible features, making it independent of the dimensionality of $\boldsymbol{f}$ during training. Furthermore, the Gaussian representation of the features is compact and fast to train, since it only requires the computation of the Gaussian feature parameters $\mu_{o^i,m}$ and $\sigma_{o^i,m}$. Additionally, sequential learning as well as sequential updates of mean and variance are possible, which has the potential of adaptively adding new observations to our trained features.

Learning multiple models per object class has two distinct advantages. For one, inferring about an observation taken from a target object will not only yield the most probable object class, but also the most probable pose of the object. Second, the model trained for each object has a smaller variance if trained only for a partial view of the object. This becomes intuitively clear since an object looks in general very different for different viewing directions. And this again is reflected in the variance of the model.

3.4. HMM state transitions

We define the state transition probability $p(o_t|o_{t-1}, a_{t-1})$ as the probability of a transition from recognition state o_{t-1} to state o_t given an action a_{t-1} (see Fig. 2). This means, how likely is the transition, if at time $t-1$ we believe to be in recognition state $o_{t-1} = \{c^k, \theta^q\}$ (i.e., a believed target object with object class c^k and orientation θ^q), then perform the motion action a_{t-1} and acquire a new observation and feature f_t, and finally believe to be in state $o_t = \{c^k, \theta^q\}$ at time t.

In order to determine the state transition probabilities, one uses the *Baum–Welch algorithm*, which uses expectation maximization to find the maximum likelihood estimate of the state transition parameters given a set of observed data. Commonly, training of such parameters is done in batch, meaning that all data needs to be available upfront. This, however, prohibits a sequential update with new data. Sequential update methods have been developed, but come with their own challenges, such as avoiding to get stuck in local minima, finding stopping criteria to reduce overfitting, or not corrupting existing knowledge. We will leave the learning of the transition probabilities for future work and use a custom-defined state transition matrix for now.

4. Action selection

When studying object recognition, uncertainty in the state estimation generally leads to an increase in variance, whereas ambiguity in feature space increases the number of modes of the belief state. Consequently, to improve the recognition accuracy, we want both to reduce the variance and to reduce the number of modes toward a unimodal belief state. The root cause for multimodal belief states is the closeness of sensor data and the training models in feature space. As a direct result, the sensor data is likely to match with multiple object hypotheses, and hence a definitive object class cannot be found. In other words, certain viewing choices or combinations of features may be well explained by more than just one hypothesis.

In order to reduce uncertainty and ambiguity, we need to select actions and reason about new information in the form of newly computed features. However, not every feature or observation position, respectively, reduces the uncertainty in belief state to the same extent (and some even result in no reduction at all). Intuitively, we only want to incorporate new features that provide new information and move to new viewpoints that reduce the ambiguity by offering views of objects that are inherently different from views of objects from other object classes.

In summary, we aim at finding the sequence of actions which maximally reduces the uncertainty about our current object belief state. The action sequence should be minimal, depending on the belief state, since every action involves a cost. Finding the sequence of actions that will reduce the uncertainty is nontrivial because we need to quantify all feasible actions. Let $w \in \Omega$ be an action and Ω be the set of all feasible actions. To quantify each action w, we use the utility function $\mathcal{U} : \Omega \to \mathbb{R}$ to define the *utility score* of action w at time t as $s_t(w) = \mathcal{U}(w)$. The best next action w^*, which reduces the uncertainty of the belief state the most, is then computed as the action with the maximum utility score

$$w^* = \underset{w \in \Omega}{\operatorname{argmax}}\ s_t(w)\,. \tag{11}$$

Compared to global viewpoint exploration, the local exploitation of a viewpoint through local feature exploration has much lower associated cost. For that reason, we do not want to optimize over the entire action set consisting of extractable features and motion actions, given by the tuple (f, a), at once. Instead, we sequentially optimize—first over Ω_f, the discrete set of actions that are expressed as the features available in $\boldsymbol{f}$, followed by optimizing over the discrete set of all feasible motion actions Ω_a.

Regarding feature selection, the action of staying at the same viewpoint and computing a next feature f_{t+1} is obtained from (11) with $w \equiv (f, a)$ as $f_{t+1} = f^* = \operatorname{argmax}_{f \in \Omega_f} s_t(f, a)$. Here, the motion action is fixed, and in our case assumed to be the zero motion action $a = \emptyset$, i.e., $w = (f, \emptyset)$. In contrast, for viewpoint selection, the motion action to reach a new viewpoint is obtained after (11) by letting $w \equiv (f, a)$ and solving for $a_t = a^* = \operatorname{argmax}_{a \in \Omega_a} s_t(f, a)$, without yet selecting a specific f. A first new feature is then extracted at the new viewpoint. In addition, a third available action is to end the recognition process, which applies whenever the stopping criteria (Section 4.4, and further specified throughout Sections 5 and 6) become true.

Our Bayesian inference framework leads naturally to an iterative information-theoretic approach for deciding on the next best action w^*. Information-theoretic formulations allow us to quantify actions in terms of information gain and uncertainty reduction, given our current belief state. Several methods from information theory have been shown to work well in action selection tasks. The more popular methods include *expected loss of entropy* (LE) [23,22], *mutual information* (MI) [3,4] and *Kullback–Leibler divergence* (K–L) [8].

First we will formulate all three methods in a general way adapted to our action selection framework, followed by the specific formulations for feature selection and viewpoint selection.

4.1. Expected loss of entropy

LE, on average, measures the expected reduction in uncertainty of our belief state. More concretely, it measures the difference in uncertainty between our current posterior belief $P(o_t|f_{1:t}, a_{1:t-1})$ and the predicted posterior belief $P(o_t|f_{1:t}, a_{1:t-1}, w)$, if we were to take an action w.

The entropy H quantifies the uncertainty of a random variable. We are interested in computing the entropy of the posterior distribution $P(o_t|f_{1:t}, a_{1:t-1})$ because it expresses a measure of uncertainty (respectively certainty) about our belief. The entropy is zero if the outcome of the experiment is unambiguous and it reaches its maximum if all outcomes of the experiment are equally likely. This means, if the uncertainty is high, multiple models are likely to explain the data. Finally, the entropy over the object classes is given by

$$H(o_t|f_{1:t}, a_{1:t-1}) = -\sum_{i=1}^{KQ} P(o^i|f_{1:t}, a_{1:t-1}) \log P(o^i|f_{1:t}, a_{1:t-1}). \quad (12)$$

Given the definition of the entropy, we can now define the utility score $s_t(w)$ for an action w based on LE as

$$s_t(w) = \sum_{i=1}^{KQ} P(o^i|f_{1:t}, a_{1:t-1})\, E[\Delta H(o^i, f_{1:t}, a_{1:t-1}, w)], \quad (13)$$

with KQ object models in the model set. LE is expressed as the expected difference in entropy $E[\Delta H(\cdot)]$; it represents our expectation on the difference in uncertainty between the current and predicted posterior distribution. This expected difference is weighted by our current belief $P(o^i|f_{1:t}, a_{1:t-1})$, such that the utility score reflects the reductions in likely object beliefs. Moreover, we must compute the average LE because we cannot be certain about our belief state—hence we average over all beliefs.

The expected difference in entropy for possible actions $w \equiv (f,\ a)$ is eventually computed as

$$E[\Delta H(o^i, f_{1:t}, a_{1:t-1}, w)] = H(o_t|f_{1:t}, a_{1:t-1}) - \int p(f|o^i, a)\, H(o_t|f_{1:t}, a_{1:t-1}, w)\, df. \quad (14)$$

As we do not know which feature to expect, we need to integrate over the entire feature space. The feature distribution $p(f|o^i, a)$ is given by the observation model. Since (14) is not possible to compute in closed form, we approximate by sampling from the feature distribution, and obtain

$$E[\Delta H(o^i, f_{1:t}, a_{1:t-1}, w)] \approx H(o_t|f_{1:t}, a_{1:t-1}) - \sum^{\Gamma} H(o_t|f_{1:t}, a_{1:t-1}, w), \quad (15)$$

where Γ is the number of samples drawn from the feature distribution $P(f|o^i, a)$. We can approximate (15) further by sampling with zero variance, which yields the mean μ_{o^i} of our trained object model o^i, and with that, a *mean feature* $f_{\mu_{o^i}}$. We eventually write

$$E[\Delta H(o^i, f_{1:t}, a_{1:t-1}, w)] \approx H(o_t|f_{1:t}, a_{1:t-1}) - P(f_{\mu_{o^i}}|o^i, a)\, H(o_t|f_{1:t}, f_{\mu_{o^i}}, a_{1:t-1}, a). \quad (16)$$

The evaluation of $s_t(w)$ is quite costly to compute since it involves a prediction step. With each $w \in \Omega$ we want to evaluate we need to compute the sequential update of the posterior distribution given the predicted observation. Even with the mean feature approximation of (16), this still involves the computation of KQ predictive posteriors. Restricting attention to only the most probable o^i object hypothesis can further reduce the computational cost.

4.2. Mutual information

In contrast to LE, which uses the predictive posterior distribution, MI relies on the conditional entropy to select the next action. We do not need to compute the update of the posterior distribution, which makes the evaluation cheaper and thus much faster.

The utility score is computed from the averaged MI as $s_t(w) = I(o_t; f|a)$, with $w \equiv (f,\ a)$, by

$$I(o_t; f|a) = \sum_{i=1}^{KQ} P(o^i|f_{1:t}, a_{1:t-1}) \times \int p(f|o^i, a) \log \frac{p(f|o^i, a)}{\sum_{j=1}^{KQ} p(f|o^j, a)} df. \quad (17)$$

Since this quantity again is not possible to compute in closed form, we approximate (17) with the mean feature $f_{\mu_{o^i}}$ as well, which results in

$$s_t(w) \approx \sum_{i=1}^{KQ} P(o^i|f_{1:t}, a_{1:t-1}) P(f_{\mu_{o^i}}|o^i, a) \log \frac{P(f_{\mu_{o^i}}|o^i, a)}{\sum_{j=1}^{KQ} P(f_{\mu_{o^i}}|o^j, a)}. \quad (18)$$

Generally speaking, the mutual information can be understood as the extent the uncertainty about variable o_t is reduced, when, given an action a, an observation f has been made.

4.3. Jeffrey's divergence

Reducing the ambiguity between object classes is equivalent to finding viewing positions which are inherently different. To this end, we can choose new viewing positions by selecting viewpoints such that, regardless of the true object and pose under observation, the most likely distributions of the resulting measurements are predictably dissimilar. Therefore, a new viewpoint must be selected in such a way that competing hypotheses will appear as different as possible, i.e., that a maximum dissimilarity is measured by the features.

A measure of dissimilarity or "distance" between observations is the relative entropy or *Kullback–Leibler divergence*. The KL-divergence presents the amount of information lost if samples from the distribution P_1 are assumed to be samples from P_2. The K–L divergence is defined as follows:

$$\mathrm{KL}(P_1 \parallel P_2) = \int_{-\infty}^{\infty} p_1(x) \log \frac{p_1(x)}{p_2(x)}\, dx. \quad (19)$$

However, $\mathrm{KL}(P_1 \parallel P_2)$ is not a true distance function since it is not symmetric and does not obey the triangle inequality. We

can formulate a symmetric version based on the K–L divergence, also known as *Jeffrey's Divergence* (JD), $J(P_1 \parallel P_2) = \mathrm{KL}(P_1 \parallel P_2) + \mathrm{KL}(P_1 \parallel P_2)$.

The evaluation of the Jeffrey's divergence is a potential computational bottleneck. In our case, the observation probability $P(f|o^i, a)$ is defined as a continuous probability density function and would require to evaluate an integral, which is not possible in closed form. Instead, we approximate the integral once more using the mean features. We evaluate the best action by computing the utility score

$$s_t(w) \approx \sum_{i=1}^{KQ} \sum_{j=1}^{KQ} P(o^i|f_{1:t}, a_{1:t-1}) P(o^j|f_{1:t}, a_{1:t-1}) \cdot J(P(f_{\mu_{o^i}}|o^i, a) \parallel P(f_{\mu_{o^i}}|o^j, a)). \quad (20)$$

4.4. Convergence and stopping criterion

The sequential update allows to continuously acquire new information and with that reduce the uncertainty of the belief state. Unfortunately, one cannot guarantee that the sequential decision process reduces the uncertainty in every single step. But, on average, by utilizing the information-theoretic action selection, the uncertainty will be reduced.

For information-theoretic methods, it is important to find adequate *stopping criteria*. In the case of feature selection, we need a stopping criterion that defines when to stop adding new features observed from the current viewpoint and instead rather move to a next viewpoint. Similarly, with respect to viewpoint selection, we need a stopping criterion that determines when to terminate the object recognition process and return the inference result made so far. Unfortunately, the results of an information-theoretic action selection method highly depend on the stopping criteria as well as the parameters controlling those criteria. Wrong choices can lead to sub-optimal performance, with all possible actions being processed in the worst case.

5. Feature selection

Modern sensors produce vast amounts of data, which enables the computation of costly and very high-dimensional features. Each feature in our total feature set is of value, since certain objects are better explained by some features than others. This means, on average, the total feature set increases the classification accuracy. But, seldom all the features are needed to correctly classify an object because many features are correlated. Features can be highly correlated across object classes, i.e., features can be very close to each other in feature space. The result is that, if such features are combined, they do not really decrease the uncertainty of our belief state.

Feature selection is a way to reduce the amount of features we need to compute during classification. Consequently, feature selection can help to decrease the computation cost of extracting features from raw data. This means, we want to extract as few features as possible, but at the same time, we want to extract those features that reduce the uncertainty of our belief state the most. Preferably, selected features will result in a unimodal distribution.

Feature selection can be performed at training time or at prediction time. At training time, the feature space is reduced in advance by selecting relevant features, such that the loss of significant information is minimal. At prediction time, features are selected and computed online. In this work, we train our classifier with all available features, while during prediction time, we only select and infer over features that yield new information. Online feature selection increases the flexibility of the classifier. It keeps complexity low during training and facilitates updates by adding new features without the need to re-train the entire training set.

Feature selection becomes especially important on robot platforms with minimal computation power and short operation time (e.g., quadcopter robots), since it allows us to only compute relevant features. In the following subsections, we describe two methods for feature selection, one using LE and one using MI.

5.1. Feature selection by expected loss of entropy

In feature selection, we are interested in finding the next best feature $f_{t+1} = f^*$. For feature selection by LE, this means that LE expresses the difference in uncertainty between the current belief state and the updated belief state after adding one additional feature f. During the selection process, we do not move the robot, so we set the motion action to the zero motion action, $a = \emptyset$. To select the next feature, we use the mean feature formulation. The action w now simply consists of selecting one mean feature $f_{\mu_{o^i}}$ we want to evaluate. $f_{\mu_{o^i}}$ is selected from the full mean feature vector $\boldsymbol{f}_{\mu_{o^i}}$ by marginalization. Since the feature selection step does not depend on the motion action a, we can neglect the term in the predictive posterior distribution. Finally, LE for feature selection is derived from (13) as

$$s_t(w) \approx \sum_{i=1}^{KQ} P(o^i|f_{1:t}, a_{1:t-1}) E[\Delta H(o^i, f_{1:t}, f_{\mu_{o^i}}, a_{1:t-1})], \quad (21)$$

with

$$E[\Delta H(o^i, f_{1:t}, f_{\mu_{o^i}}, a_{1:t-1})] = H(o_t|f_{1:t}, a_{1:t-1}) - P(f_{\mu_{o^i}}|o^i)\, H(o_t|f_{1:t}, f_{\mu_{o^i}}, a_{1:t-1}). \quad (22)$$

5.1.1. Stopping criterion

For feature selection by LE, we stop integrating more features when the expected loss of entropy for the selected feature is small, i.e., $\max(s_t(w)) < \tau_1$, where τ_1 is a constant positive threshold parameter.

5.2. Feature selection by mutual information

Similar to LE for feature selection, the feature selection by MI does not depend on the robot motion action either. Thus, using (18) and the mean feature formulation, we can compute the utility score as

$$s_t(w) \approx \sum_{i=1}^{KQ} P(o^i|f_{1:t}, a_{1:t-1})\, P(f_{\mu_{o^i}}|o^i)\, \log \frac{P(f_{\mu_{o^i}}|o^i)}{\sum_{j=1}^{KQ} P(f_{\mu_{o^i}}|o^j)}. \quad (23)$$

5.2.1. Stopping criterion:

We stop to select a new feature from the current location as soon as $s_t(w)$ has decreased compared to the previous value [5].

6. Viewpoint selection

The goal in view planning is to evaluate and select new viewpoints that yield novel and useful information about the object. Initially, we have not observed anything. So every observation has equal usefulness under the assumption that we do not have any prior knowledge and a uniform belief state. However, every subsequent observation from a different viewpoint bears additional information that may help to identify the object

correctly. The challenge is to evaluate potential new viewpoints which gain us new information, but also viewpoints that help us reduce the uncertainty of our belief state. We want to find new viewpoints that reduce the ambiguity and with that reduce the modes of our belief state. In the following subsections, we formulate three methods for viewpoint selection based on three different information-theoretic measures, namely LE, MI and the KL-based Jeffrey's divergence.

6.1. Viewpoint selection by expected loss of entropy

Compared to the LE formulation for feature selection, we now want to move the sensor to a new observation position, thus we need to incorporate the motion action a. In addition, we must define what feature to use. We again rely on the mean features $f_{\mu_{o^i}}$. However, in contrast to the case of feature selection above, we now use the full mean feature vector $\boldsymbol{f}_{\mu_{o^i}}$ instead of only the marginals. As the full set of features performs overall on average the best, it makes sense to employ the full vector including all mean features. Since we can sample the mean features from our training models, there is no overhead in feature computation, as there would be for making an actual observation. For viewpoint selection, we thus define the action as the tuple $w \equiv (\boldsymbol{f}_{\mu_{o^i}}, a)$. Again, we do not get around to compute a complete update of the posterior distribution—this time also including the transition of the sensor from the currently believed state to a new state by performing motion action $a_t = a^*$. Based on (13), we can now write

$$s_t(w) \approx \sum_{i=1}^{K\,Q} P(o^i|f_{1:t}, a_{1:t-1}) E[\Delta H(o^i, f_{1:t}, \boldsymbol{f}_{\mu_{o^i}}, a_{1:t-1}, a)], \tag{24}$$

with

$$E[\Delta H(o^i, f_{1:t}, \boldsymbol{f}_{\mu_{o^i}}, a_{1:t-1}, a)] = H(o_t|f_{1:t}, a_{1:t-1}) - P(\boldsymbol{f}_{\mu_{o^i}}|o^i, a)\, H(o_t|f_{1:t}, \boldsymbol{f}_{\mu_{o^i}}, a_{1:t-1}, a). \tag{25}$$

6.2. Viewpoint selection by mutual information

As seen for feature selection before, the difference between MI and LE is that for MI the second term in (25) is the conditional entropy instead of the entropy of the posterior. From (18), we now derive the utility score for viewpoint selection, using MI and the full mean feature vector $\boldsymbol{f}_{\mu_{o^i}}$, as

$$s_t(w) \approx \sum_{i=1}^{K\,Q} P(o^i|f_{1:t}, a_{1:t}) P(\boldsymbol{f}_{\mu_{o^i}}|o^i, a) \log \frac{P(\boldsymbol{f}_{\mu_{o^i}}|o^i, a)}{\sum_{j=1}^{K\,Q} P(\boldsymbol{f}_{\mu_{o^i}}|o^j, a)}. \tag{26}$$

The motion action a is indirectly used; the mean feature vector is generated, given a recognition state $o_t = o^i$, once the motion action a has been applied.

6.3. Viewpoint selection by Jeffrey's divergence

Similar to the MI formulation in (26) above, we can obtain the utility score $s_t(w)$ for the viewpoint selection by JD through the general action selection equation (20) from Section 4, utilizing the full vector of mean features $\boldsymbol{f}_{\mu_{o^i}}$:

$$s_t(w) \approx \sum_{i=1}^{K\,Q} \sum_{j=1}^{K\,Q} P(o^i|f_{1:t}, a_{1:t-1}) P(o^j|f_{1:t}, a_{1:t-1}) \cdot J(P(\boldsymbol{f}_{\mu_{o^i}}|o^i, a) \parallel P(\boldsymbol{f}_{\mu_{o^i}}|o^j, a)). \tag{27}$$

6.4. Stopping criterion

The stopping criterion for whether to select a new viewpoint or to terminate the recognition process depends on the entropy. For all three viewpoint selection methods (Sections 6.1–6.3), we will stop and return the MAP estimate of the posterior distribution whenever $H(P(o_t|f_{1:t}, a_{1:t-1})) < \tau_2$. τ_2 is a second threshold parameter, which is set to a constant positive value.

7. Experiments

In this section, we present the results from extensive evaluations of our framework on real world data and with a quadcopter robot. All the evaluations are performed on two different sets of features. The first feature set, *simple features*, are easy and fast to compute off-the-shelf features. The second set of features, *kernel descriptors*, are more complex, generalize better, but have a higher computation cost associated. The evaluation over the two different feature sets provides an insight into recognition performance and computational cost, and demonstrates that our system is general and can handle different kinds of feature types.

We report on the performance of individual modules and then demonstrate the performance of the complete system. First, we show the performance of single-view object recognition when using the full feature set, which serves as the baseline. Our feature selection framework achieves a reduction in computation time, while the recognition accuracy remains comparable to the accuracy obtained for the baseline evaluation. Following, we evaluate the unified feature selection and viewpoint selection approach in a multi-view object recognition setting. For both feature sets, we give an intuition of the trade-off between increased recognition accuracy and additional computation cost of the multi-view approach. In addition, we test our system for its applicability to resource-constrained platforms through quadcopter experiments as well as demonstrate its usability in related tasks like object change detection.

7.1. RGB-D object dataset

We evaluated our object recognition and action selection framework on the publicly available RGB-D object dataset introduced in [21]. The dataset contains 41 877 data points, of which we use roughly one-third as the test points. The dataset consists of a color image, a 3D point cloud and the orientation of the object, for a total of 300 everyday objects. For each object, the data is recorded from directions all around the object, including the three different viewing angles of 30°, 45° and 60° per direction (see Fig. 3 on the left and in the center). The overall dataset is challenging, since the objects in the dataset can look very similar, especially when they belong to the same object category. Furthermore, the objects in the dataset exhibit large changes in illumination due to the different viewing angles from which the objects were captured, which may make classification difficult even across different categories.

The feature vector $\boldsymbol{f}$, which is used throughout our framework, is comprised of N independent different features for each feature set. During this evaluation, we will use two different feature sets, one containing simple features, the other one containing the more complex kernel descriptors. In accordance with (10), each component of the feature vector is expressed as an independent normal distribution. The parameters of the independent normal distribution are learned during training. One big advantage of this representation is the possibility of sequential training. This means we can sequentially update already learned parameters or add a new model without re-training all model parameters. The scaling factor ψ^n is chosen empirical and set to {Color, Bbox, SIFT, VFH} =

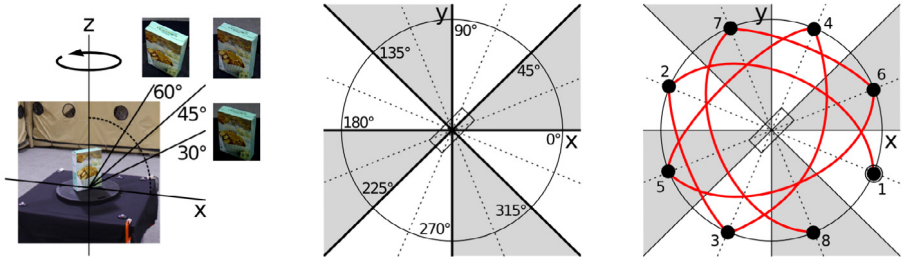

Fig. 3. Experimental setup. Left: Each object is placed on a turntable that rotates around the z-axis (yaw rotation) of the world coordinate frame. Object data is recorded from three different viewing angles of 30°, 45° and 60° per viewing direction. Center: Viewing directions denote directions in the x–y plane of the coordinate frame from which the object can be observed. The plane is subdivided into bins of 45° around the object. Observations of the object from the same bin will lead to similar estimations of the object's pose. Right: In multi-view object recognition, a sequence of viewpoints can be planned in different ways. The shown path (red) results from the simple heuristic "motion along a star pattern", which assumes that the viewpoint opposite to the current viewing direction yields the most information and thus is the one to visit next.

[1.0, 0.02, 0.2, 0.001] for the simple features. When using the Kernel descriptors we have set the scaling factor to 1.0 for all dimensions.

In all our experiments, we use the same evaluation procedure as proposed in [21,35,1]. To speedup the training process we subsample the training set to use every 5th data point. We test our framework on the task of recognizing the instance of an object. We train our object models from $\boldsymbol{f}$ using data captured from viewpoints at 30° and 60° in each direction, and test them against features computed from data that is recorded from viewpoints at 45°. Based on the computed features, we generate $Q = 8$ different object models per object instance, which clusters the viewing directions horizontally into bins spanning 45°. This subdivision trades computation cost off against performance, and defines the accuracy of the objects' pose estimates.

All the reported accuracies of our multi-view object recognition results are expressed as *recall* and are average values over 10 trials. Regarding the stopping criteria, we terminate the action selection procedure whenever either the entropy of the posterior distribution has been reduced to $\tau_2 = 0.1$ or the MAP estimate of the posterior distribution exceeds 60% in accuracy. For the LE formulation in the feature selection, we set $\tau_1 = 0.1$. After termination, we use the MAP estimate of the posterior distribution according to (3) to determine the most likely object.

7.2. HMM state transitions

As mentioned in Section 3.4, for this work, we have designed four simple state transitions, which are stored in a look-up table. The state transition probabilities are chosen with the following premise. When transitioning from one belief state o_t to the next belief state o_{t+1} given a motion action a_t, we want to favor a transition that stays in the same object instance and ends up in the correct object orientation. This allows us to incorporate the knowledge of how the robot has traveled during a motion action. Furthermore, we define transitions where the object belief would change into a different instance or category as not favorable. This prevents constant changing of the belief state from one instance to another, which occurs often for multimodal belief states. Learning the transition probabilities could certainly improve the overall performance and lead to higher recognition accuracies. However, we will leave the learning of the transition probabilities for future work, and for now use the state transitions defined as {[Same instance, correct angle = 0.4], [Same instance, wrong angle = 0.3], [Wrong category = 0.2], [Wrong instance = 0.1]}.

7.3. Simple features

We designed a set of simple features that are fast and easy to extract from RGB-D data. We will show that such simple features can be very powerful in terms of low computation time and achieve a high recognition accuracy in a multi-view setting. In total, we have designed $N = 4$ independent features, which are all included in the feature vector $\boldsymbol{f}$: object bounding box, color histogram, SIFT [2] and viewpoint feature histogram (VFH) [36]. Each feature represents a different object property: size, color, scale-invariant image description and geometry. In accordance with (10), each single feature or component of a feature histogram, respectively, is expressed as independent normal distribution. As we will see, the *color feature* is the fastest and most powerful single feature among the four features, and thus the first feature we evaluate at each viewpoint.

7.4. RGB-D kernel descriptors

We also tested our framework on the more complex state-of-the-art RGB-D kernel descriptors [1,37]. We use the code provided by the authors to compute the kernel descriptors. In total, the kernel descriptors consist of $N = 7$ features, each of which has the dimensionality of 1000. From the RGB-D data we extract three RGB features, two point cloud features and two depth features. The features are much more expressive than the simple features, however, this comes at the price of higher extraction cost for all features. Additionally, to compute the features, the entire training set needs to be available upfront. As the first feature for our feature selection process, we extract the *RGB gradient kernel*, which has performed the best as a single feature.

7.5. Online feature selection

As a start, we evaluate the usefulness of online feature selection in a single-view recognition task. We show that selecting the right subset of features can reduce the computation cost drastically, while keeping comparable recognition performance. We will show the performance and computation cost on the two feature sets. We present the evaluation for the simple features first, followed by the evaluation using the kernel descriptors.

7.5.1. Simple features

Each feature has a cost associated for computation; the average computation times of our simple features are as follows: 0.01 s for the bounding box, 0.0005 s for the color histogram, 0.025 s for the SIFT features, and 2.0 s for the VFH. Fig. 4 shows the average total cost of computation per object and the recognition accuracy plotted against individual feature combinations. The color feature seems to have the best single recognition performance and is the cheapest to compute. The VFH on the other hand does not have a lot of recognition power as such and is rather expensive. From a cost vs. performance standpoint, the feature combination {Color, Bbox, SIFT} performs best with an accuracy of 68.91% and a low average feature computation cost of 0.04 s. However, note that the

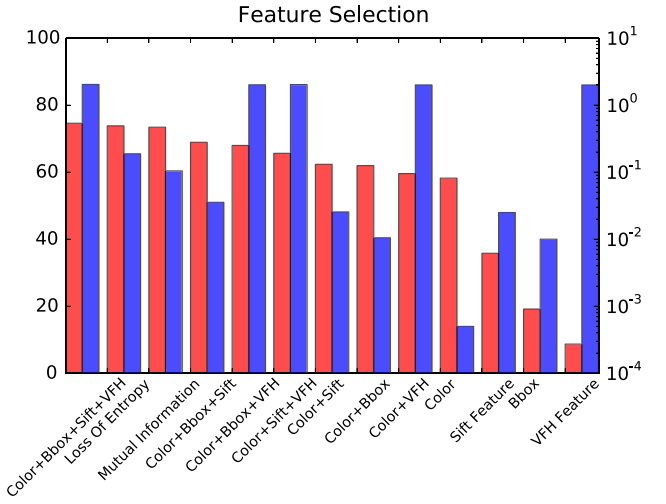

Fig. 4. Simple features—Performance and cost of different feature combinations in single-view object recognition. The recognition accuracies are displayed as red bars (left side scale, in %). The average cost of computing the feature combinations per object is shown in log scale as blue bars (right side scale, in seconds). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

VFH also has its value since, if included, it increases the recognition accuracy to 74.59%, as detailed in Table 1.

Table 1 clearly shows that employing feature selection can reduce the average total cost of the feature computation considerably (see the third and fourth row of column "F Cost" in Table 1). Note that the recognition accuracy keeps similar values as if all the features were used. Here, we utilize LE and MI for feature selection according to Section 5. Among the two different feature selection methods, the LE measure seems to select more features, and more importantly, when compared to MI, is computationally much more expensive because we need to update the posterior distribution and compute a full update step.

We compared all results with the results of the original dataset in [21] and [1]. The first paper has very comparable results to our HMM framework and uses similar features to ours. No timing has been provided but the computation cost is expected to be equal to (or slightly higher than) the cost of our features. The latter paper reports much higher recognition accuracies; this is mainly due to a more descriptive feature which seems to generalize very well (also refer to the following subsection). The cost for such a feature, however, is a very high computation time. Moreover, the features used in the two papers are learned in batch, which makes them less flexible compared to our independent feature learning approach.

7.5.2. RGB-D kernel descriptors

Similar to the simple features, we evaluate the performance of our feature selection framework when used together with the kernel descriptors. In Table 2, we show the accuracy and cost for single-view object recognition in combination with feature selection. The first row of the table shows the recognition performance of the kernel descriptor with our Bayesian framework. We can see that compared to the linear SVM classifier used in [1], our accuracies are slightly lower with 90.3% vs. 91.2%, respectively. Furthermore, we can see that feature selection using the LE formulation, compared to the MI formulation, is still significantly slower due to the increased computational overhead. Both approaches, however, show a significant reduction in the average computation time of the features, meaning significantly less time is spent extracting features, when the feature selection is used. For the MI formulation, for example, we achieve a reduction from 3.2 to 0.83 s. The MI formulation arrives at the lowest total cost of 1.29 s for feature computation and feature selection. This is a reduction of a factor of 2.5 in total computation time. Even though we substantially reduce the cost, and with that the amount of features to extract, the recognition accuracy stays comparable to the case where the full feature set is used. The MI approach achieves 90.61% in accuracy.

Table 1
Simple features—Individual performance for single-view object recognition. "Loss Entropy" and "Mutual Information" denote the feature selection method, with "F Cost" being the average cost per object for computing the features and "FS Cost" being the average cost per object for feature selection.

Method	Accuracy	F Cost (s)	FS Cost (s)	Total (s)
C + B + S	68.91%	0.04	0.00	0.04
C + B + S + V	74.59%	2.04	0.00	2.04
Loss entropy	73.82%	0.19	1.05	1.24
Mutual information	73.21%	0.1	0.2	0.3
[21] (kSVM)	74.8%	–	–	–
[1] (Kernel descriptor)	91.2%	3.2	0.0	3.2

Table 2
Kernel descriptors—Individual performance for single-view object recognition. "Loss Entropy" and "Mutual Information" denote the feature selection method, with "F Cost" being the average cost per object for computing the features and "FS Cost" being the average cost per object for feature selection.

Method	Accuracy	F Cost (s)	FS Cost (s)	Total (s)
Kernel descriptor	90.3%	3.2	0.00	3.2
Loss entropy	89.94%	0.88	1.71	2.59
Mutual information	90.61%	0.83	0.46	1.29
[1] (Kernel descriptor)	91.2%	3.2	0.00	3.2

7.5.3. Discussion

Overall, we can see that feature selection reduces the set of features to extract – and thus the computation cost – drastically. The result that the accuracy is comparable, or in some cases even slightly better than when using the full feature set, further validates the usefulness of our approach.

The feature selection in the case of simple features reduces the cost of feature computation the most. From Fig. 4 we see that the VFH feature does not hold a lot of discriminative information for most of the objects. This is reflected in the large reduction in computation time, since most of the times, the VFH feature is not computed, hence the low average feature computation time of 0.1 s.

A similar trend can be observed in the reduction of the average feature computation time for the more complex kernel descriptors. However, here the reduction is not as pronounced. More features are useful, but more importantly, there is not one single feature that has, compared to all the others, a large computation cost, which can be saved.

The average computation cost for computing the next informative feature is higher for the kernel descriptors. The kernel descriptors are of higher dimensionality in total, which slightly increases the computation time. In contrast to the simple features, the total average cost of the kernel descriptors is about a factor of four higher. However, this increase in computation cost is well justified, since it is accompanied by a large increase of 15% in recognition accuracy.

7.6. Multi-view object recognition

We now present the performance of our unified approach of feature and viewpoint selection. We evaluate the performance for different view planning methods; in total, we compare five different ways to estimate the next viewpoint: *Random*, *Loss of Entropy*, *Mutual Information*, *Jeffrey's Divergence* as well as with a simple *Heuristic*. The simple heuristic serves for comparison with our information-theoretic approaches and is based on

the assumption that viewpoints opposite the current viewing directions yield the most information. Intuitively, the sensor then moves in a "star pattern" around the target object to acquire new information from the opposite side, as shown by the path in Fig. 3 on the right.

7.6.1. Simple features

Although the computation time can be substantially reduced through feature selection, the achievable recognition accuracies are limited around 75% for single-view object recognition. The reasons are the relative simplicity of our four feature types and their limitation in generalizing well for the given RGB-D dataset. However, even with fairly simple features, we can achieve improved recognition accuracies for both object class and pose by utilizing an information-theoretic multi-view object recognition approach. In the following tests, we use the MI measure for feature selection, since it is on par with the LE formulation regarding recognition performance, but the computational cost (see Table 1) is significantly lower.

Table 3 shows the performance and Table 4 summarizes the cost of multi-view object recognition. The first row shows that using random viewpoint selection with all features (i.e., no feature selection) has a very high cost but already increases the recognition accuracy by about 10%. In the second row, we can see a reduction in computation time and an increase in accuracy due to MI-based feature selection and random viewpoint selection. On average, random viewpoint selection terminates after a little more than 3 observations. The simple heuristic performs a little better than random selection, but it similarly needs on average about 3 observations to achieve a comparable accuracy. The computation time is also very similar. However, the simple heuristic does not perform as well as the information-theoretic methods. By using the information-theoretic measures for viewpoint selection, we find that LE, MI and JD all perform similarly well—with MI being the best in terms of recognition accuracy. The lowest computation time is achieved by using JD because it can compute new viewing positions in almost no time. The MI formulation is superior to the LE formulation in speed, since it does not need to compute the predictive posterior.

We further evaluate our multi-view approach for estimating an object's pose. As previously mentioned, we train each object with a number of different object models, in our case $Q = 8$ models per object. When observing a feature, we can infer from which model it has most likely been generated, and by this, further infer about the orientation of the object. The performance of the object's pose estimation is presented in Table 3. The table shows the average bin error of the estimated viewing direction, which is computed from the difference between the inferred and the true observation angle. For the first row, all the features were computed (i.e., no feature selection) and the viewpoints were selected at random. All the other experiments use the MI formulation to select the informative features at each given observation position. On average, using random and heuristic viewpoint selection approaches, we are >2 bins off in our pose estimation, which corresponds to an average pose error of about 67.5°. If we apply one of the three more sophisticated viewpoint selection methods, we yield an average bin error of <2 bins. This results in an average pose error of 45°. For comparison, the experiments in [35] report an average pose error of 44.8°, which is on par with our results.

7.6.2. RGB-D kernel descriptors

Similar to the simple features, we evaluate the kernel descriptors with our unified feature and viewpoint selection framework. Tables 5 and 6 summarize the performance of accuracy and cost, respectively. From Table 5, we see that, by using a multi-view approach with random viewpoint selection but no feature selection, the recognition accuracy, has increased from 90.61% to 95.01%, with 1.8 observations on average (first row). Furthermore, we can see that random and heuristic viewpoint selection perform almost identical, still the heuristic achieves a slightly higher accuracy but also needs on average slightly more observations. In contrast, the information-theoretic viewpoint selection methods perform again better as the simple ones, with the best result achieved by the MI formulation. The MI formulation achieves a respectable recognition accuracy of 98.14% and on average needs about 1.4 observations. Interestingly, the kernel descriptors reach similar results in terms of pose estimation as the simple features. We think the reason is that objects still look very similar when moving the observation position within 45°, making it hard to classify a target object as the exact object (i.e., correct object class and orientation angle) the model was created from.

Additionally, we also evaluated the computation cost of individual methods and modules, as can be seen in Table 6. Due to its high computation cost, we can see, that the LE formulation is by far the most costly, making it again a poor choice—especially, when compared to the other information-theoretic approaches. The MI formulation achieves the highest accuracy and requires the lowest number of observations, however, the cost is higher than for the random, heuristic or JD formulation. Overall, the fastest method is the method that is based on JD; it only requires 2.36 s of computation time.

7.6.3. Discussion

With both feature sets, we can clearly see that having the option of taking additional observations increases the recognition accuracy. In the case of the simple features, the increase is over 15%. With about 8%, the kernel features do not experience such a large gain, however, the single-view recognition rate has here already been relatively high.

Information does not automatically mean useful information—this can especially be seen when we compare the heuristic with the MI-based viewpoint selection approach. Adding new information does not necessarily add valuable knowledge to find the correct object class. In terms of information content, the heuristic should add the most information, since we move the sensor to the complete opposite sides. Instead, when positions are chosen in terms of uncertainty reduction, which also reduces object ambiguity, better performance can be observed.

In comparison to the single-view approach, the computation time is increased due to computing the next viewpoint—but also due to travel time of the robot. Depending on the robot platform additional costs may be associated with an action like motion planning or motion risk. Currently we do not incorporate any extra costs associated with motion, however we think that could be an interesting future research direction.

For the simple feature, on average, we need to take 1–2 additional views for an improved accuracy. When using the kernel descriptors instead, it suffices to take none or only one additional observation on average. However, the computation cost for the kernel descriptors is about 2.7 times higher (when using the JD measure) compared to the simple features.

The JD formulation is by far the fastest, i.e., a factor of about 1.7 times faster compared to MI in finding a new viewpoint. In contrast, the MI method achieves about 2% better accuracy and needs on average less observations. Finally, view planning with the MI formulation seems to pick better observation positions, compared to the JD formulation, trading off accuracy for speed.

For many robotic applications, for instance manipulation, the estimated object pose estimates are not precise enough. However, the estimation of the pose within a recognition framework is of great value, because in a sense we get this information for free. For State-of-the-art object pose estimation methods to work, a rough initialization of the object pose is required. The values estimated in this work are well suited as an initial pose for refinement methods like ICP.

Table 3
Simple features—Performance of our multi-view object recognition framework. All but the first row uses the MI measure in the feature selection method. We show the average recognition accuracy, the average number of observations and the average bin error of estimated viewing directions. Each bin represents a range of observation angles of 45°.

Method	Accuracy	Observations	AvgBin Err	MedBin Err
C + B + S + V*	84.48% ± 1.4	3.4 ± 0.2	2.41 ± 0.07(bins)(±67.5°)	2.13 ± 0.09(bins)
Random	86.76% ± 0.4	3.2 ± 0.2	2.19 ± 0.02(bins)(±67.5°)	2.21 ± 0.05(bins)
Heuristic (Star)	87.16% ± 0.6	3.1 ± 0.3	2.28 ± 0.02(bins)(±67.5°)	2.19 ± 0.09(bins)
Jeffrey's divergence	88.34% ± 0.6	2.7 ± 0.1	1.62 ± 0.01(bins)(±45.0°)	1.52 ± 0.03(bins)
Loss entropy	89.82% ± 1.3	2.4 ± 0.2	1.54 ± 0.01(bins)(±45.0°)	1.51 ± 0.02(bins)
Mutual information	91.87% ± 0.7	2.5 ± 0.1	1.48 ± 0.01(bins)(±45.0°)	1.45 ± 0.03(bins)

* (No feature selection).

Table 4
Simple features—Average computation cost per object for our multi-view object recognition framework. The columns show the average cost for computing the feature ("F Cost"), average cost for feature selection ("FS Cost") and average cost for viewpoint selection ("VS Cost").

Method	F cost (s)	FS cost (s)	VS cost (s)	Total cost (s)
C + B + S + V*	7.02 ± 0.42	0.00	0.0	7.02 ± 0.42
Random	0.40 ± 0.04	0.83 ± 0.05	0.0	1.23 ± 0.09
Heuristic (Star)	0.45 ± 0.08	0.79 ± 0.04	0.0	1.24 ± 0.12
Jeffrey's divergence	0.30 ± 0.03	0.58 ± 0.02	3.E−4 ± 2.E−4	0.88 ± 0.05
Loss entropy	0.23 ± 0.04	0.50 ± 0.04	5.4 ± 0.6	6.13 ± 0.68
Mutual information	0.24 ± 0.03	0.51 ± 0.03	1.1 ± 0.2	1.85 ± 0.26

* (No feature).

Table 5
Kernel descriptors—Performance of our multi-view object recognition framework. All but the first row uses the MI measure in the feature selection method. We show the average recognition accuracy, the average number of observations and the average bin error of estimated viewing directions. Each bin represents a range of observation angles of 45°.

Method	Accuracy	Observations	AvgBin Err	MedBin Err
Kernel descriptors*	95.01% ± 0.8	1.9 ± 0.1	1.95 ± 0.06(bins)(±45.0°)	2.01 ± 0.02(bins)
Random	95.16% ± 0.6	1.9 ± 0.3	1.85 ± 0.03(bins)(±45.0°)	1.94 ± 0.04(bins)
Heuristic (Star)	95.83% ± 0.4	2.0 ± 0.1	1.88 ± 0.03(bins)(±45.0°)	1.91 ± 0.02(bins)
Jeffrey's divergence	96.32% ± 0.5	1.5 ± 0.2	1.53 ± 0.01(bins)(±45.0°)	1.49 ± 0.07(bins)
Loss entropy	97.98% ± 1.0	1.4 ± 0.3	1.41 ± 0.01(bins)(±45.0°)	1.43 ± 0.01(bins)
Mutual information	98.14% ± 0.4	1.4 ± 0.2	1.43 ± 0.02(bins)(±45.0°)	1.39 ± 0.03(bins)

* (No feature selection).

Table 6
Kernel descriptors—Average computation cost per object for our multi-view object recognition framework. The columns show the average cost for computing the feature ("F Cost"), average cost for feature selection ("FS Cost") and average cost for viewpoint selection ("VS Cost").

Method	F Cost (s)	FS Cost (s)	VS Cost (s)	Total Cost (s)
Kernel descriptors*	6.11 ± 0.31	0.0	0.0	6.11 ± 0.31
Random	2.51 ± 0.16	0.94 ± 0.10	0.0	3.45 ± 0.26
Heuristic (Star)	2.56 ± 0.11	0.98 ± 0.08	0.0	3.54 ± 0.19
Jeffrey's divergence	1.64 ± 0.09	0.69 ± 0.02	0.03 ± 0.05	2.36 ± 0.16
Loss entropy	1.43 ± 0.1	0.67 ± 0.04	8.1 ± 0.5	10.02 ± 0.54
Mutual information	1.45 ± 0.13	0.68 ± 0.03	1.9 ± 0.4	4.03 ± 0.56

* (No feature selection).

7.7. Quadcopter experiments

In order to show the overall performance of our active multi-view object recognition framework, we have conducted experiments on our custom-built quadcopter platform. As shown in Fig. 5, the quadcopter is equipped with an ASUS Xtion RGB-D camera for data acquisition and can fly autonomously to a new observation position. Given a target object, the task is to fly the quadcopter around the object and take as many observations as necessary to correctly recognize the object. Compared to the simulated experiments above, where data is captured with an RGB-D camera mounted on a stable tripod, experiments on a robotic platform are generally more challenging. Specific challenges regarding object recognition on a fast moving platform include motion blur and constant changes in illumination.

As we do not have any of the objects used in the RGB-D dataset at hand, we captured 10 new objects and added them to the dataset, which now consists of a total of 310 objects. We captured the data in the same way as described in [21] and chose objects similar to the ones already contained in the dataset. We trained our models with all 310 objects and the two viewing directions of 30° and 60°. In the actual experiment, the quadcopter then flies around the object at a distance and height that allows for a viewing angle of 45° roughly.

In each run of the experiments, the robot starts at a random position with respect to the target object, from where it acquires the first observation of the object. From there on, it selects the following actions as described in Sections 4–6—either inferring about another feature from $\boldsymbol{f}$ or flying to a new viewpoint. We terminate the process once our stopping criteria are reached. Fig. 5 shows an example of the quadcopter robot taking observations of a cereal box object during one of the experiments.

7.7.1. Simple features

Table 7 summarizes the experimental results from the quadcopter experiments, averaged over eight runs with different starting positions. We can see that for all experiments, whether

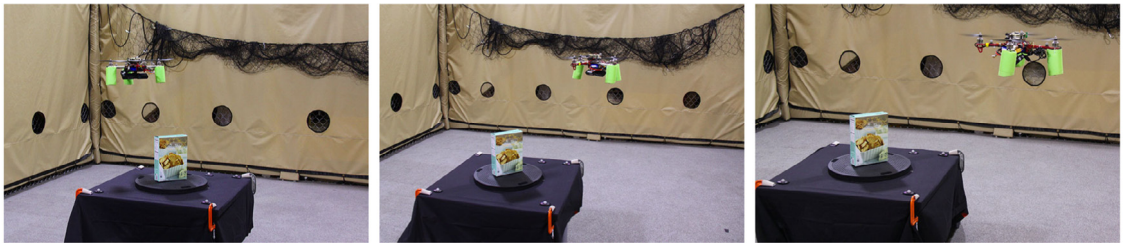

Fig. 5. Quadcopter experiment. The quadcopter robot, equipped with an RGB-D camera, collects multiple views of a target object (e.g., a cereal box) in an active object recognition task.

Table 7
Simple features—Single-view and multi-view object recognition based on data from the quadcopter robot. In both cases, results with and without feature selection are presented.

Method	Accuracy	Obsv.	Avg. bin error	Avg. cost (s)
Single-view				
C + B + S + V*	82.16%	1	2.1(bins)(±67.5°)	2.14
Feature selection (MI)	87.32%	1	1.9(bins)(±45°)	0.19
Multi-view				
Mutual information*	96.5% ± 0.5	2.0 ± 0.1	1.6 ± 0.1(bins)(±45°)	5.63 ± 0.3
Heuristic (Star)	95.7% ± 0.3	2.2 ± 0.2	1.9 ± 0.2(bins)(±45°)	0.98 ± 0.1
Mutual information	99.8% ± 0.1	1.6 ± 0.01	1.1 ± 0.1(bins)(±45°)	1.48 ± 0.2

* (No feature selection).

conducted with single-view or multi-view object recognition, online feature selection helps to decrease the computation time and increase the recognition accuracy. The average pose error is similar to the error in the simulation experiments, however, the average bin error is smaller, which results in more correctly classified bins in total.

Multi-view recognition on average only required 1.6 observations, which means that often we are able to make a confident decision after only one observation. In some instances, however, by acquiring additional observations, the recognition accuracy could be increased by more than 10% to totally 99.8%. This, of course, comes at the price of actually moving the robot to the new viewpoint, which in turn increases the cost of recognizing the object. Whether the extra cost of moving is justified in practice, finally depends on the robot platform used, the surrounding environment, as well as the trade-off between desired recognition accuracy and energy savings. With respect to online feature selection, we can clearly see that the adaptive selection of features offers huge benefits by decreasing the computation time, which is especially beneficial for vehicles with low energy budgets like quadcopters.

7.7.2. Kernel descriptors

We present results of the quadcopter experiments using the kernel descriptors in Table 8. Again, the kernel descriptors generalize much better than the simple features, achieving a much higher recognition accuracy in single-view object recognition. However, the cost of recognizing the target object is much higher compared to the simple features. The MI-based approach for multi-view object recognition achieves 100% recognition accuracy with on average 1.2 observations needed. It seems that the pose estimation has also improved by the more complex features, with the best result achieving an average angle error of 22.5° only. Basically, we can see that the recognition result of the single-view approach is already very good. The multi-view approach can be seen as a complement for these experiments—often, it only takes one view to recognize the object, at other times, there is a benefit from taking more than one observation.

7.7.3. Discussion

The experiments in this subsection show our multi-view system applied to a quadcopter to perform information gathering. First of all, the experiments confirm the practicability of our approach regarding the real application on a robot platform. The accuracy presented for these experiments will probably not hold in general. Although we added 10 objects to the database, which we tried to pick, such that they were similar and of the same categories as the objects already contained in the database, they seem to be different enough to show a rather drastic increase in recognition accuracy. Therefore, the accuracy results need to be taken with some caution. Nevertheless, what the system does show is the advantages of a multi-view system in combination with online feature selection. We again achieve an increase in accuracy when given the chance of taking additional observations. Furthermore, we see a reduction in computation time thanks to the online feature selection framework. Given the limited processing capabilities of a quadcopter, these are very promising results.

With regard to energy consumption and cost to move, quadcopters are somewhat special platforms. To be able to hover or fly, the energy a quadcopter consumes is in general quite high. Compared to other robots which do not consume a lot of battery power when stationary (e.g., ground or aquatic robots), for quadcopters, moving to a new position by flying does not cost that much more energy than staying in place by hovering. This makes an active multi-view object recognition system, as the one we have presented in this paper, particularly suitable for quadcopter robots.

7.8. Object change detection

As a final application, we now show how to utilize our framework for detecting changes in the environment. Given a prior map of the environment and an object database, we want to detect if objects in the environment have been replaced, taken away or changed their pose. We use our generative object model to express the probability $P(o^i|f)$ that the object has been generated by the feature f. Given an expected object model $\hat{o}^i$ (prior information), we can compute the probability $P(u)$ that the object has changed with $P(u) = 1 - P(\hat{o}^i|f)$. If the probability $P(u) > \tau$, with $\tau \in [0,\ 1]$, we know that the feature observed cannot be generated by the model $\hat{o}^i$, hence either object class or object pose has changed. This can also occur when the feature does not match the model anymore due to changes in lighting or noise; in either case we need to perform further observations to arrive at a definitive conclusion

Table 8
Kernel descriptors—Single-view and multi-view object recognition based on data from the quadcopter robot. Results with and without feature selection are presented.

Method	Accuracy	Obsv.	Avg. bin error	Avg. cost (s)
Single-view				
[1] (Kernel descriptor)	94.27%	1	1.5(bins)(±45.0°)	3.2
Kernel descriptors*	93.67%	1	1.6(bins)(±45.0°)	3.2
Feature selection (MI)	94.05%	1	1.5(bins)(±45°)	1.7
Multi-view				
Mutual information*	100.0%	1.3 ± 0.05	1.3 ± 0.03(bins)(±45.0°)	9.42 ± 0.5
Heuristic (Star)*	100.0%	1.5 ± 0.07	1.1 ± 0.1(bins)(±45°)	2.37 ± 0.1
Mutual information	100.0%	1.2 ± 0.02	0.7 ± 0.2(bins)(±22.5°)	3.63 ± 0.3

* (No feature selection).

Table 9
Simple features—Scenario where the object has not changed compared to the prior knowledge. The table shows the detection accuracy, number of observations needed as well as the total average computation cost in seconds.

Method	Accuracy	Observations	Avg. cost (s)
Random	91.19 ± 0.19	2.2 ± 0.18	1.02 ± 0.36
Heuristic (Star)	92.82 ± 0.26	2.1 ± 0.21	0.91 ± 0.49
Jeffrey's divergence	96.13 ± 0.62	1.4 ± 0.12	0.54 ± 0.27
Loss entropy	96.71 ± 0.48	1.3 ± 0.2	3.43 ± 0.49
Mutual information	98.33 ± 0.53	1.3 ± 0.13	0.83 ± 0.38

Table 10
Kernel descriptors—Scenario where the object has not changed compared to the prior knowledge. The table shows the detection accuracy, number of observations needed as well as the total average computation cost in seconds.

Method	Accuracy	Observations	Avg. Cost (s)
Random	95.92 ± 0.11	1.9 ± 0.12	3.3 ± 0.47
Heuristic (Star)	96.83 ± 0.15	1.8 ± 0.14	3.2 ± 0.51
Jeffrey's divergence	99.15 ± 0.17	1.3 ± 0.23	2.15 ± 0.27
Loss entropy	99.81 ± 0.21	1.2 ± 0.19	9.01 ± 0.26
Mutual information	99.74 ± 0.24	1.2 ± 0.17	3.8 ± 0.23

Table 11
Simple features—Scenario where both the object and pose have changed compared to the prior knowledge. The table shows the detection accuracy, number of observations needed as well as the total average computation cost in seconds.

Method	Accuracy	Observations	Avg. cost (s)
Random	80.24 ± 0.25	2.4 ± 0.18	0.98 ± 0.37
Heuristic (Star)	81.71 ± 0.28	2.5 ± 0.21	1.12 ± 0.32
Jeffrey's divergence	86.25 ± 0.36	1.9 ± 0.12	0.78 ± 0.23
Loss entropy	89.25 ± 0.27	1.7 ± 0.2	3.03 ± 0.43
Mutual information	88.95 ± 0.31	1.8 ± 0.13	1.16 ± 0.29

Table 12
Kernel descriptors—Scenario where both the object and pose have changed compared to the prior knowledge. The table shows the detection accuracy, number of observations needed as well as the total average computation cost in seconds.

Method	Accuracy	Observations	Avg. cost (s)
Random	90.84 ± 0.25	2.2 ± 0.21	3.72 ± 0.53
Heuristic (Star)	89.75 ± 0.28	2.1 ± 0.19	3.61 ± 0.46
Jeffrey's divergence	92.68 ± 0.36	1.8 ± 0.26	3.01 ± 0.36
Loss entropy	94.91 ± 0.27	1.5 ± 0.16	10.97 ± 0.41
Mutual information	93.17 ± 0.31	1.4 ± 0.21	4.12 ± 0.29

about the change. If we are uncertain about the object after our first observation, we compute new viewpoints that allow for observations that are unique in terms of feature space. We acquire new observations, evaluate additional features and compute new likelihoods of the features given the expected object model $\hat{o}^i$.

In the following, we explore three different scenarios, when (1) the object has not changed, (2) the object has changed completely (i.e., it belongs to a different object class), or (3) the object has changed its orientation only (i.e., it belongs to the same class but its rotation differs in yaw). All experiments are conducted with threshold $\tau = 0.6$, meaning that we need to have a high enough certainty of the object being changed. All experiments are conducted with feature selection by MI, and then evaluated for the different view planning methods.

7.8.1. No object change

No change in object and pose refers to the scenario where the object and pose remain the same as the reference object given our prior map knowledge. This means, we want to validate whether the measurement(s) we take of the currently observed object conforms with the object we expect to see. Knowing which object we expect to see makes the recognition or validation task somewhat easier since observations need to conform very strongly with the expected object model. Similarly, when selecting new viewpoints, the expected feature at a new observation position will be very similar to the actual measurement. This makes it easier to select new viewpoints, which leads to observations that are very dissimilar to observations of other likely object candidates.

In Tables 9 and 10, we show the results for the case of no change in the environment, both for the simple features and the kernel descriptors, respectively. In the two tables, we can see that both feature sets perform very similar with respect to accuracy of detecting no change and number of required observations. Both feature representations achieve on average accuracies of more than 98% and need 1.3 or fewer observations. However, we can see that the simple features are by far less expensive to compute, with 0.83 s for the simple features compared to 3.8 s for the kernel descriptors. The high accuracies show that the task of validating the measurements with respect to the expected object model is easier compared to the task of pure object recognition without any prior knowledge.

7.8.2. Object change

In this scenario, the object and pose have changed compared to the map previously acquired. The task here is to detect that the target object has indeed changed compared to the expected object model. Compared to the object recognition results, this task is more difficult due to the stricter criteria for correct estimation of change. In the object recognition setting, for correct recognition, we require the MAP estimate to be the correct object. In contrast, here, to be correctly classified as change, we require the probability $P(u) > \tau$. This means, objects that look very similar and are only observed from similar positions will also have similar likelihoods. The result is that $P(u)$ will in general not be greater than τ. Finding new observation positions with object beliefs that are inherently different from other object beliefs is equally difficult as for the object recognition task, since the prior knowledge does not provide us with any further knowledge about the current object we observe. And with that, the expected features when moving the sensor might be different from the features actually measured after moving the sensor.

In Tables 11 and 12, we show again results for both feature representations. We can see that both accuracies are lower

Table 13
Simple features—Scenario where only the object's pose (yaw angle) has changed. The table shows the detection accuracy, number of observations needed as well as the total average computation cost in seconds.

Method	Accuracy	Observations	Avg. cost (s)
Random	87.15 ± 0.31	2.7 ± 0.23	1.31 ± 0.29
Heuristic (Star)	88.68 ± 0.36	2.6 ± 0.16	1.21 ± 0.36
Jeffrey's divergence	95.97 ± 0.25	1.7 ± 0.15	0.61 ± 0.19
Loss entropy	97.75 ± 0.14	1.5 ± 0.11	2.72 ± 0.31
Mutual information	97.12 ± 0.18	1.6 ± 0.16	0.91 ± 0.21

Table 14
Kernel descriptors—Scenario where only the object's pose (yaw angle) has changed. The table shows the detection accuracy, number of observations needed as well as the total average computation cost in seconds.

Method	Accuracy	Observations	Avg. cost (s)
Random	93.23 ± 0.44	2.2 ± 0.18	3.92 ± 0.13
Heuristic (Star)	93.98 ± 0.31	2.2 ± 0.20	3.89 ± 0.16
Jeffrey's divergence	96.48 ± 0.19	1.6 ± 0.21	3.01 ± 0.21
Loss entropy	98.42 ± 0.21	1.4 ± 0.16	10.12 ± 0.34
Mutual information	98.11 ± 0.36	1.4 ± 0.11	4.51 ± 0.27

compared to the object recognition task in Section 7.6 due to the stricter recognition criteria. On average, we achieve correct change classification for 88.95% of the objects using the simple features and for 93.17% of the objects using the kernel descriptors. Again, similar to the object recognition task, the kernel descriptors seem to be more descriptive, achieving better accuracy, with the trade-off of a higher cost. The computation cost of the kernel descriptors is by about a factor of four higher compared to the simple features.

7.8.3. Pose change

In the third scenario only the object pose changes, while keeping the object the same. In our case, this means the object has rotated around its z-axis (yaw). Given our rough object pose estimation based on model training for different viewing directions, we are only able to estimate pose changes that exceed the angle discretization. In our case, we can possibly detect pose changes that are greater than 22.5° in yaw angle. A correct classification is made when the probability $P(u) > \tau$, indicating the object has changed its orientation.

Compared to the previous case, where the entire object has changed, a pose change is again easier to detect, since the expected object corresponds to the object we are observing. This in turn simplifies the task of finding new viewpoints that are different in feature space from other likely candidates. This, of course, is only a problem for objects that look very similar to other objects (in feature space) contained in the database.

In Tables 13 and 14, we show results for the simple features and kernel descriptors. We can see that the accuracy results for the two feature representations are fairly similar with 97.12% for simple features and 98.11% for kernel descriptors, respectively. The kernel descriptors need on average 0.2 observations less but again have a higher computation time of about a factor of four.

7.8.4. Quadcopter experiments

Similar to the quadcopter experiments in Section 7.7, we also evaluated the change detection framework on our quadcopter platform. The experimental setup remains the same as with the previous quadcopter experiments. However, instead of recognizing a target object, we are interested in detecting any change of the object given prior information. This again means, we want to detect whether the object remains unchanged, the complete object has changed, or only the pose of the object has changed.

The experiments conducted make use of the MI formulation for feature selection as well as viewpoint selection, with all parameters remaining set as before. In Table 15, we show the averaged results for the detection accuracy, number of observations needed and computation cost in seconds, evaluated for the simple features as well as the kernel descriptors. The quadcopter experiments confirm that detecting a complete change of an object with certainty is more difficult than detection in the other two cases. Furthermore, the quadcopter experiments show results comparable with the results of the simulation experiments evaluated on the two feature sets.

Table 15
Quadcopter experiments—Feature selection and viewpoint selection are evaluated for the three different change detection scenarios using MI. The table shows the detection accuracy, number of observations needed as well as the total average computation cost in seconds.

Method	Accuracy	Observations	Avg. cost (s)
Simple features			
No change	98.3 ± 0.21	1.2 ± 0.19	0.79 ± 0.25
Pose change	99.4 ± 0.12	1.3 ± 0.22	0.89 ± 0.32
Object change	92.4 ± 0.15	1.9 ± 0.27	1.63 ± 0.46
Kernel descriptors			
No change	100	1.4 ± 0.13	4.01 ± 0.48
Pose change	100	1.6 ± 0.11	4.35 ± 0.29
Object change	98.5 ± 0.15	2.1 ± 0.12	5.46 ± 0.38

7.8.5. Discussion

The experimental results show that, given some prior information in form of a map, our framework can also be used to detect change in the environment. In our multi-view framework, on average, we can detect that no changes have occurred with an accuracy higher than 98% and a required number of equal or less than 1.4 observations. Detecting the change of the complete object, meaning the object was exchanged with another one, is clearly a more difficult task. Results show that we can detect such cases with an average accuracy of 88–93%. Lastly, due to our ability of roughly estimating object orientations, we can also detect change in the yaw rotation of an object. For this case, we achieved average accuracies of 97% and more. As of now, our framework can only detect changes of objects in place, meaning translational changes would need to be detected separately, for instance by utilizing global position information.

8. Conclusion

We have presented an information-theoretic action selection framework for object recognition (and object change detection) which is capable of sequential training and classification, view planning as well as online feature selection. The recognition framework has successfully been tested in simulation and on a custom-built quadcopter robot equipped with an RGB-D camera.

Our online feature selection method has been shown to drastically reduce the computation time by computing only informative features used for inference during runtime. Online feature selection, compared to offline estimation, allows for much greater flexibility during training as well as execution. It is especially useful for robotic tasks like ours, since there decisions are made online, which stands in contrast to estimating the best performing set of features a priori during training phase. Furthermore, our experiments show that smart view planning, when we are uncertain about our belief state, can help to increase the recognition accuracy substantially.

Through extensive evaluations of various methods, we have shown the strengths and weaknesses in cost and accuracy of information-theoretic, heuristic as well as random strategies for feature and viewpoint selection. In general, one can say that information-theoretic viewpoint selection outperforms random selection in recognition accuracy, but increases computation cost. The selection of informative viewpoints is important to increase the accuracy and, at the same time, keep the number of observations minimal. In particular, this becomes apparent

when using a simple heuristic like "motion along a star pattern". From an information point of view, moving to the opposite side yields a lot of new information. However, this information does not necessarily help us in distinguishing objects from each other in feature space—not every information is equally useful. Information-theoretic approaches allow for selecting the most informative actions in a principled way; they increase recognition accuracy and reduce operating cost.

Compared to state-of-the-art single-view object recognition methods, our multi-view object recognition approach, when used with simple features, achieves comparable results. Our approach is not primarily meant to replace single-view methods but rather to complement them. Put in other words, if a confident decision can be made about an object after only one observation, there will be no need for taking a second view. However, there will always be objects with inherent ambiguity, and it is in those cases in which the multi-view approach is brought to bear. Moreover, feature selection is useful regardless of the number of views and the object in question, since it reduces the dimensionality of the feature space and decreases the overall computation cost.

Our framework can also utilize more complex features, such as the kernel descriptors from [1]. Although these features are more expensive to compute, compared to the simple features, they result in an increase of recognition accuracy. An interesting trade-off can be seen here, the complex features have more expressiveness whereas the simple features are more cost-efficient to compute.

Acknowledgment

The authors would like to thank Jörg Müller for his help on the development of the quadcopter robot platform.

References

[1] L. Bo, X. Ren, D. Fox, Depth kernel descriptors for object recognition, in: Proc. of the IEEE/RSJ Int. Conference on Intelligent Robots and Systems, IEEE, 2011, pp. 821–826.
[2] D.G. Lowe, Distinctive image features from scale-invariant keypoints, Int. J. Comput. Vis. 60 (2) (2004) 91–110.
[3] N. Atanasov, B. Sankaran, J.L. Ny, G.J. Pappas, K. Daniilidis, Nonmyopic view planning for active object classification and pose estimation, IEEE Trans. Robot. 30 (5) (2014) 1078–1090. arXiv:1309.5401v1.
[4] J. Denzler, C.M. Brown, Information theoretic sensor data selection for active object recognition and state estimation, IEEE Trans. Pattern Anal. Mach. Intell. 24 (2) (2002) 145–157.
[5] M. Verleysen, F. Rossi, Advances in Feature Selection with Mutual Information, CoRR arXiv:0909.0635v1.
[6] C. Potthast, G.S. Sukhatme, A probabilistic framework for next best view estimation in a cluttered environment, J. Vis. Commun. Image Represent. 25 (1) (2014) 148–164.
[7] R. Bajcsy, Active perception, Proc. IEEE 76 (8) (1988) 966–1005.
[8] C. Laporte, T. Arbel, Efficient discriminant viewpoint selection for active bayesian recognition, Int. J. Comput. Vis. 68 (3) (2006) 267–287.
[9] G.A. Hollinger, U. Mitra, G.S. Sukhatme, Active Classification: Theory and Application to Underwater Inspection, CoRR abs/1106.5.
[10] K. Tarabanis, R.Y. Tsai, A. Kaul, Computing occlusion-free viewpoints, IEEE Trans. Pattern Anal. Mach. Intell. 18 (3) (1996) 279–292.
[11] H. Gonzalez-Banos, J.-C. Latombe, Planning robot motions for range-image acquisition and automatic 3d model construction, in: Proc. of the AAAI Fall Symposium Series, 1998.
[12] C. Stachniss, G. Grisetti, W. Burgard, Information gain-based exploration using rao-blackwellized particle filters, in: Robotics: Science and Systems, vol. I, 2005, pp. 65–72.
[13] M. Krainin, B. Curless, D. Fox, Autonomous generation of complete 3d object models using next best view manipulation planning, in: Proc. of the IEEE Int. Conference on Robotics and Automation, 2011, pp. 5031–5037.
[14] G.A. Hollinger, B. Englot, F.S. Hover, U. Mitra, G.S. Sukhatme, Active planning for underwater inspection and the benefit of adaptivity, Int. J. Robot. Res. 32 (1) (2013) 3–18.
[15] S. Ekvall, P. Jensfelt, D. Kragic, Integrating active mobile robot object recognition and SLAM in natural environments, in: Proc. of the IEEE/RSJ Int. Conference on Intelligent Robots and Systems, 2006, pp. 5792–5797. http://dx.doi.org/10.1109/IROS.2006.282389.
[16] R. Finman, T. Whelan, M. Kaess, J.J. Leonard, Toward lifelong object segmentation from change detection in dense rgb-d maps, in: Proc. of the European Conf. on Mobile Robots, 2013, pp. 178–185.
[17] M. Gupta, J. Muller, G. Sukhatme, Using manipulation primitives for object sorting in cluttered environments, IEEE Trans. Autom. Sci. Eng. 12 (2) (2015) 608–614.
[18] D. Holz, M. Nieuwenhuisen, D. Droeschel, J. Stückler, A. Berner, J. Li, R. Klein, S. Behnke, Active recognition and manipulation for mobile robot bin picking, in: Gearing Up and Accelerating Cross-fertilization between Academic and Industrial Robotics Research in Europe, in: Springer Tracts in Advanced Robotics, vol. 94, 2014, pp. 133–153.
[19] K. Welke, J. Issac, D. Schiebener, T. Asfour, R. Dillmann, Autonomous acquisition of visual multi-view object representations for object recognition on a humanoid robot, in: Proc. of the IEEE Int. Conference on Robotics and Automation, 2010, pp. 2012–2019.
[20] A. Collet, S. Srinivasa, Efficient multi-view object recognition and full pose estimation, in: Proc. of the IEEE Int. Conf. on Robotics and Automation, 2010, pp. 2050–2055.
[21] K. Lai, L. Bo, X. Ren, D. Fox, A large-scale hierarchical multi-view rgb-d object dataset, in: Proc. of the IEEE Int. Conference on Robotics and Automation, IEEE, 2011, pp. 1817–1824.
[22] H. Borotschnig, L. Paletta, M. Prantl, A. Pinz, Active object recognition in parametric eigenspace, in: Proc. of the British Machine Vision Conference, 1998, pp. 63.1–63.10.
[23] L. Paletta, A. Pinz, Active object recognition by view integration and reinforcement learning, Robot. Auton. Syst. 31 (1) (2000) 71–86.
[24] M. Brown, D.G. Lowe, Unsupervised 3d object recognition and reconstruction in unordered datasets, in: Proc. of the Int. Conference on 3-D Digital Imaging and Modeling, 2005, pp. 56–63.
[25] M. Luber, L. Spinello, K.O. Arras, People tracking in RGB-D data with on-line boosted target models, in: Proc. of the IEEE/RSJ Int. Conference on Intelligent Robots and Systems, 2011, pp. 3844–3849.
[26] H. Grabner, H. Bischof, On-line boosting and vision, in: Proc. of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 1, 2006, pp. 260–267.
[27] I. Guyon, A. Elisseeff, An introduction to variable and feature selection, J. Mach. Learn. Res. 3 (2003) 1157–1182.
[28] H. Peng, F. Long, C. Ding, Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy, IEEE Trans. Pattern Anal. Mach. Intell. 27 (8) (2005) 1226–1238.
[29] H. Ali, Z.-C. Marton, Evaluation of feature selection and model training strategies for object category recognition, in: Proc. of the IEEE/RSJ Int. Conference on Intelligent Robots and Systems, 2014, pp. 5036–5042.
[30] T.G. Dietterich, Ensemble methods in machine learning, in: Proc. of the First Int. Workshop on Multiple Classifier Systems, Springer-Verlag, 2000, pp. 1–15.
[31] F. Schimbinschi, L. Schomaker, M. Wiering, Ensemble methods for robust 3d face recognition using commodity depth sensors, in: IEEE Symposium Series on Computational Intelligence, 2015, pp. 180–187.
[32] Z.-C. Marton, F. Seidel, F. Balint-Benczedi, M. Beetz, Ensembles of strong learners for multi-cue classification, Pattern Recognit. Lett. 34 (7) (2013) 754–761.
[33] X.S. Zhou, D. Comaniciu, A. Krishnan, Conditional feature sensitivity: a unifying view on active recognition and feature selection, in: Proc. of the IEEE Int. Conference on Computer Vision, vol. 2, 2003, pp. 1502–1509.
[34] C. Potthast, A. Breitenmoser, F. Sha, G.S. Sukhatme, Active multi-view object recognition and online feature selection, in: Proc. of the International Symposium on Robotics Research, Sestri Levante, Italy, 2015.
[35] L. Bo, X. Ren, D. Fox, Unsupervised feature learning for rgb-d based object recognition, in: Experimental Robotics — The 13th Int. Symp. on Exp. Robotics, 2012, pp. 387–402.
[36] R.B. Rusu, G. Bradski, R. Thibaux, J. Hsu, Fast 3d recognition and pose using the viewpoint feature histogram, in: Proc. of the IEEE/RSJ Int. Conference on Intelligent Robots and Systems, IEEE/RSJ, 2010.
[37] L. Bo, X. Ren, D. Fod, Kernel descriptors for visual recognition, in: Advances in Neural Information Processing Systems (NIPS), 2010.

Christian Potthast is a Ph.D. candidate at the University of Southern California in the Robotics Embedded Systems Laboratory working with Prof. Gaurav S. Sukhatme. Prior to USC, he was working with Prof. Frank Dellaert at Georgia Tech. He received a Diplom in Computer Science from the Technical University of Munich in 2006. During his time at TUM, he worked as a research assistant with Prof. Michael Beetz. Before that, he received a pre-diploma from the University of Freiburg.

Andreas Breitenmoser is a postdoctoral research associate in the Robotic Embedded Systems Laboratory of Prof. Gaurav S. Sukhatme at the Department of Computer Science, University of Southern California. He received his Master of Science in Electrical Engineering and Information Technology in 2008 and his Doctor of Sciences under the supervision of Prof. Roland Siegwart in 2013, both from ETH Zurich. During his Ph.D. studies, he was involved in collaborations with the ALSTOM company, Disney Research and the Distributed Robotics Laboratory at CSAIL, MIT.

Fei Sha has been a faculty at USC since the August of 2008. Prior to joining USC, he was a postdoctoral researcher at U. of California (Berkeley) with Prof. Michael I. Jordan and Prof. Stuart Russell. He also spend a year at Yahoo! Research as a research scientist. He has a B.Sc. and a M.Sc. degree in biomedical engineering from Southeast University (Nanjing, China), and a Ph.D. in Computer and Information Sciences from the University of Pennsylvania, under the supervision of Prof. Lawrence K. Saul (now at UC San Diego). Prof. Sha was selected as a Sloan Research Fellow and was also awarded an Army Research Office Young Investigator Award in 2012 and was a member of 2010 DARPA Computer Science Study Panel.

Gaurav S. Sukhatme is a Professor of Computer Science (joint appointment in Electrical Engineering) at the University of Southern California (USC). He received his undergraduate education from IIT Bombay in Computer Science and Engineering, and M.S. and Ph.D. degrees in Computer Science from USC. He is the co-director of the USC Robotics Research Laboratory and the director of the USC Robotic Embedded Systems Laboratory which he founded in 2000. His research interests are in multi-robot systems and sensor/actuator networks. He has published extensively in these and related areas. Sukhatme has served as PI on numerous NSF, DARPA and NASA grants. He is a Co-PI on the Center for Embedded Networked Sensing (CENS), an NSF Science and Technology Center. He is a fellow of the IEEE and a recipient of the NSF CAREER award and the Okawa foundation research award. He is one of the founders of the Robotics: Science and Systems conference. He was program chair of the 2008 IEEE International Conference on Robotics and Automation and is program chair of the 2011 IEEE/RSJ International Conference on Robots and Systems. He is the Editor-in-Chief of Autonomous Robots and has served as Associate Editor of the IEEE Transactions on Robotics and Automation, the IEEE Transactions on Mobile Computing, and on the editorial board of IEEE Pervasive Computing.