Bayesian Active Object Recognition via
Gaussian Process Regression

Marco F. Huber
AGT Group (R&D) GmbH
Darmstadt, Germany
marco.huber@ieee.org

Tobias Dencker, Masoud Roschani, Jürgen Beyerer
Institute for Anthropomatics
Karlsruhe Institute of Technology (KIT), Germany
{tobias.dencker|masoud.roschani|juergen.beyerer}@kit.edu

Abstract—This paper is concerned with a Bayesian approach of actively selecting camera parameters in order to recognize a given object from a finite set of object classes. Gaussian process regression is applied to learn the likelihood of image features given the object classes and camera parameters. In doing so, the object recognition task can be treated as Bayesian state estimation problem. For improving the recognition accuracy and speed, the selection of appropriate camera parameters is formulated as a sequential optimization problem. Mutual information is considered as optimization criterion, which aims at maximizing the information from camera observations or equivalently at minimizing the uncertainty of the state estimate.

I. INTRODUCTION

Research on computer vision mostly focuses on the object or scene observed by the camera system. It is assumed that the parameters of the camera (e.g., position, illumination, or focus) are given or determined off-line in a time-consuming trial-and-error process involving human interaction. Particular operations are then applied on the acquired images in order to solve the considered vision task like recognizing an object. In such passive vision systems, the camera parameters are not adapted on-line. This is in contrast to an active vision system, where the next camera observation is carefully planned based on the previously acquired images and prior information about the considered scene.

While various approaches for passive object recognition exist (see e.g. [1] and references therein), active object recognition still is in its early stages. One of the first approaches to active object recognition can be found in [2], where the object models are learned via the eigenspace approach introduced in [3]. The planning algorithm greedily chooses the view that leads to the maximum entropy reduction of the object hypotheses. In [4], from a finite set of views the one maximizing the mutual information between observations and classes is selected. The approach is designed for arbitrary features, but requires approximate mutual information calculation via Monte Carlo sampling, which prevents a direct extension to continuous views. An upper bound of the Jeffrey divergence is employed in [5]. Again, merely a finite set of viewpoints is considered. Reinforcement learning approaches for active object recognition are proposed in [6], [7]. Here, learning the object models and planning is performed simultaneously. A comparison of some of the aforementioned approaches can be found in [8].

The active object recognition method proposed in this paper consists of two parts (see Fig. 1). In the off-line learning part described in Section IV-A, for each object a so-called object model is created. For varying camera parameters, e.g., focus or position, 2D images of each 3D object are generated. Gaussian process regression is then applied on the sample images to learn the object models. As explained in Section III, Gaussian processes can be considered distributions over functions and thus, allow capturing the variations in images due to noise and errors in image pre-processing.

In the on-line recognition part, planning the next-best camera view (see Section IV-C) and Bayesian state estimation (see Section IV-B) are performed alternately. For planning, mutual information is maximized with respect to the camera parameters. Mutual information quantifies the reduction of the uncertainty in the current object estimate given a particular camera parameter. Based on the chosen parameter, the object estimate is updated via Bayesian estimation under consideration of the learned object models.

In contrast to prior art, the proposed method is very general as it is not restricted to specific image features. Furthermore, camera parameters can be arbitrary and continuous valued. All derivations in this paper regarding Bayesian estimation hold for arbitrary Gaussian process kernel functions. The performance of the proposed approach is demonstrated by means of simulations in Section V.

II. PROBLEM FORMULATION

In this paper, the object recognition problem is treated in a probabilistic fashion in order to account for uncertainties arising for example from camera noise, occlusion, or feature extraction. Based on a feature vector \( \mathbf{z}_k \in \mathcal{Z} \subseteq \mathbb{R}^{n_z} \), acquired from images at stage \( k = 0, 1, \ldots \), the goal is to estimate the true latent object class \( x \in \mathcal{X} = \{x_1, x_2, \ldots, x_N\} \subseteq \mathbb{N} \), with \( N \) being the finite number of possible object classes. For estimation purposes, the true object class is approximated by a discrete random variable \( x_k \in \mathcal{A} \), which forms the object class estimate. By means of the camera parameters \( a_k \in \mathcal{A} \subseteq \mathbb{R}^{n_a} \) the estimation process can be actively driven. Potential camera parameters are position, orientation, focal length, or exposure time, just to name a few.

The object class estimate \( x_k \) given all features and camera parameters up to and including stage \( k \) is characterized via...
the probability distribution \( p_{k|k} := p(x_k|z_{0:k}, a_{0:k}) \), with \( z_{0:k} = (z_0, z_1, \ldots, z_k) \). It is calculated recursively by means of Bayes’ equation [9] according to

\[
p_{k|k} = \frac{1}{c} \cdot p(z_k|x_k, a_k) \cdot p_{k|k-1} ,
\]

with normalization constant \( c := p(z_k|x_{0:k-1}, a_{0:k}) \) and \( p_{k|k-1} := p(x_k|z_{0:k-1}, a_{0:k-1}) = p_{k-1|k-1} \) being the distribution at stage \( k-1 \) typically denoted as prior distribution. The recursion (1) commences from \( p_0 := p(x_0) \) being the prior distribution of the object class estimate at stage \( k = 0 \). Furthermore, \( p(z_k|x_k, a_k) \) in (1) is the likelihood defined by the nonlinear transformation

\[
\hat{z}_k = h(x_k, a_k) + v_k .
\]

This measurement model with nonlinear measurement function \( h(.) \) relates the object class to a feature vector given the camera parameters. Here, the measurement noise \( v_k \) subsumes all uncertainties arising during image acquisition.

So far, the action \( a_k \) was assumed to be given. But in active object recognition, an action is chosen automatically by the imaging system itself for acquiring high informative observations. For this purpose, the optimization problem

\[
\hat{a}_k^* = \arg \max \limits_{a_k} I(x_k, \hat{z}_k|a_k)
\]

is formulated to determine the optimal action \( \hat{a}_k^* \) to be applied at stage \( k \). Since solving (3) results in the camera parameters to be applied next, it is often referred to as next-best-view planning (see e.g. [10]). As target function in (3), the mutual information \( I(x_k, \hat{z}_k|a_k) \) between state and observation given an action is considered. This measure quantifies the amount of information the knowledge of an observation reveals about the state and vice versa. It is closely related to Shannon’s entropy and zero only iff both variables are independent [11].

To solve the next-best-view problem given by (3), several problems arise: 1) Analytical expressions for the measurement model (2) and the likelihood \( p(z_k|x_k, a_k) \), respectively, are not given in general as both describe a complex transformation of a potentially high-dimensional feature vector to an abstract object class. 2) Calculating \( p_{k|k} \) in (1) cannot be performed in closed form for arbitrary likelihoods and priors \( p_{k|k-1} \) [9]. 3) Evaluating mutual information is only possible for some special cases, e.g., if \( x_k \) and \( \hat{z}_k \) are normally distributed. 4) The optimization problem is non-convex and thus, getting trapped in a sub-optimal solution becomes an issue. A novel active object recognition method addressing these problems is described in the following sections.

III. Gaussian Process Regression

To tackle the issue of not having analytic expressions of the measurement model and the likelihood, a machine learning tool named Gaussian processes (GPs) is employed. GPs allow non-parametric learning of regression functions from noisy training data. They can be considered Gaussian posterior distributions over functions conditioned on the training data [12]. Thus and in contrast to classical regression approaches, GPs provide not only a regression function but also provide uncertainty estimates (error bars) depending on the noise and the variability of the data.

For GP regression, it is assumed that a set of training data \( \mathcal{D} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\} \) is drawn from the noisy process

\[
y_i = h(x_i) + \epsilon ,
\]

where \( x_i \) are the training inputs, \( y_i \) are the training outputs, and \( \epsilon \sim \mathcal{N}(0, \sigma^2) \) is zero-mean Gaussian noise with variance \( \sigma^2 \). For brevity reasons, \( \mathbf{X} = [x_1, \ldots, x_n] \) are all training inputs and \( \mathbf{y} = [y_1, \ldots, y_n]^T \) are the corresponding training outputs in the following.

The GP is used to infer the latent function \( h(.) \) from the data \( \mathcal{D} \) and is completely specified by a mean function \( m(.) \) and a positive semi-definite covariance function \( k(.,.) \), also called a kernel. Throughout this paper, a zero mean function and the squared exponential (SE) kernel

\[
k(x, x') = \alpha^2 \cdot \exp \left( -\frac{1}{2} (x - x')^T \Lambda^{-1} (x - x') \right)
\]

are used, where \( \Lambda \) is a diagonal matrix of the characteristic length-scales for each input dimension and \( \alpha^2 \) is the variance of the latent function \( h \). It is worth mentioning that the active object recognition approach proposed in this paper is not restricted to an SE kernel. All derivations presented in the following hold for arbitrary kernels.

The posterior distribution of the function value \( h_\ast = h(\alpha) \) for an arbitrary test input \( \alpha \), is Gaussian with mean

\[
\hat{h}(\alpha_\ast) = E\{h_\ast\} = \mathbf{k}_\ast^T (\mathbf{K} + \sigma^2\mathbf{I})^{-1} \mathbf{y} ,
\]

and variance

\[
\sigma^2_\ast(h_\ast) = \text{var}\{h_\ast\} = k_{\ast\ast} - \mathbf{k}_\ast^T (\mathbf{K} + \sigma^2\mathbf{I})^{-1} \mathbf{k}_\ast ,
\]

with \( E\{\} \) being the expectation value, \( \text{var}\{\} \) being the variance, \( \mathbf{k}_\ast := k(\mathbf{X}, \alpha) \), \( k_{\ast\ast} := k(\alpha_\ast, \alpha_\ast) \), and \( \mathbf{K} \) being the kernel matrix with elements \( K_{ij} = k(x_i, x_j) \). Note that the variance depends on the noise \( \epsilon \) as well as on the correlation between test input and training data.
The parameters \( \sigma, \alpha, \Lambda \) of a GP are called the hyperparameters, which are learned automatically by maximizing the log-likelihood of the training data using numerical optimization [12]. Learning the hyperparameters corresponds to selecting a GP model describing the training data and thus, the process (4) adequately.

IV. ACTIVE OBJECT RECOGNITION

The GP regression introduced in the previous section forms the basis of the proposed active object recognition approach. All components necessary for object recognition using GP regression are described in the following. For an overview and an illustration of the interactions between the components see Fig. 1.

A. Learning Object Models

To apply GP regression, it is necessary to map the considered measurement model (2) to the latent process (4). It is obvious that in (4) merely one-dimensional outputs are considered. In object recognition however, multi-dimensional outputs resulting from feature extraction are typical. The straightforward way used in this paper to apply GP regression to the multi-dimensional case is to learn a separate GP for each output dimension \( e = 1, \ldots, n_z \). Thus, \( n_z \) GPs are learned independently using the same training inputs \( X \) but different training outputs \( \bar{z}^e = [z_1^e, \ldots, z_n^e]^T \) for each output dimension \( e \). In doing so, it is assumed that any two output dimensions are conditionally independent given the input.

For a deterministic input—here the deterministic action \( \bar{a} \)—this results in a posterior Gaussian with diagonal covariance matrix. For an uncertain input however, the covariance matrix is no longer diagonal [13]. An alternative approach resulting in non-diagonal covariance matrices even for deterministic inputs is the recently developed multi-output GP regression (see for example [14]).

Furthermore, learning the GPs for each output dimension has to be performed independently for each object class \( z \), \( l = 1, \ldots, N \). This results in \( N \) multi-variate GPs \( \bar{h}_l(\cdot) \sim \mathcal{GP} \) of dimension \( n_z \) named object models in the following. To learn an object model \( \bar{h}_l \), samples \( \bar{a}_l, i = 1, \ldots, n \) of the action space \( \mathcal{A} \) are used as training inputs \( X \). For each input sample \( \bar{a}_l \), an object of the class \( x_l \in \mathcal{X} \) is observed by the camera resulting in the feature vector \( \bar{z}_l = [z_1^l, z_2^l, \ldots, z_n^l]^T \) acting as training output. In total, for \( n_z \) output dimensions and \( N \) object classes, \( n_z \times N \) GPs are learned. Since learning these measurement models is an off-line task (see Fig. 1), the required computation time is independent of the computation time for object recognition. Furthermore, for high-dimensional features, which may be obtained for instance by means of the scale-invariant feature transform (SIFT, [15]), dimensionality reduction techniques like principal component analysis [16] or GP latent variable models [17] can be employed in order to reduce the number of GPs to be learned.

B. Bayesian Estimation

Given the learned object models, the next component towards an active object recognition is the estimation of the object class given an arbitrary but fixed action \( \bar{a}_k \in \mathcal{A} \). Determining the next-best action is content of Section IV-C. To solve Bayes’ equation (1), it is at first necessary to provide the representations of all involved distributions.

1) Prior Distribution: As the latent object class \( x \) is a discrete random variable, the prior distribution \( p_k|k-1 \) at stage \( k \) can be characterized by means of

\[
p_k|k-1 = \sum_{i=1}^{N} \omega_{k-1,i} \cdot \delta_{x_k,i} ,
\]

where the weight \( \omega_{k-1,i} \) represents the probability that object \( x \) belongs to class \( i \). The weights are non-negative and sum up to one. Further, \( \delta_{x_k,i} \) is defined as

\[
\delta_{x_k,i} = \begin{cases} 
1, & \text{if } x_k = i \\
0, & \text{otherwise}
\end{cases}
\]

and known as the Kronecker delta.

2) Likelihood: In case of a given object class \( x_k = i \), the likelihood \( p(\bar{z}_k| x_k = i, \bar{a}_k) \) corresponds to the GP \( \bar{h}_i(.) \). If in addition the action \( \bar{a}_k \) is given, the likelihood becomes a Gaussian density \( \mathcal{N}(\bar{z}_k; \hat{\bar{z}}_{k,i}, \mathbf{C}_{k,i}^z) \) with mean vector and covariance matrix according to

\[
\hat{\bar{z}}_{k,i} = \begin{bmatrix} z_{k,i}^1, z_{k,i}^2, \ldots, z_{k,i}^{n_z} \end{bmatrix}^T,
\]

\[
\mathbf{C}_{k,i}^z = \text{diag}(\sigma_{1,i}^2, \sigma_{2,i}^2, \ldots, \sigma_{n_z,i}^2)
\]

(9)

respectively. The elements in (9) corresponding to dimension \( e = 1, \ldots, n_z \) are calculated according to (5) and (6), respectively, with the given action \( \bar{a}_k \) being the test input and \( \bar{z}^e \) being the training output vector. Overall, the likelihood for a fixed action \( \bar{a}_k \) can be characterized by means of the hybrid conditional distribution

\[
p(\bar{z}_k|x_k, \bar{a}_k) = \sum_{i=1}^{N} \delta_{x_k,i} \cdot \mathcal{N}(\bar{z}_k; \hat{\bar{z}}_{k,i}, \mathbf{C}_{k,i}^z) .
\]

(10)

It is important to note that for a fixed observation \( \bar{z}_k \)—as required for solving Bayes’ equation—the conditional distribution in (10) becomes a weighted sum of Kronecker deltas as in (7), because all Gaussian components are evaluated at \( \bar{z}_k \) and thus, become scalar weighting coefficients.

3) Normalization Constant: Finally, the normalization constant \( c \) in (1) can be calculated by marginalizing the product of prior and likelihood over \( x_k \), which results in

\[
c = p(\bar{z}_k| \bar{z}_{0:k-1}, \bar{a}_{0:k}) = \sum_{x_k} p(\bar{z}_k; \bar{z}_{0:k-1}, \bar{a}_{0:k})
\]

\[
= \sum_{x_k} \left( \sum_{i=1}^{N} \omega_{k-1,i} \cdot \delta_{x_k,i} \cdot \mathcal{N}(\bar{z}_k; \hat{\bar{z}}_{k,i}, \mathbf{C}_{k,i}^z) \right)
\]

\[
= \sum_{i=1}^{N} \omega_{k-1,i} \cdot \mathcal{N}(\bar{z}_k; \hat{\bar{z}}_{k,i}, \mathbf{C}_{k,i}^z) .
\]

(11)

Thus, the normalization constant is a Gaussian mixture evaluated at the given observation \( \bar{z}_k \).
4) Posterior Distribution: With the closed-form representations of all required distributions at hand, it is now possible to solve Bayes’ equation resulting in the posterior distribution of \( x_k \)

\[
p_{k|k} = \frac{1}{c} \cdot \left( \sum_{i=1}^{N} \delta_{x_k, i} \cdot N(\hat{z}_k; \hat{z}_{k,i}, C_{k,i}^z) \right) = \frac{1}{c} \cdot \sum_{i=1}^{N} \omega_{k-1,i} \cdot \delta_{x_k, i} = \sum_{i=1}^{N} \omega_{k,i} \cdot \delta_{x_k, i}
\]

with weights \( \omega_{k,i} := \frac{1}{c} \cdot \omega_{k-1,i} \cdot N(\hat{z}_k; \hat{z}_{k,i}, C_{k,i}^z) \). As expected, the incorporation of a new observation \( \hat{z}_k \) leads to an adaption of the prior probability \( \omega_{k-1,i} \) of each object class \( i \) depending on the individual likelihood \( N(\hat{z}_k; \hat{z}_{k,i}, C_{k,i}^z) \) of the object class.

C. Next-Best-View Planning

The final component in Fig. 1 is the planning of the next-best-view and optimal action \( a_k^* \in A \), respectively, allowing for fast and accurate object recognition. As discussed in Section II, the optimal action results from solving the optimization problem (3), where the mutual information

\[
I(x_k, z_k | a_k) = H(x_k) - H(x_k | z_k, a_k) = H(z_k | a_k) - H(z_k | x_k, a_k)
\]

is employed for quantifying the utility of a particular action \( a_k \in A \). In (12) and (13), the first term \( H(.) \) denotes Shannon’s entropy

\[
H(x) = - \sum_x p(x) \cdot \log p(x)
\]

for discrete random variables and the differential entropy

\[
H(x) = - \int_x p(x) \cdot \log p(x) \, dx
\]

for continuous random variables, respectively (see [11]). The second term \( H(.,.) \) denotes the conditional entropy given by

\[
H(z|x) = - \int_x p(x) \int_z p(z|x) \cdot \log p(z|x) \, dz \, dx
\]

for continuous random variables \( x \) and \( z \). By replacing the integrals with sums, a similar expression for the conditional entropy can be found for discrete random variables.

1) Evaluation of Mutual Information: Unfortunately, neither (12) nor (13) allow an analytical calculation of the mutual information value. An approximate evaluation of the mutual information based on (12), however, is inappropriate for many reasons. While the first term \( H(x_k) \) is straightforward to evaluate as it is Shannon’s entropy (14) of the discrete prior distribution \( p_{k|k-1} \), the second conditional entropy term can only be evaluated approximately by discretizing the Gaussian mixture distribution \( p(z_k | z_{0:k-1}, a_{0:k}) \), e.g., by means of random sampling or the unscented transform [18]. Depending on the number of samples used, this approach of approximating mutual information becomes computationally demanding. For each sample, Bayes’ equation has to be evaluated completely in order to provide the posterior distribution \( p_{k|k} \) required for the inner integral in (15). Furthermore, random sampling precludes classical optimization techniques like gradient descent for solving the optimization problem (3).

Directly approximating mutual information via (13) is also critical, but (13) allows calculating a lower bound, which is very convenient for the maximization in (3). Here, the first term needs special treatment as it requires the calculation of the entropy of the Gaussian mixture (11), which is not possible in closed form in general due to the logarithm of a sum of exponential functions. Fortunately, the entropy of a Gaussian mixture can be bounded from below according to [19]

\[
H(\hat{z}_k | a_k) = - \sum_{z} p(\hat{z}_k | a_k) \cdot \log p(\hat{z}_k | a_k) \, d\hat{z}_k
\]

\[
= - \sum_{i=1}^{N} \omega_{k-1,i} \cdot \int_{z} \sum_{j=1}^{i} N(\hat{z}_k; \hat{z}_{k,j}, C_{k,j}^z) \, d\hat{z}_k
\]

\[
\geq - \sum_{i=1}^{N} \omega_{k-1,i} \cdot \int_{z} \log \left( \sum_{j=1}^{i} N(\hat{z}_k; \hat{z}_{k,j}, C_{k,j}^z) \right) \, d\hat{z}_k
\]

with shorthand term \( p(\hat{z}_k | a_k) := p(\hat{z}_k | z_{0:k-1}, a_{0:k}) \) and

\[
c_{ij} = N(\hat{z}_k; \hat{z}_{k,j}, C_{k,j}^z) + N(\hat{z}_k; \hat{z}_{k,j}, C_{k,j}^z)
\]

being the value resulting from integrating over the product of the two Gaussians \( N(\hat{z}_k; \hat{z}_{k,i}, C_{k,i}^z) \) and \( N(\hat{z}_k; \hat{z}_{k,j}, C_{k,j}^z) \). The lower bound follows directly from applying Jensen’s inequality [11], which allows pulling the logarithm out of the integral. With regard to complexity, the lower bound scales quadratically with the number of object classes \( N \) and thus is computationally very efficient as the number of classes is expected to be a few tens.

Utilizing the sifting property of the Kronecker delta (8) and the analytical evaluation of the entropy of a Gaussian distribution, the second conditional entropy term in (13) can written as

\[
H(\hat{z}_k | a_k) = - \sum_{i=1}^{N} \omega_{k-1,i} \cdot \int_{z} p(\hat{z}_k | x_k, a_k) \cdot \log p(\hat{z}_k | x_k, a_k) \, d\hat{z}_k
\]

where \( |.| \) is the determinant of a matrix. Putting (16) and (17)
of (13) is used in (3) to approximate the mutual information value.

2) Solving the Optimization Problem: Solving the optimization problem (3) for finding the optimal action or next-best-view $a^*_k \in \mathcal{A}$ for the current stage $k$ requires to calculate the maximum of the mutual information and its lower bound (18), respectively. Unfortunately, the optimal action cannot be calculated in closed form. Additionally, the maximum of the mutual information with respect to the actions $a_k$ is not unique and thus, the optimization problem is non-convex, which further complicates numerical optimization.

To increase the probability of finding the optimal action or at least to ensure finding an action that is very close to the optimal one, so-called multi-start optimization is performed (see e.g. [20]). Here, optimization is repeated from varying initial points. To cover the action space $\mathcal{A}$ uniformly, the initial points form a regular grid on $\mathcal{A}$. For each initial point, the lower bound (18) is maximized by means of the BFGS method [21]. This well known quasi-Newton numerical optimization technique utilizes—in contrast to a classical gradient ascent—an estimate of the Hessian matrix, which results in an increased converge speed towards the sub-optimal solution. The derivation of the required gradient with respect to $a_k$ can be found in Appendix A.

V. Simulation Results

The effectiveness of the proposed active object recognition approach is now demonstrated by means of numerical simulations. At first, the setup of all simulations is described. Then, two different object sets are considered for comparison.

A. General Simulation Setup

The considered objects are synthetic 3D models rendered by means of the Visualization Toolkit (VTK)$^2$. In Fig. 2, for each set some of the objects are depicted. For learning and recognition, $100 \times 100$ pixel normalized grayscale images are generated from these object, where zero-mean Gaussian noise with variance 14.7 is added.

1D and 2D features are extracted from the images. In the 1D case, the mean gray value is considered. The eigenspace or principal component decomposition approach proposed in [3] is used for extracting 2D features, where the two largest eigenvalues are taken into account. It is important to note that although low-dimensional features are considered here for simplicity, the proposed approach has been derived without any restrictions on the features. Thus, even very complex and high-dimensional features like SIFT can be employed as well.

The simulations focus on actions that change the camera position in one or two dimensions. In the 1D case, the camera moves on a circle that is parallel to the horizontal plane and centered at the object. In the 2D case, the camera position can be varied on a sphere centered at the object. Here, the actions correspond to the azimuth and elevation angles.

To learn the GPs, each dimension of the action space is sampled regularly in $10$ decimal degree steps, i.e., for the one-dimensional circular action space, this leads to $36$ sample images.

For comparison, the following active object recognition approaches are considered:

- **Planner** The proposed approach, where 5 and 15 initial points for optimization are exploited for the 1D and 2D action space, respectively.
- **Grid** An approach similar to [4], where at each stage the action maximizing the mutual information is taken from a finite set. Here, this finite set coincides with the set of initial points of the Planner.
- **Random** Actions are selected uniformly at random.

All approaches merely differ in the way the next action is selected, while for instance the same GP object models are used and the Bayesian update step is performed identically. Furthermore, Planner and Grid utilize the lower bound (18) of the mutual information.

For each set of objects and each combination of feature and action space, $50$ Monte Carlo simulation runs are performed, where the true object is selected uniformly at random. The initial distribution $p_0$ is uniform. A decision about the object type is made if either the probability of one object estimate exceeds 0.95 or after eight stages.

B. Example I: Cups

The first set of objects consists of eight cups that are identical except for the label that is cut through the surface (see Fig. 2). The labels of six cups are visible from the same perspective, one is visible from the opposite point of view and one cup is not labeled at all.

For the 2D action space, the mutual information surface for three cups is plotted in Fig. 3 (a). Here, the optimal action is indicated by the red circle, which corresponds to an elevation angle of approximately $45^\circ$. For this action, the corresponding views on the three cups are depicted in Fig. 3 (b)–(d). It can

---

$^2$http://www.vtk.org/
be seen that this view facilitates to look inside the cups and thus, allows an easy discrimination of all three cups.

The average values over the 50 simulation runs in terms of recognition rate, number of views, and maximum object probability are listed in Table I. It can be seen that the Planner performs best with respect to almost any performance indicator. In comparison to Random, the number of stages after which a recognition decision is made is significantly lower. Simultaneously, the certainty in this decision is much higher as the average maximum object probability indicates. The performance of the Grid approach is often close to the proposed approach. But the significantly lower number of views of the Planner shows the benefits of performing a continuous optimization for next-best-view planning. In contrast to both Grid and Random, the proposed Planner can take advantage of an increasing feature and action dimension, i.e., with an increasing dimension the recognition rate increases as well and the number of views decreases.

A high object probability not necessarily coincides with the best recognition rate as seen in the case of the 1D action space and 2D feature space. While Random merely relies on the GP object models for inference, Grid and Planner additionally use the models for decision making. Thus, a bootstrapping effect can cause the decision maker to get stuck in a repetitive pattern. The quality of the GP models is essential for the recognition process and thus, under- and over-fitting require special attention.

C. Example II: Toy Manikins

The second set of objects used for simulation consists of nine toy manikins that carry different pieces of equipment (bow, quiver, sword, emblem, helmet, and crest—see Fig. 2).

TABLE I
CUP RECOGNITION. (A) RECOGNITION RATE IN PERCENT, (B) AVERAGE NUMBER OF VIEWS, (C) AVERAGE MAXIMUM OBJECT PROBABILITY.

<table>
<thead>
<tr>
<th>Dim. A / Z</th>
<th>Planner (a)</th>
<th>Planner (b)</th>
<th>Planner (c)</th>
<th>Grid (a)</th>
<th>Grid (b)</th>
<th>Grid (c)</th>
<th>Random (a)</th>
<th>Random (b)</th>
<th>Random (c)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 / 1</td>
<td>66 0.06</td>
<td>62 0.74</td>
<td>50 0.32</td>
<td>72 0.52</td>
<td>94 0.81</td>
<td>68 0.74</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 / 2</td>
<td>88 3.08</td>
<td>74 0.97</td>
<td>76 0.72</td>
<td>68 0.81</td>
<td>76 0.72</td>
<td>68 0.81</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 / 1</td>
<td>92 2.5</td>
<td>62 0.95</td>
<td>76 0.72</td>
<td>68 0.81</td>
<td>76 0.72</td>
<td>68 0.81</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 / 2</td>
<td>100 1.88</td>
<td>88 2.50</td>
<td>68 0.97</td>
<td>68 0.81</td>
<td>76 0.72</td>
<td>68 0.97</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Compared to the cups, the toy manikins have much more details and the differences between each object are more subtle. In Fig. 4, the decision making of the Planner is shown for the 2D action space and 1D feature space. The first view reveals most of the equipment items in such a way that the differences to other manikins are significant regarding the rather simple mean gray value feature. The next two views highlight the sword as well as the crest and thus, help to distinguish the manikin from those without these items.

In Table II, the same performance indicators as in the cup scenario are listed. While Planner and Random perform nearly identical for the 1D action and feature space, for higher dimensions, the Planner clearly is the best object recognition algorithm. Interestingly, all algorithms perform better than in the cup scenario. This is mainly due to the more details of the manikins and thus, much more views exist that allow discriminating different manikins from each other.

VI. CONCLUSION AND FUTURE WORK

The proposed approach exploits Gaussian process regression for object recognition. Thanks to the probabilistic nature of the GPs, the variability in image acquisition—resulting for instance from changing light conditions, occlusion, or changing background—is incorporated and robust object models over continuous action spaces are generated from few training samples. In combination with recursive Bayesian estimation and optimizing the next view, this approach allows a reliable recognition even with low dimensional and thus, rather simple image features. The proposed approach can be applied in various recognition scenarios as it is not restricted to specific features, action spaces, or kernel functions.

Future work is devoted to applying the proposed approach in a real-world experiment, where a camera is mounted on a six degree-of-freedom robotic arm. By this means, the camera can be moved either in 2D or 3D space, as it is done in the simulations. So far, a recognition or classification problem has been considered. It is also intended to combine classification with pose estimation, i.e., to simultaneously identify the object.
class as well as its orientation and location in space. An improved recognition rate is expected—especially in situations with for instance time or kinematical constraints [22]—if actions are planned is a non-myopic fashion, i.e., for more than one stage ahead. Furthermore, learning and planning are currently decoupled. By means of reinforcement learning techniques [23], both steps could be performed simultaneously.

REFERENCES


<table>
<thead>
<tr>
<th>stage</th>
<th>x1</th>
<th>x2</th>
<th>x3</th>
<th>x4</th>
<th>x5</th>
<th>x6</th>
<th>x7</th>
<th>x8</th>
<th>x9</th>
<th>H(x)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>.111</td>
<td>.111</td>
<td>.111</td>
<td>.111</td>
<td>.111</td>
<td>.111</td>
<td>.111</td>
<td>.111</td>
<td>.111</td>
<td>1.0</td>
</tr>
<tr>
<td>1</td>
<td>.149</td>
<td>.383</td>
<td>.000</td>
<td>.000</td>
<td>.013</td>
<td>.000</td>
<td>.259</td>
<td>.188</td>
<td>.009</td>
<td>.643</td>
</tr>
<tr>
<td>2</td>
<td>.000</td>
<td>.000</td>
<td>.000</td>
<td>.000</td>
<td>.000</td>
<td>.322</td>
<td>.674</td>
<td>.004</td>
<td>.298</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>.000</td>
<td>.000</td>
<td>.000</td>
<td>.000</td>
<td>.000</td>
<td>.000</td>
<td>.000</td>
<td>.998</td>
<td>.001</td>
<td>.006</td>
</tr>
</tbody>
</table>

Fig. 4. Recognition of object x8 via proposed approach: selected views (left) and corresponding distributions pkl(k with entropy (right).
\[
\frac{\partial \gamma_{ij}}{\partial a} = \frac{\partial [2\pi C_{ij}]^{-\frac{1}{2}}}{\partial a} \cdot g_{ij} + [2\pi C_{ij}]^{-\frac{1}{2}} \cdot \frac{\partial g_{ij}}{\partial a} \\
= -\frac{g_{ij}}{2} [2\pi C_{ij}]^{-\frac{1}{2}} \text{Tr} \left( C_{ij}^{-1} \cdot \frac{\partial C_{ij}}{\partial a} \right) - \frac{1}{2} \frac{c_{ij}}{2} \left( 2 \left( \frac{\partial \tilde{z}_{ij}}{\partial a} \right)^T \cdot C_{ij}^{-1} \cdot \tilde{z}_{ij} - \tilde{z}_{ij}^T \cdot C_{ij}^{-1} \cdot \frac{\partial C_{ij}}{\partial a} \cdot C_{ij}^{-1} \cdot \tilde{z}_{ij} \right)
\]

(24)

with \( g_{ij} := \exp \left( -\frac{1}{2} \cdot \tilde{z}_{ij}^T \cdot C_{ij}^{-1} \cdot \tilde{z}_{ij} \right) \) and \( \tilde{z}_{ij} := \tilde{z}_i - \tilde{z}_j \).

Applying (21) and (22) on (23) gives the result in (24).

The remaining derivatives \( \frac{\partial \gamma_{ij}}{\partial a} \) and \( \frac{\partial C_{ij}}{\partial a} \) can easily be decomposed into the derivatives of the respective summands. Furthermore, calculating the derivatives can be performed dimension-wise. Thus, the remaining partial derivatives \( \frac{\partial \gamma_{e}}{\partial a_{i}} \) and \( \frac{\partial (\sigma^2)^e}{\partial a_{i}} \) for each dimension \( e = 1, \ldots, n \), correspond to the derivatives

\[
\frac{\partial \gamma}{\partial a} = \left( \frac{\partial k}{\partial a} \right)_e \left( K + \sigma^2 I \right)^{-1} y,
\]

\[
\frac{\partial \sigma^2}{\partial a} = \frac{\partial k}{\partial a} \cdot -2 \left( \frac{\partial \gamma}{\partial a} \right)_e \left( K + \sigma^2 I \right)^{-1} \gamma,
\]

of (5) and (6) with respect to \( a \), respectively, with the matrix

\[
\frac{\partial k_e}{\partial a} = \left[ \frac{\partial k}{\partial a} k(a_1, a), \ldots, \frac{\partial k}{\partial a} k(a_n, a) \right].
\]

(25)

Here, \( a_1, \ldots, a_n \) are the training inputs. The derivative of the SE kernel in (25) for \( i = 1, \ldots, n \) is given by

\[
\frac{\partial k(a_i, a)}{\partial a} = \alpha^2 \cdot \Lambda^{-1} \cdot (a_i - a) \cdot \exp \left( -\frac{1}{2} (a_i - a)^T \Lambda^{-1} (a_i - a) \right).
\]