A Feature Gradient based Reinforcement Learning Approach to Interactive Segmentation

Bharath Sankaran, Mrinal Kalakrishnan, Jeannette Bohg, Stefan Schaal

Abstract— **For robots to be successful in human environments they need to have the ability to intelligently interact with their environment. Intelligent interaction entails two fundamental problems, one of optimal action selection and the other includes online model adaptation. In this paper we introduce a feature gradient based reinforcement learning approach to address problems in the interactive perception domain. We specifically deal with the problem of clutter segmentation as it provides a good test case to apply this logic. We introduce an approach that learns the an optimal action selection strategy offline based on supervised max entropy learning and adapts this learned model online using a gradient optimization based reinforcement learning approach, where the gradients are optimized in the feature space. We demonstrates the utility of our approach by performing several 100s of experiments on the robot where the robot had no prior knowledge of the environment. We also compared our approach against other state of the art approaches to compare performance**

I. Introduction

Personal robots need to have the ability to intelligently interact with their environments to be able to successfully operate in human environments. In this paper we focus on a specific class of intelligent interaction; namely the ability to pick the most optimal action given any perceptual input and adapt this action selection model to any variation in inputs online. Most autonomous systems engaging in forceful interaction with the environment are tasked with this fundamental problem of action selection, i.e given the current perceptual input what is the most optimal action to select to achieve some given objective. When the objective of such tasks is strictly perceptual the tast of action selection can be cast into either of one of the two classes of problems:

1) Active Perception : Agent's actions does not change the physical state of the environment

2) Interactive Perception : Agent's actions can change the physical state of the environment

The problem of active perception for visual input has been one that has been long studied in the computer vision community since the early 80s. The multiple flavors of active perception such as sensor management, next best view, PTZ-control deal with the fundamental problem of how to control the sensor to achieve some perceptual objective. In previous work the problem of active perception for object detection was addressed by Sankaran and Atanasov ([1]: TODO: Fix citiation to ICRA '13). In the current work we look at addressing the problem of interactive perception, specifically the problem of clutter segmentation.

The problem of clutter segmentation is an interactive perception problem as every action the agent selects to declutter the environment changes the physical state of the environment. In this domain of clutter segmentation, we assume no prior knowledge of any objects in the environment. The action selection in our approach is restricted to a set of movement primitives that can be executed at any location in the environment. These movement primitives can includes actions like grasping, pushing, pulling etc.

Fig. 1. Robot Operating in Clutter: TODO: Put in a better picture

II. Related Work

The related work needs to be written from scratch. The stuff written below is from another previous paper and is being used as a place holder. The approaches in sensor management [2], [3] can be classified according to sensor type into *mobile* (sensors have dynamic states) and *stationary* (sensors have fixed states). Also, the targets of interest might be mobile or stationary. The process of choosing sensor configurations may quantify the utility of the next configuration only (*myopic*) or may optimize over a sequence of future sensor configurations (*non-myopic*). Finally, the objective may be to identify a target and estimate its state or simply improve the state estimate of a detected target.

The earliest work in active perception can be attributed to Bajscy [4], [5]. It was focused on 3D position estimation

This work was supported by the

B. Sankaran, M. Kalakrishnan and S.Schaal are with the Computational Learning and Motor Control Lab, University of Southern California, Los Angeles, CA 90089, bsankara@usc.edu, kalakris@usc.edu

J.Bohg and S.Schaal are with the Autonomous Motion Department, Max Planck Institute for Intelligent Systems, Tuebingen Germany, jbohg@mpi-is.de,sschaal@mpi-is.de

through control of the sensor's intrinsic parameters. Pito's 1999 paper [6] addresses the next best view problem as one that maximizes information gain by increasing spatial resolution. The movement of the sensor is constrained to a circle centered around the object of interest.

The work that is closest to ours [7] uses a mobile sensor to classify stationary objects on a table and estimate their poses. Static detection is performed using SIFT matching. An object's pose distribution is represented with a Gaussian mixture. The authors use a myopic strategy to reduce the differential entropy in the pose and class distributions. This work differs from ours in that the sensor has models of all the objects so the detection is never against background. Moreover, by formulating hypotheses about an object's identity and by choosing a small discrete space for the possible sensor poses, we are able to plan non-myopically.

Velez and coworkers [8], [9] consider the problem of detecting doorways, while a mobile sensor is traveling towards a fixed goal point. The unknown state of a candidate detection is binary: "present" or "not present". Stereo disparity and plane fitting are used for pose estimation. An entropy field is computed empirically for all view-points in the workspace and is used to myopically select locations with high expected information gain. The authors assume that the static detector provides sufficiently accurate pose estimates and do not optimize them during the planning.

In our work we use a depth sensor, which validates the assumption that the position estimate of a stationary object is accurate and does not need to included in the optimization objective. However, the orientation estimates can be improved through active planning. Inspired by the work on hypothesis testing [10], we introduce a rough discretization of the space of orientations so that the hidden object state takes on several values, one for "object not present" and the rest for "object present" with a specific orientation. As a post-processing step, the rough orientation estimate is used to seed a robust alignment procedure, which provides an accurate pose estimate. In our previous work we considered a dual hypothesis problem aimed at model completion [11].

Karasev et al. [12] plan the path of a mobile sensor for visual search of an object in an otherwise known and static scene. The problem statement is different from ours but the optimization is surprisingly similar. The authors hypothesize about the pose of the object and minimize the probability of an incorrect decision. Since different object locations need to be considered, the optimization is intractable. Instead, a mathematical model of the sensing process is used to maximize the conditional entropy of the next measurement.

A lot of the work in sensor management assumes a fixed sensor position, which simplifies the problem considerably because the trade-off between minimizing movement energy and maximizing view-point informativeness is avoided [13], [14]. Often, the action selection process is myopic. In contrast, we consider a mobile sensor, include the detection process in the optimization, and use non-myopic planning. Golovin and Krause [15] showed that myopic planning for an adaptively submodular objective function is merely by a constant factor worse than the optimal strategy. Unfortunately, the objective in our formulation is not adaptively submodular and even with a fixed sensor state, a myopic strategy can perform arbitrarily worse than the optimal policy [10].

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

III. Problem Formulation

Consider a set of rigid objects piled-up on any surface denoted $T_1, ..., T_n \in \mathcal{T}$, where n is the number of objects. This set of rigid objects can be represented by a graph $\mathcal{G}$ where each vertex $g_i \in \mathcal{G}$ corresponds to a node with unique appearance. Here appearance can be defined by a number different attributes combined into a feature vector.

Every object in our set $\mathcal{T}$ can be represented by a rigid clique in this graph $\mathcal{G}$ comprising of one or more interconnected nodes g_i. Edges between rigid cliques are dynamic edges which are removed when two rigid objects are separated. Hence each rigid object corresponds to a set of nodes $\{g_{i1}, ..g_{in}\}$ in a clique $\mathcal{C}_i$. A collection of such cliques with

with edges between them represents our scene graph denoted by $\mathcal{G}$. We cast the clutter segmentation problem as a graph separation problem where our objective is to pick a set of minimal actions to remove the dynamic edges from the graph and separate the graph into a set of minimal cliques each of which represent a rigid body.

In the initialization phase the dynamic edges between cliques represent the rigid objects that are either touching each other or entagled with each other in a pile of objects.

Given an initial scene graph $\mathcal{G}$ our objective is to pick the most optimal sequence of actions from a discrete set of actions $\{a_1, ...a_m\} \in \mathcal{A}$ which optimizes the objective of graph separation and reduces the number of actions required to be executed. As the problem of trying to identify the set of actions that optimizes

the graph separation objective for the set of all possible graphs is intractable (given all possible set of graphs that one might encouter in natural scenes). To circumvent this problem we try to learn a mapping from actions $\mathcal{A}$ to features $\mathcal{F}$. These features $\mathcal{F}$ represent the current state of the environment. Instead of mapping actions to graphs we constraint the problem by trying

to learn a mapping between actions $\mathcal{A}$ and features $\mathcal{F}$. The quality of the features can be evaluated by the reward observed after executing each action. This learning problem can be cast as a supervised learning problem to classify the features based on action labels. Our learning approach discussed in detail Section V. Since we need to label features, we use Learning from

Demonstration (LfD) to execute actions and label them with user specified rewards. We use a Max Entropy learner as the dimensionalty of the feature vector we use is much larger compared to the number of samples we collect.

As the dimensionalty of the feature space is incredibly high, we may still not be able to capture the entire variance in the feature space. To account for this variation, we let the learner adapt online to the features. Hence we utilize an online policy gradient styled approach where we optimized parametrized actions with respect to an expected reward by gradient descent. The details of the approach are dicussed in Section V.

IV. LEARNING FROM DEMONSTRATION

In our approach to clutter segmentation, since the learner/robot is not capable of exploring the entire space of all possible adjacency matrices for graph separation, we constraint the problem by exploring in the feature space rather than the space of all possible graphs. To accomplish this we define a set of features over the scene graph and execute actions on them

and observe rewards. Each action is demonstrated by an expert who also labels the actions as good and bad based on the observed reward. So once an expert demonstrates and action, the action is executed on a randomly selected scene graph and the reward is observed. After observing a sequence of such rewards and the corresponding features computed for that specific scene graph,

we feed the features and their corresponding labels (1/0) into a Max Entropy learner to learn the weights for each action. These weights are then used in the online phase to select appropriate actions for a set of observed features. In the online phase the weights are also updated using a policy gradient approach discussed in Section V.

A. Feature Selection

To learn the utility of actions for a given scene graph $\mathcal{G}$, we define a set of features $\mathcal{F}$ over the scene graph. The features we use are color histograms, textons, grasp templates ([1] TODO: Cite Alex here), push-shape templates and entropy maps. The computation of these features are discussed in detail in the following subsections

1) Color Histogram: To compute a color histogram over a given image patch we convert the patch from rgb to hsv color space and quantize the hue space into 3 bins and saturation space into 6 bins. The number of bins was selected heuristically to satisfy computational efficiency and sparsity constraints. So for each hue bin there are 6 corresponding saturation bins. Hence we quantize the hue-saturation values into 18 bins.

2) Texton Features: The idea of textons was first introduced by Olshausen and Field as fundamental microstructures in generic natural images ([1] TODO: Fix Citation). We utilize the version first introduced by Leung and Malik in ([1] TODO: Fix Citation). Here we convolve the image patch with a series of gabor filters at six evenly spaced orientations at two different pyramid scales.

Once the patch is the image patch is convolved with this series of gabor filters, we take each individual filter response of each image patch and cluster them in an unsupervised manner using k-means clustering. (In our implementation we use k=24 for computational efficiency reasons). Once the responses from the filters are clustered into visual words, we quantize the gabor filter responses corresponding to the computed

visual words using a nearest neighbour approach. This histogram of quantized responses now serves as a texton feature.

3) Entropy Map: To quantify the entropy of the local image patch we quantize the colors of the image into RGB histograms. These histograms are normalized and their frequency is computed by summing the bins in each individual channel. The entropy of the image is then computed by summing the shannon entropy of each bin in each channel.

This is computed as shown below:

$$Entropy = \sum_j p_j * log(p_j)$$

where p_j is given by (Bin Value)/(Frequency of Channel). Hence this gives us three values for entropy, one for each color channel.

4) Push Template: Here we introduce the notion of a push template. A push template is a region in the point cloud which is amenable to pushing by a manipulator. To define such a template we extract holes in images, which are recovered by projecting euclidean clusters on some nominal plane and looking for voids on this plane. Once these voids are detected,

we extract the largest pointcloud surrounding such a void in an image by looking at the total number of points inside an oriented bounding box. The orientations of the bounding box are determined by uniformly discretizing the space of all yaws. The size of the bounding box is constrained by the radius of the size of the gipper of the manipulator in use.

Of all the tested orientations the bounding box with the maximum number of point is selected. We then compute quantized local normals (Fast Point Feature Histograms [1] TODO: Fix Citation) for the pointcloud inside this bounding box and take the mean of the feature computed over all points.

5) Grasp Templates: As a final feature we compute grasp templates which are local heightmaps computed for a given sensor viewpoint and gripper pose. These features were first introduced by Herzhog et al in ([1] TODO: Fix Citation). The reader is encouraged to read the paper mentioned to learn about the computation of this specific feature.

Once these features are computed we concatenate these features in to a 112 dimensional feature vector to describe the current scene graph $\mathcal{G}$. Once we record this feature $\mathcal{F}$ for a given scene graph, then we execute one of the actions in our action set $\mathcal{A}$ and label the feature as 1-0 based on the observed reward.

B. Max Entropy Learning

After number of executions of each action we take the labeled features $\mathcal{F}_l$ and run them through a Max-Entropy Learner to learn weights on these features. We learn a set of weights w_i corresponding to each action $a_1, ...a_i \in \mathcal{A}$. We use L-1 logistic regression as our max entropy learner.

The objective function we try to optimize in our learner is given by:

$$J = \sum_{i=1}^{m} -log(\frac{1}{1+\exp^{w^T x_i y_i}}) + \lambda ||w||_1$$

This objective function is optimized by the limited memory version of the BFGS algorithm namely, L-BFGS. At test time the labels of a given feature can be determined by

$$y_{test} = sgn(w^T x_{test})$$

We use L-1 regularized logistic regression as opposed to L-2 regularized logistic regression as it performs better when the dimensionality of the features exceeds that of the number of available training samples. (TODO: Cite that paper that Mrinal cites in his work). We use these learned weigths in conjunction with the features observed online by sampling from a gibbs distribution as suggested by ([1] TODO: Fix Citation, Cite Peters and Bagnell)

$$\pi(a|s) \sim (\frac{\exp^{\phi(s,a)^T \theta}}{\sum_b \exp^{\phi(s,b)^T \theta}})$$

The action selection and the online adaptation of the weights is discussed in more detail in Section V.

V. Online Adaptation

The online phase of the algorithm we create a scene graph $\mathcal{G} = \{V, E\}$, this graph at a selected vertex and update the action parametrization by updating the feature weights by gradient descent in feature space. The online phase of the algorithm has three stages; a supervised classification phase, a data association phase and an online gradient update phase. The three phases are discussed in detail in the sections below.

A. Supervised Classification

When a scene is presented to the agent (robot) in the form of an rgb image and corresponding point cloud is used to construct a scene graph $\mathcal{G}$. We first pre-process the point cloud to look for euclidean clusters supported by some dominant plane in the image. Once these clusters have been detected we project them back on to the image to get the corresponding image patches.

To construct the scene graph we oversegment these image patches in to superpixels using the graph based segmentation approach described in ([1] TODO: Fix Citation). We then construct the scene graph by analyzing the nearest neighbours of each superpixel segment. Every vertex V_i of the graph $\mathcal{G}$ would be correspond to a superpixel and the edges E_j would be be drawn between superpixels touching each other. From such a respresentation the notion of rigid and dynamic edges are readily apparent as rigid edges are those edges that connect superpixels on the same object and dynamic edges are those edges that connect superpixels between two objects that are touching or in close proximity to each other. This is illustrated in the figure below Fig 2.

Fig. 2. Illustrates the scene graph construction from a set of objects on a table

Once we generate the scene graph $\mathcal{G}$, we compute the set of features described in Section IV for the vertex of maximum degree V_{max} in the graph. The intuition for only attempting to manipulate the Vertex of maximum degree is that it is connected the most number of other vertices hence manipulating this vertex can impact the graph adjacency matrix more severly than any other node.

Once we compute the features for the max degree vertex V_{max} we use the weights computed using the max entropy learner in the offline learning phase discussed in Section IV. Using these weights we select the best action to execute by sampling the best action from a gibbs distribution as shown below.

$$a_t = \max(\frac{\exp^{w^T f(x)}}{\sum_b \exp^{w^T f(x)}})$$

Once we generate the action to be executed we execute the action a_t and observe the corresponding reward R_t. Here t is the current time step and $f(x)$ is the feature function.

B. Data Association

Once the action is executed we track the vertices of the scene graph $\mathcal{G}_t$ using optical flow. The positions of the vertices of the graph are updated using the Gunnar Farnebaack

Optical Flow algorithm ([1] TODO: Cite Gunnars' ICPR 2000 paper) to obtain the predicted scene graph update $\tilde{\mathcal{G}}_{t+1}$. Once the vertices in the graph are updated to get the predicted graph $\tilde{\mathcal{G}}_{t+1}$, the predicted graph is matched with the observed scene graph $\hat{\mathcal{G}}_{t+1}$ to get the actual scene graph $\mathcal{G}_{t+1}$. The graph matching is accomplished by measuring the Bhattacharya distance given by

$$D_B(p,q) = -\log \sum_{x \in \mathcal{X}} \sqrt{p(x)q(x)}$$

between the appearance histograms of the vertices of each graph. Using this distance metric we are able to establish an accurate correspondence between $\tilde{\mathcal{G}}_{t+1}$ and $\hat{\mathcal{G}}_{t+1}$, to get $\mathcal{G}_{t+1}$. This correspondence matching is show below in Figure 3.

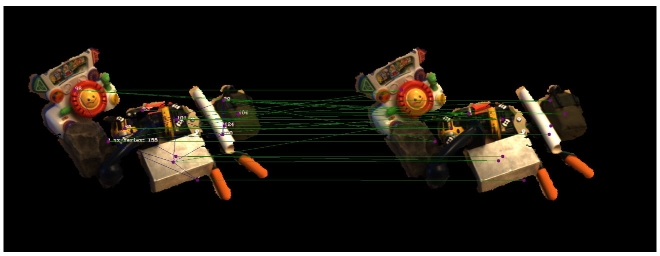

Fig. 3. Show's the result of the Graph Matching algorithm after the Optical Flow update

Once we get the updated scene graph $\mathcal{G}_{t+1}$ we perform an online policy gradient update utilizing the expected reward and the observed reward from the updated scene graph.

C. Online Feature Gradient Update

A policy gradient method is a **reinforcement learning** method that directly optimizes a parametrized control policy by gradient descent. Since we use discrete actions in our approach; as opposed to performing gradient descent in policy space we peform gradient descent in feature space, thereby adapting the action selection to the current scene being observed. This is accomplished in a method similar to that employed by policy gradient approaches.

Our feature gradient also follows the gradient of the expected return where the feature weights are updated as follows.

$$w_{k+1} = w_k + \alpha_k \bigtriangledown_w J(\pi_w)|_{w=w_k}$$

where $J(\pi_w)$ the gradient on the reward is computed using regular regression

$$\bigtriangledown_w = (\Delta W^T \Delta W)^{-1} \Delta W^T \Delta J$$

The mean-subtracted reward return ΔJ is obtained by subtracting the observed reward from the expected reward. The expected reward is given a function of the features weights computed using the Max Entropy learner in the offline phase. Hence the expected reward is $J(\hat{w}_k) = W^T f(x)$. The observed reward is expected reward weighted by the variation in spectral norm of the adjacency matrix. This is given by

$$J(\bar{w}_k) = \frac{1}{(\|A_{t+1}\| - \|A_t\|)} * \beta * J(\hat{w}_k)$$

where β is an appropriate scaling factor. The reason we use the spectral norm is because the spectral norm measures the magnitude of the largest singular value of a matrix. And to measure the similarity between two matrices we can compare their spectral norms. Hence using this logic the gradient update is performed as long as the observed adjaceny matrix and the actual adjacency matrix are different.

Once this update is performed a new action is sampled from the gibbs distribution parametrized by the updated feature weights. The actions are sampled until the variation in the spectral norm goes to zero or the feature weights top updating.

VI. Implementation

We evaluated our approach on the Barrett WAM robot with a WAM arm and a Barrett BH280 hand used and an end effector. The robot consists of a head-mounted Asus Xtion Sensor and a Bumblebee 2 camera. The entire head mount is calibrated with respect to the rest of the kinematic chain of the robot.

The motion planner used in conjuction with the movement primitives was presented by Kalakrishnan et al ([1] TODO: Cite Mrinal). The actual movement primitives are encoded by basic cartesian control and force control motions. The set of movement primitives used in our experiements are: Grasp, Push Forward, Pull Backward, Push Right, Push Left. The direction of the movement primitives is determined with respect to a fixed base axis. The pushing movement primitives are executed as basic cartesian control moves for a fixed threshold distance. The grasping movement primitive is encoded as an overhead grasp where the grasping motion is first position controlled and then force controlled.

The input to the feature computation pipeline are a RGB image from the Bumblebee 2 camera and a corresponding RGBD pointcloud from the ASUS Xtion Camera. All feature computations are parallelized in the implementation. In the offline phase the action selection is initialized by an expert who instructs the systems on what action to execute. Once the action is executed, the expert applies a binary label to the action, 1 if the action succeeded and zero otherwise. Once a series of actions have been executed, the features and their corresponding labels are collected. These labeled features are used to train a L1 regularized logistic regressor which is optimized using a commerically available LBFGs package (([1] TODO: Cite website)).

A. Online Phase

In the online phase the RGBD pointcloud is first preprocessed, i.e the dominant supporting plane is removed from the pointcloud and only the clusters corresponding to the points above the plane are extracted.

VII. Conclusion

References

[1] M. Naghshvar and T. Javidi, "Active sequential hypothesis testing," *IEEE Transactions on Information Theory (to appear)*, March 2012.

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

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

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

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

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

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

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

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

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

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

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

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

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

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