Reinforcement Learning for Semantic Segmentation in Indoor Scenes

Md. Alimoor Reza
Department of Computer Science
George Mason University
Fairfax, VA, USA
Email: mreza@gmu.edu

Jana Košecká
Department of Computer Science
George Mason University
Fairfax, VA, USA
Email: kosecka@gmu.edu

***Abstract*—Future advancements in robot autonomy and sophistication of robotics tasks rest on robust, efficient, and task-dependent semantic understanding of the environment. Semantic segmentation is the problem of simultaneous segmentation and categorization of a partition of sensory data. The majority of current approaches tackle this using multi-class segmentation and labeling in a Conditional Random Field (CRF) framework [18] or by generating multiple object hypotheses and combine them sequentially [3]. In practical settings, the subset of semantic labels that are needed depend on the task and particular scene, and labelling every single pixel is not always necessary. We pursue these observations in developing a more modular and flexible approach to multi-class parsing of RGBD data based on learning strategies for combining independent binary object-vs-background segmentations in place of the usual monolithic multi-label CRF approach. Parameters for the independent binary segmentation models can be learned very efficiently, and the combination strategy—learned using reinforcement learning—can be set independently and can vary over different tasks and environments. Accuracy is comparable to state-of-art methods on a subset of the NYUD-V2 dataset of indoor scenes [24] , while providing additional flexibility and modularity.**

I. Introduction

The problem of semantic understanding of indoors environments is central to many service robotic applications where objects need to be sought and recognized. The two most common strategies for tackling the problem of object detection/search is to either train an object category or instance specific detector or approach the problem as one of semantic segmentation where a partition of the sensory data is labeled with the semantic categories. The later approach can be more powerful as it enables exploitation of various contextual relationships between objects and background or region categories and also between objects themselves. This problem has been studied extensively by the Computer Vision community and evaluated on a variety of benchmark datasets [8, 12, 24].

Most current semantic segmentation approaches use either a multi-label Conditional Random Field (CRF), e.g. [18], or generate multiple object hypotheses and combine them sequentially, e.g. [3, 9, 10]. Building on the expressive power of these approaches and the availability of large amounts of labeled data, there have been notable recent improvements in accuracy [10]. One appealing aspect of these approaches is that, given enough training data, a system can be trained to perform well on a specific benchmark dataset for semantic segmentation.

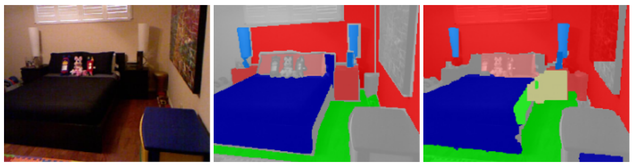
Fig. 1. From left to right, a sample RGB image, corresponding ground truth image, and the semantic segmentation output from our learned policy in bedroom scene.

The motivation of this paper is to revisit the problem setting from a methodological standpoint. First we observe that depending on the current task or scene, considering all possible labels may not be necessary, while focusing on only task-relevant labels may improve the accuracy of semantic segmentation that is relevant for the task. Second, as robotics considers wider goals including life-long learning and adaptation, it is less reasonable to assume a known fixed set of labels will be used for parsing. Instead there is a need to represent and acquire new semantic concepts in a modular and incremental manner while linking them together with additional concepts, e.g. as represented in a large graph by [21].

Both of these observations touch on one of the core advantages and potential difficulties of approaches that use a multi-label CRF to combine models of local image content based on a variety of features with measures of consistency between nearby labels. These approaches must optimize the trade-off between these factors with respect to all of the multiple classes. This works well when the set of labels is fixed, and the task is semantic segmentation with all those labels, but cannot be easily adapted to consider only a subset of categories relevant for a specific task or environment, or to accommodate a new category.

We propose an alternative approach for semantic understanding of indoors environments, combining sequential binary object/background segmentations in Reinforcement Learning framework. Central to this approach is that instead of training a multi-label Conditional Random Field (CRF), we propose to learn a number of independent binary object/background segmentations. We formulate the problem of multi-class semantic segmentation as a Markov Decision Process (MDP) by learning an optimal policy for composition of binary segmentations. We also propose a different superpixel representation, where large super-pixel regions are formed by robust geometric fitting of planar supporting surfaces and small compact superpixels are using appearance cues. The contributions are summarized below:

- Modular formulation of semantic segmentation as sequential composition of binary object/background segmentations learned in Reinforcement Learning framework.
- Novel super-pixel formation to accommodate both large planar surfaces and smaller objects.
- Competitive evaluation on subset of NYUD-V2 dataset.

This allows appealing modularity and enables the use of simpler training and inference of individual components.

II. Related Work

The presented work is related to different strategies for multi-class semantic segmentation, for object/background segmentation and context modeling for both RGB and RGB-D images. Here we focus on the methodologies used and discuss the suitability of different choices for building modular adaptive life-long learning robotic perceptual systems. Silberman et al. [23] introduced the NYUD-V1 dataset with complex indoor scenes for semantic segmentation, followed by NYUD-V2 dataset [24], with more object categories across a diverse set of scenes. Several strategies were proposed on these datasets ranging from multi-label CRFs, bottom-up segmentation followed with a classification step of the regions, or deep convolution neural network to bypass complex feature computation and segmentation [18, 9, 10, 7, 15].
In [18] the authors used superpixel hierarchies endowed by features extracted from the entire path of the segmentation tree. In later work, excellent results were achieved by considering bottom up unsupervised segmentation and boundary detection to generate and classify candidate regions [9, 10]. Additional strategies for generating and ranking and classifying multiple object proposals have been also explored by [6]. More recent approaches achieved superior results, by using deep convolutional networks for feature extraction and semantic segmentation [7, 15].
The problem of binary object/background classification has been also tackled with the commonly used sliding window processing pipelines, with window pruning strategies and features adapted to RGB-D data [17, 28]. More complex, discriminatively-trained part-based models, which incorporated some context information have been introduced recently [16]. The bounding boxes generated by the sliding windows approaches, while efficient, provide only poor segmentation and are often suitable for objects whose shape can be well approximated by a bounding box.

A body of work explored Reinforcement Learning techniques for robotics and image understanding problems [11, 13]. Karayev and colleagues [11] learn a policy that optimizes object detections in a time restricted manner. Kwok et al.[13] introduced an augmented MDP formulation for the task of picking the best sensing strategies in a robotics experiment and utilizes least square policy iteration to solve for the optimal policy.

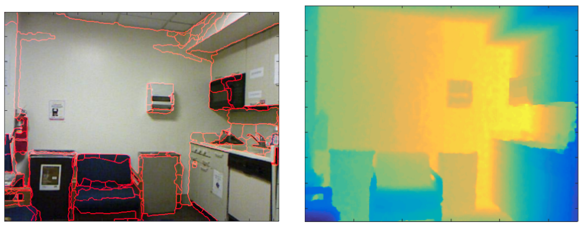
Fig. 2. Our superpixel image (bounded by red lines) and the depth image from NYUD V2. Notice how our superpixels capture the large planar surfaces.

III. Proposed Approach

We start by learning binary object/background segmentation models for each *object* category. While these do use a CRF to integrate appearance and label consistency cues, we only consider a single category and the background. The resulting object/background segmentations may overlap, and need to be combined into a single pixel-level labeling. The simplest strategy of taking the most confident prediction for the overlapping regions is not ideal as it requires calibrating the categories with respect to each other. This strategy is something we explicitly avoid as it often prefers large objects of interest which can be detected with high confidence compared to smaller objects. Instead, we adopt a sequential strategy for combining the binary segmentations, found using reinforcement learning. This approach is favorable compared to a fixed ordering, which naturally gives priority to large or frequent categories due their contribution to the overall pixel level accuracy performance measure and can dynamically adjust based on each image.

In the following sections we describe the ingredients of our approach starting with learning and inference for generating binary object/background segmentation, the sequential combination strategy and learning the optimal sequencing of object background segmentations in reinforcement learning framework.

A. Object-vs-background Segmentation

Motivated by the observation that indoor scenes contain many planar structures, we segment an image into superpixels of dominant planar and non-planar surfaces. Our framework starts by identifying the dominant planar surfaces using a RANSAC based approach by sampling 3D point triplets from the depth image [25]. The remaining regions not explained by the planar surfaces are represented by compact SLIC superpixels [1] generated from the RGB image. Figure 2 shows our superpixels, where the region boundaries are marked in red. We refer to these regions as superpixels in the following sections.

We formulate the binary object/background segmentation in a Conditional Random Field (CRF) framework. The conditional distribution $p(\mathbf{x}|\mathbf{z})$, is a log-linear combination of feature functions or potentials as shown in Equation 1. Final prediction for the hidden random variables is found as the *maximum a-posteriori (MAP)* solution $\mathbf{x}$ for which the conditional probability $p(\mathbf{x}|\mathbf{z})$ is maximum.

$$p(\mathbf{x}|\mathbf{z}) = \frac{1}{Z(\mathbf{z})} \exp(w_1 \sum_{i}^{V} \theta_d(x_i, \mathbf{z}) + w_2 \sum_{(i,j)}^{E} \theta_{ps}(x_i, x_j, \mathbf{z})) \tag{1}$$

The hidden random variables $\mathbf{x}$=$\{x_1, x_2, ..., x_{|V|}\}$ correspond to the nodes in the CRF graph, $\mathbf{z}$ refers to the vector of observation variables, and $Z(\mathbf{z})$ is the partition function. For our case, each hidden random variable can take one of two discrete values: x_i = $\{$*object, background*$\}$, where *object* $\in$ $\{$*bed, sofa, ... , wall*$\}$ in indoor scenes experiment. Note that the object here simply refers to one of the semantic labels available in the dataset. The *unary feature function* associated with each node, x_i, comes from the output probability $p(x_i|\mathbf{z})$ of a RUSBoost classifier.

$$\theta_d(x_i, \mathbf{z}) = -log(p(x_i|\mathbf{z})) \tag{2}$$

where the observations $\mathbf{z}$ are computed for each superpixel i using a subset of features described in Section III-A1. The *pairwise feature function* is computed for every edge $(j, k) \in E$ in the CRF graph. The graph is defined over all pair of neighboring pixels in a 4-connected grid structure of the CRF graph. We define our pairwise feature function with a constant Potts penalty for dissimilar labels of the adjacent pixels. With the choice of our unary and pairwise functions, the energy function corresponding to Equation 1 can be minimized exactly in polynomial time using GraphCut [4]. We perform our inference using publicly available efficient GraphCut optimization code of [2]. In our experiment, we used a fixed set values for our w_1 and w_2 in all of our experiments by finding them using a grid search. These values can instead be learned from a set of labeled training examples for each binary object-vs-background segmentation [20].

1) Features: We use a set of rich discriminative features to characterize the superpixels capturing both local appearance and geometry and global statistics of the scene. We exploit a subset of geometric and generic features proposed in [9], computed over superpixels. Appearance is encoded using the traditional representation of *color* and texture features introduced in [26]. The color distribution of a superpixel is represented by a fixed histogram of color in HSV color space and texture histogram is formed from a set of oriented Gaussian derivatives in a fixed set of orientations. The geometric features computed over these 3D points associated with each superpixel are described in detail in [5]. In total we compute 386 dimensional feature for each superpixel. For additional details, we refer to [19].

2) RUSBoost Classifier for Superpixels: Once we compute the features for our superpixels, we train RUSBoost classifier [22] in a one-vs-all fashion for each object class. RUSBoost classifier addresses the class imbalance problem of positive-negative instances to improve the classification performance of the learned model. We train a RUSBoost classifier for a object of interest from the training image set $I = \{I_1, I_2, ..., I_n\}$, where that object class is present. In addition to all the available negative instances in the training set I, we also include additional negative instances from other remaining training images in the entire training dataset. These additional important negative instances for training are sampled according to the distribution of *object*'s co-occurrence with other objects in the full training images. We select negative samples from a particular object class in proportion to the object's co-occurrence value in the distribution. We normalize the scores predicted by the learned RUSBoost classifier to find a probabilistic output for a superpixel s as $p(s|\mathbf{z})$, where $\mathbf{z}$ is the feature set computed for superpixel s. Finally, we assign the probability $p(s|\mathbf{z})$ to each constituent pixel x_i in superpixel s as follows:

$$p(x_i|\mathbf{z}) = p(s|\mathbf{z}) \tag{3}$$

where $x = \{x_1, x_2, ..., x_N\}$ is the set of pixels inside the superpixel s. The negative log value for each pixel is used as the unary term of our CRF Equation 2.

B. Sequential combination of binary segmentations

Traditional approaches for semantic segmentation look for all possible labels in the scene [27, 9]. They either approach the problem by training a multi-class CRF or one-vs-all classifiers for all semantic labels and computing the confidences for all the semantic labels. We next describe our sequential object selection mechanism formulated using reinforcement learning. The resulting policy dynamically selects a sequence of objects, based on the image content.

We generate the binary object/background segmentations for different objects in each scene and sequentially combine their predicted segmentation masks. To resolve the conflict between an overlapping region we utilize a heuristic that considers overlap ratios of two competing segments. For example, in the sequential combination process of the *bedroom* scene, object *bed* is considered first followed by object *pillow*. C is the overlapped region between segments B and A, which have different labels as shown in Figure 3. The regions are broken into the following $A = C + A'$ and $B = C + B'$, where A' and B' are the non-intersected portion of segments A and B respectively. Prior to considering *pillow* object in the sequential process, the overlapped part C was assigned the label C_{bed} along with A'_{bed}. The non-overlapping part of *pillow* has label B'_{pillow}. We consider the segment-size-ratio heuristic, which prefers small-sized segments that can take place on-top-of larger segments, e.g. *pillow* can lie on-top-of *bed*. We compute the ratios $\frac{C}{A}$, and $\frac{C}{B}$ then assign C to labels of the segment that has the larger ratio value. For an image, we sequentially combine obtained binary segmentations from three background classes in the order of *wall, floor*, and *ceiling* irrespective of the scenes we consider, then follow up with sequential selection of objects given by a policy we learn using reinforcement learning.

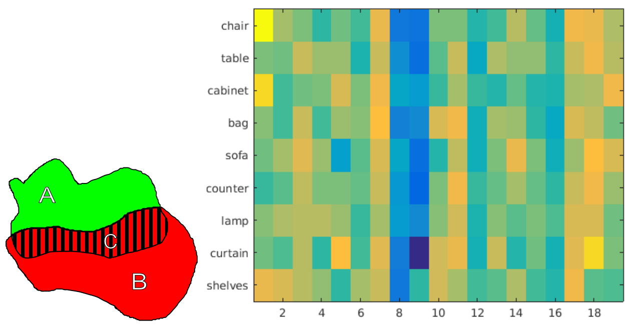

Fig. 3. a) Conflict resolution of intersected region between two competing object segments discussed in III-B. b) Learned weight vector (best viewed in color) $\mathbf{w}_\pi$ corresponds to our *LSPI* learned policy for the *dining room* the 3-Fold cross validation experiment. The yellowish entries denote positive values where as bluish entries denote negative values. y-axis shows the actions and each row denotes the weights for each action block.

1) Markov Decision Processes: In this section we will review the reinforcement learning framework based on Least Squares Policy Iteration (LSPI) introduced in [14]. The general framework of learning control policies (sequences of actions) based on experience and rewards is that of Markov Decision Processes (MDP). MDP models the agent acting in the environment as a tuple $\{S, A, T, R\}$, where S is a set of states, A finite set of actions, $T(s, a, s')$ is the transition model which describes the probability of of ending up in state s' after executing an action a in state s and $R(s, a, s')$ is a reward obtained in state s when agent executes action a and transitions to s'. The goal of solving MDP is to find a policy $\pi : S \rightarrow A$ for choosing actions in states such that cumulative future reward is maximized. If parameters T and R are known then optimal control policy can be computed efficiently using value iteration. In case the parameters of MDP are now known, the reinforcement learning can be used to learn the optimal policy. One set of RL techniques aims at learning the model parameters T and R and then proceeds with standard value iteration, the other *model-free* techniques learn the policy directly. For our problem we adopt the *model-free* approach and apply least squared policy iteration introduced in [14] and also used in [13] and [11] in the context of robotics and image understanding problems.

2) Model: In our case the state $s \in S$ consists of our current estimate of the distribution of the objects in the scene $P(B) = \{P(B_1), P(B_2), ..., P(B_K)\}$, where $P(B_i)$ is the probability of an object class-i is present in the image. Additionally, we also keep track of objects that are already taken into consideration into a observation variable O. A set of actions $a_i \in A$, where action a_i corresponds to the selection of object/background segmentation for object i in the sequential semantic segmentation. Our reward function R is defined in terms of pixel-wise frequency weighted Jaccard Index computed over the set of actions taken at any stage of an episode. *Jaccard Index* JI_i is the intersection-over-union ratio for the object i's predicted segmentation and the ground truth segmentation. In the training stage we consider each image as an episode. Length of an episode is the maximum number of actions in our action set A. Lets assume that in the current episode, starting with action a_1 we reached to position *k* of the episode by taking a_k. MDP state maintains actions taken so far in observation variable O. We compute the current reward $R(s, a_k, s')$ in state s at the episode as follows:

$$R(s, a_k, s') = w_1 * JI_{a_1} + w_2 * JI_{a_2} + ... + w_k * JI_{a_k} + r_{bg}$$

$$r_{bg} = w_{wall} * JI_{wall} + w_{floor} * JI_{ceiling} \tag{4}$$

w_i are the ratio of the pixels predicted by the segmented object i to the total pixels in the image. The r_{bg} is the *pixel-wise frequency weighted Jaccard Index* score of the three fixed background classes. Our dynamic policy finds the state-action value function $Q^\pi(s, a)$, which estimates a real-valued score to each state-action pair as defined by equation 5.

$$Q^\pi(s, a) = E[R(s, a, s^{'}) + \gamma Q^\pi(s^{'}, \pi(s^{'}))] \tag{5}$$

The policy is defined on the $Q^\pi(s, a)$ as defined by equation 6. Our state space is continuous and intractable, hence we can featurize the policy function with an approximation function with features $\psi(s, a)$ and a set of linear weights w_π^T.

$$\begin{aligned} \pi(s) &= \arg\max_a Q^\pi(s, a) \\ &= \arg\max_a w_\pi^T \psi(s, a) \end{aligned} \tag{6}$$

3) State Featurization of MDP: Our MDP is not fully observable hence we follow a pursuit of Augmented MDP [11, 13] which includes uncertainty variables into the state featurization. $\psi(s, a)$ is defined by following a block-featurization approach where features specific to each action are inserted into a specific block. All blocks are concatenated one after another to give us a fixed length feature vector. Each block in $\psi(s, a)$ has fixed order of entries as follows:

$$\begin{aligned} \psi(s, a) = \{P(B^a_{prior}), P(B_1|\mathbf{O}), ..., P(B_K|\mathbf{O}, \\ U(B_1|\mathbf{O}), ..., U(B_K|\mathbf{O})\}. \end{aligned} \tag{7}$$

Here $P(B^a_{prior})$ is the prior probability of the object a in the image. $P(B_i|\mathbf{O})$ are the object probabilities conditioned on the current set of observation. $U(B_i|\mathbf{O})$ are the uncertainties conditioned on the observations, which is computed as the Entropy measure from the $P(B_i|\mathbf{O})$. Each block is of size 1+2K, where K is the size of a single block feature as mentioned above. If size of the Actions set is $|A|$, then our featurization function $\psi(s, a)$ has size of $|A|$(1+2K). All but the selected action block is zero in this featurization approach.

4) Modeling the Featurization Components: The block-featurization of action a is computed as follows. The prior term $P(B^a_{prior})$ is computed from the MAP probability of the binary segmentation for object *a*. The observation terms update the belief-state each time an action is performed. The selection of object changes the belief about the presence or absence of the object in the image. In order to capture this information, we update the observation terms $P(B_i|\mathbf{O})$ for *i*-th object if action a_i is in the observation vector $\mathbf{O}$ by the following equation:

$$P(B_i|\mathbf{O}) = \max\{P(B_i|CC^1), ..., P(B_i|CC^M)\}. \quad (8)$$

Each CC^j term is the connected component from the segmentation mask of object i and $P(B_i|CC^j)$ is computed from the output of a RUSBoost [22] classifier. We train the classifier from the ground truth segmentation masks for each object from the training images. We used the same features as described in Section III-A1. $P(B_i|\mathbf{O})$ retained the value $P(B_i)$ if action a_i is not the observation vector $\mathbf{O}$.

5) Reinforcement Learning for Policy: The goal is to learn a policy, which maximizes the state-action value function, $Q^\pi(s,a)$ defined in Equation 5. $Q^\pi(s,a)$ is defined recursively by the expected cumulative future rewards. The discount factor defined by γ controls the amount of future rewards to be taken into consideration. We learn our policy using Least Square Policy Iteration (LSPI), a model-free Reinforcement Learning approach. Instead of directly estimating $Q^\pi(s,a)$ defined in Equation 5, the LSPI learns the linear functional approximation of $Q^\pi(s,a)$ as follows:

$$Q^\pi(s,a) \approx w_\pi^T \psi(s,a) \quad (9)$$

LSPI learns this functional approximation from the samples of MDP. Lets assume that our MDP currently is in state *s*. It receives a reward of *r* by selecting an action a and transitions into state s'. This event will generate a sample of the form (s,a,s',r) for the MDP. We generate such samples for each image until the end of an episode. We start in image as a new episode and generate such $|A|$ samples per episode. A collection of such samples define the sample set D from which linear functional approximation weight vector w_π is estimated. More specifically, lets assume that matrix $C \in \Re^{KxK}$ and vector $\mathbf{b} \in \Re^K$; both are initialized with matrix and vector of zeros respectively. Then C and b are updated for each consecutive samples of the form (s,a,s',r) from D using the following equations [14][13]:

$$C = C + \psi(s,a)(\psi(s,a) - \gamma\psi(s^{'}, \pi(s^{'}))^T \quad (10)$$

$$b = b + \psi(s,a)r \quad (11)$$

New policy for the next iteration is computed by the following:

$$w = C^{-1}b \quad (12)$$

We refer to the works of Lagoudakis et al. [14] for the detailed derivation. Starting with an initial policy, we generate a sample set D. The new policy is estimated using the Equations (10) and (11). This new policy is used to generate sample set for the next iteration of the policy. We follow ϵ-greedy action selection mechanism, starting with a large ϵ value that allows for sufficient space exploration of the policy during training, followed by reduction of ϵ values in successive iterations. Although LSPI can reuse the same sample set D generated during the first iteration, we find it more useful to use the sample sets generated in successive iterations to learn our policy. During the testing step of our policy, we set the ϵ value to be 0.005. We experimentally found that maximum of 10 iterations is sufficient for learning our policy. We used γ value of 0.3 for all our experiments. Figure 3b) shows the learned weights for the *dining room* experiment shown in Figure 5. The weight vector is reshaped according to action blocks in each row for better demonstration.

IV. Experiments

We conducted our experiments on the NYUD-V2 dataset [24]. For the definition of the scene category, we used the 9 most common scene categories as defined in [9]. These scenes contain multiple instances of objects and are well represented across all the scene categories provided in the original dataset. We use the standard train/test split of 795/654 images as used in [24, 9]. We run binary object/bacground segmentation Equation (1) for each of the object categories in different scenes to get binary object/background segmentations.

A. Policy Evaluation using Random 3-Fold Cross-Validation

We aim at learning a dynamic policy that can effectively give us an object selection mechanism learned from the data. Learned policy should select object based on the content of the image to maximize the reward, which we define to be *pixel wise frequency weighted Jaccard Index* 4 of the image. To test the effectiveness of our learned policy, we conducted experiment on the standard test split portion of NYUD-V2 dataset, which comprises 654 images of various scenes. Note that the 795 training images of the standard split was used to train the object specific RUSBoost classifier and to generate our binary object/background segmentation. We conducted a scene-specific experiment on the 9 scenes in NYUD-V2. In each scene we do a 3-fold cross validation images, training the policy using 2-fold images and test the policy on the third fold. Figure 5 compares our LSPI learned policy against three other baseline policies defined as follows:
i) *Fixed Order Policy.* In this policy mechanism, the order of selecting objects is fixed and is defined to be the frequency of object's occurrence in the scene-specific standard split of the training images [24]. For example, in *bedroom* scene the objects are considered in the following order $\{$*bed, pillow, lamp, night stand, dresser, box, clothes, chair, books*$\}$. There are 191 images in the standard test split of NYUD-V2. Similar statistics are computed for the remaining 8 scenes of *bookstore, classroom, dining room, homeoffce, kitchen, living-room, office*.
ii) *Random Order Policy.* An action is selected randomly based on the uniform distribution. The set of actions is again chosen to be the recognition of most-frequently occurring 9 objects as described in the *Fixed Order Policy*.
iii) *Oracle Policy.* In this policy an oracle tells the policy to pick from the objects that are already present in the current image. Again the set of actions are the most frequent 9 objects. The three baseline policies are colored *blue, red, cyan* respectively in the evaluation plots. Each curve demonstrates how each policy evolves as we take more actions into consideration. Figure 5 reveals that our policy does better than *Random Order*

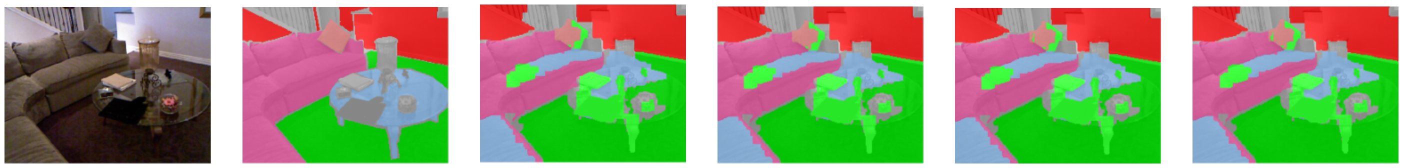

Fig. 4. Visualization of semantic segmentations from the *LSPI*-learned policy against the other baselines in *livingroom* scene. From left to right the images are *rgb, ground truth* following up with semantic segmentation output from *LSPI, Fixed order, Random order*, and *Oracle* policies respectively.

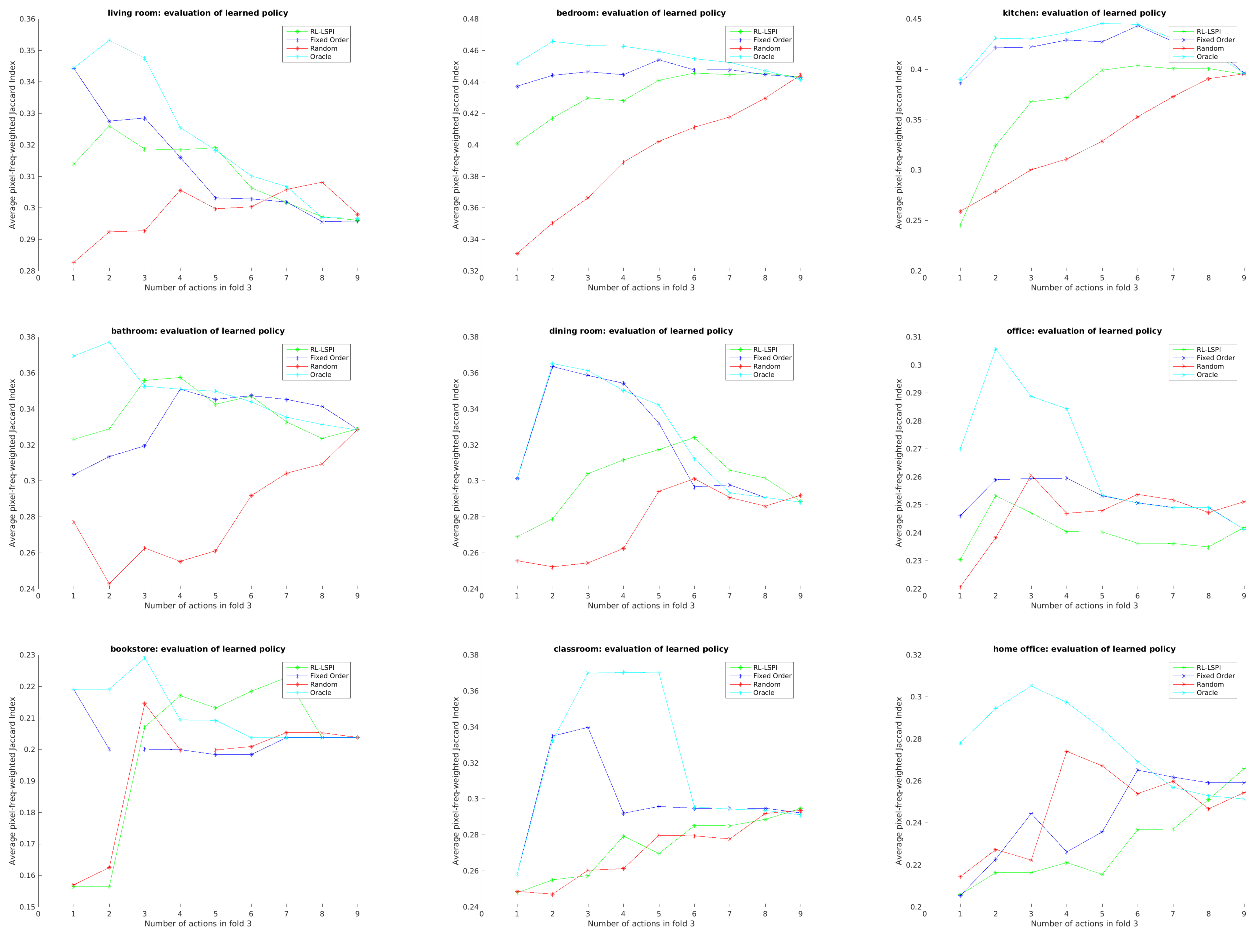

Fig. 5. Comparison of our **LSPI** learned policies against three baselines: **Fixed Order**, **Random Order**, **Oracle** from 9 different scenes in NYUD-V2 dataset (best viewed in color). Each plot shows number of actions taken into consideration in the horizontal axis. The vertical axis shows the average value of the reward metric *pixelwise frequency weighted Jaccard Index* on the test-fold images. Each curve demonstrates how the policy evolves as we take more actions into consideration. The first row shows comparisons in *living room, bedroom, kitchen* scenes respectively. These three scenes have the reasonably good number of images to train the policies from. Second row shows the comparisons on *bathroom, dining room, office* scenes respectively. These has moderate number of images to learn the policy. The last row shows the comparisons on *bookstore, classroom, home office* respectively with fewer images for policy learning.

Policy. It is comparable with the *Fixed Order Policy* in some of the 9 scene experiments. The *Oracle Policy* does better than other policies since it has the correct information about the presence or absence of objects in the image.

B. *Policy Evaluation on Controlled 3-Fold Cross-Validation*

In the 3-Fold cross validation experiment, frequently occurring objects may dominate the test-fold images where the policy is evaluated. Hence it may cause the *Fixed Order Policy* to perform better than others in our metric of comparison. We created a more controlled set of train/test-fold images to evaluate the learned policy. In this experiment, we consider the most-frequently occurring 9 objects in *living room* scene from the standard test split of NYUD-V2 dataset. These are *sofa, table, pillow, chair, lamp, cabinet, books, bookshelf* and *bag*. We create two control sets by deliberately choosing those images for our controlled train/test fold, where a pair of most-frequently occurring objects does not appear in every image of the folds. In *Set I*, the pair chosen is *pillow, chair* and in *Set II* the objects chosen are *table, chair*. A graphical comparison of our learned policy with the three baselines is shown in

Figure 6 for all the folds. Unlike previous experiment our learned policy performs similarly or better compared to the *Fixed Order Policy* in all the folds.

	cabinet	toilet	sink	counter	towel	bathtub	shower c.	bag	shelves	wall	floor	ceiling
Our	25.6	33.3	26.5	**55.6**	21.0	31.8	**16.2**	**0.3**	**6.4**	41.5	61.3	0
[9]	44.8	46.5	35.7	52	25.9	31.1	9.7	0	4.5	67.6	81.2	61.1
[10]	44.9	55.1	37.5	51.3	16.3	38.2	4.2	0.2	3.5	68	81.3	60.5

TABLE I
PERFORMANCE COMPARISON ON BATHROOM SCENE IN NYUD-V2 DATASET IN PIXELWISE PERCENTAGE JACCARD INDEX.

	bed	pillow	lamp	night s.	dresser	box	clothes	chair	books	wall	floor	ceiling
Our	55.9	26.9	15.5	10.0	28.5	**2.1**	6.9	16.6	**7.3**	61.0	**82.3**	19.6
[9]	57.0	30.3	16.3	21.5	24.3	2.1	4.7	36.7	5.5	67.6	81.2	61.1
[10]	60.5	34.4	34.8	27.2	34.8	2.1	7.4	47.9	6.4	68	81.3	60.5

TABLE II
PERFORMANCE COMPARISON ON BEDROOM SCENE IN NYUD-V2 DATASET IN PIXELWISE PERCENTAGE JACCARD INDEX.

	cabinet	counter	sink	chair	bag	fridge	table	towel	box	wall	floor	ceiling
Our	**46.1**	36.7	7.9	32.1	**2.1**	**25.9**	12.7	7.9	**2.9**	46.5	**86.9**	50.2
[9]	44.8	52	35.7	36.7	0	16.2	28	25.9	2.1	67.6	81.2	61.1
[10]	44.9	51.3	37.5	47.9	0.2	14.5	29.9	16.3	2.1	68	81.3	60.5

TABLE III
PERFORMANCE COMPARISON ON KITCHEN SCENES IN NYUD-V2.

	bookshf	shelves	books	clothes	box	cabinet	table	lamp	desk	wall	floor	ceiling
Our	**26.3**	**14.9**	**12.0**	1.8	**2.2**	0	4.9	0	0	30.3	78.4	18.3
[9]	19.5	4.5	5.5	7.4	2.1	44.8	28	16.3	7.1	67.6	81.2	61.1
[10]	18.1	3.5	6.4	4.7	2.1	44.9	29.9	34.8	11.3	68	81.3	60.5

TABLE IV
PERFORMANCE COMPARISON ON BOOKSTORE SCENE IN NYUD-V2.

C. *Comparison of Sequential against Optimal Combination*

To determine the best sequence of objects given our model, we evaluated the sequential order of object selection against the optimal combination. In order to find the optimal combination of objects order, we enumerate all permutations of K objects in consideration. For each permutation sequence, we sequentially combine the supervised segmentations of the objects and compute the reward. Out of all such permutations, we pick the one that gives us the maximum reward as the optimal combination. The number of permutation grows exponentially with the increase of number of total objects K in consideration. In this experiment, we picked 5 most frequently occurring objects in *living room* scene and find out the optimal combination by enumerating all 120 permutations in each image. We pick the as optimal combination the one

	chair	table	cabinet	desk	shelves	box	person	books	bag	wall	floor	ceiling
Our	27.9	8.2	11.9	**26.2**	**5.4**	0.9	1.2	0	0	35.0	64.4	**63.0**
[9]	36.7	28	44.8	7.1	4.5	2.1	5	5.5	0	67.6	81.2	61.1
[10]	47.9	29.9	44.9	11.3	3.5	2.1	0.2	6.4	0.2	68	81.3	60.5

TABLE V
PERFORMANCE COMPARISON ON CLASSROOM SCENE IN NYUD-V2.

	chair	table	cabinet	bag	sofa	counter	lamp	curtain	shelves	wall	floor	ceiling
Our	43.9	26.5	23.3	**4.4**	2.4	0.2	8.5	16.1	0	45.8	72.2	**83.3**
[9]	36.7	28	44.8	0	40.8	52	16.3	28.6	4.5	67.6	81.2	61.1
[10]	47.9	29.9	44.9	0.2	47.9	51.3	34.8	29.1	3.5	68	81.3	60.5

TABLE VI
PERFORMANCE COMPARISON ON DINING-ROOM SCENE IN NYUD-V2.

that gives the maximum rewards in that image. Figure 7 shows the graphical plot of the evaluation and it reveals that optimal combination outperforms the baseline methods and our LSPI learned policy.

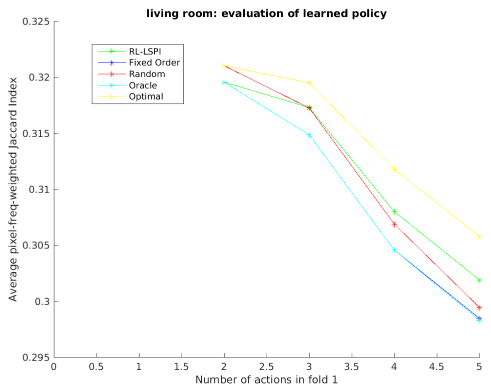

Fig. 7. Comparison of our *LSPI* learned policy against the *Optimal Combination* as well as three baselines: *Fixed Order*, *Random Order*, *Oracle* in the *living room* scene in NYUD-V2 dataset (best viewed in color). The yellow curve shows the evolution of *Optimal Combination* as more actions are taken into consideration. The Optimal Combination of segmentation outperforms all three baselines and our LSPI learned policy since it picks the combination that results in maximum reward for semantic segmentation.

D. *Comparison with Other Methods*

The quantitative comparison of the final semantic segmentation output from our LSPI learned policy on the 3-fold cross-validated experiments (explained in Section IV-A) is reported in Tables I to IX. We use the *pixelwise percentage Jaccard-Index*, the same evaluation metric used by the methods of [9, 10] for comparison. The metric is defined in Equation 13 as follows:

$$JI = \frac{100 * TP}{TP + FP + FN} \tag{13}$$

where TP, FP, FN are total number of true positive, false positive, and false negative pixels respectively for an object. These numbers are found from the confusion matrix of the entire scene specific test set. Each table shows our scene-specific

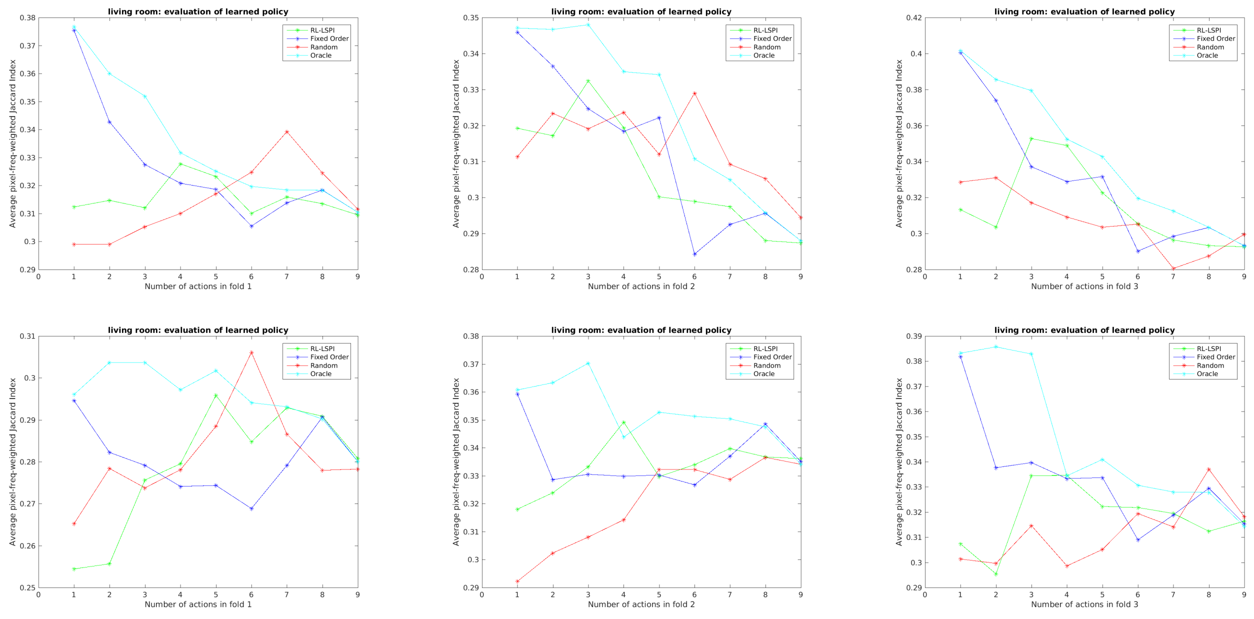

Fig. 6. Comparison of our *LSPI*-learned policies against three baselines: *Fixed Order*, *Random Order*, *Oracle* in the *living room* scene in NYUD-V2 dataset (best viewed in color). Each plot shows number of actions taken into consideration in the horizontal axis. The vertical axis shows the reward metric, i.e., average of the *pixelwise frequency weighted Jaccard Index* on the test-fold images. The first row shows comparisons in *Set I*. Second row shows the comparisons in *Set II*. Notice in this control experiment, we carefully prepared the train/test fold, where a pair of most-frequently occurring objects do not appear in every image of the folds. Our learned performs policy similar or better in all the folds compared to *Fixed Order policy*.

	chair	bookshf	table	desk	books	sofa	cabinet	bag	lamp	wall	floor	ceiling
Our	28.0	15.5	5.7	**11.7**	**28.9**	19.7	6.9	**6.3**	4.9	47.9	72.6	0
[9]	36.7	19.5	28	7.1	5.5	40.8	44.8	0	16.3	67.6	81.2	61.1
[10]	47.9	18.1	29.9	11.3	6.4	47.9	44.9	0.2	34.8	68	81.3	60.5

TABLE VII
PERFORMANCE COMPARISON ON HOME-OFFICE SCENE IN NYUD-V2.

	chair	cabinet	box	desk	table	shelves	counter	books	bag	wall	floor	ceiling
Our	29.9	19.2	**33.1**	6.3	4.3	4.1	0	**12.2**	0	37.9	**82.7**	59.4
[9]	36.7	44.8	2.1	7.1	28	4.5	52	5.5	0	67.6	81.2	61.1
[10]	47.9	44.9	2.1	11.3	29.9	3.5	51.3	6.4	0.2	68	81.3	60.5

TABLE IX
PERFORMANCE COMPARISON ON OFFICE SCENES IN NYUD-V2.

	sofa	table	chair	cabinet	pillow	lamp	bag	tv	books	wall	floor	ceiling
Our	27.1	14.4	14.1	6.2	10.2	10.1	**2.8**	9.0	5.3	50.8	69.7	**74.4**
[9]	40.8	28	36.7	44.8	30.3	16.3	0	4.8	5.5	67.6	81.2	61.1
[10]	47.9	29.9	47.9	44.9	34.4	34.8	0.2	31	6.4	68	81.3	60.5

TABLE VIII
PERFORMANCE COMPARISON ON LIVING-ROOM SCENE IN NYUD-V2.

experiment of sequential segmentation from the *LSPI* learned policy. Our method performs better in some categories in each scene and those numbers are highlighted. For example Table I shows that our method performed better for objects *counter, shower curtain, bag, and shelves* compared to methods of [9, 10]. Similarly Table III reveals that our method performs better on object categories *cabinet, bag, fridge, box, and floor*. Figure 4 demonstrates some qualitative results on *living room* scene.

V. CONCLUSION

We have demonstrated an alternative approach for semantic segmentation which affords simplicity and modularity, with comparable performance to the state-of-the-art to some categories when evaluated in a scene-specific manner. The methodological differences come from the need to adapting the existing models to different tasks, instead of training large complex models where the task is to maximize performance on the benchmark dataset. In the process we have also used alternative superpixel representations which favorably exploit both geometric and appearance properties of indoor environments. In the current work we evaluate the performance with respect to the task of semantic segmentation, but in the future we plan to explore the utility of the proposed representation for a larger variety of tasks.

ACKNOWLEDGMENTS

This work was supported by NSF IIS-1527208.

REFERENCES

[1] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and Sabine. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2012.
[2] Shai Bagon. Matlab wrapper for Graph Cut, December 2006.

[3] D. Banica and C. Sminchisescu. CPMC-3D-O2P: semantic segmentation of RGB-D images using CPMC and second order pooling. *CoRR*, 2013.

[4] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via Graph Cuts. *IEEE Transaction Pattern Analysis Machine Intelligence (PAMI)*, 2001.

[5] C. Cadena and J. Košecká. Semantic parsing for priming object detection in indoors RGB-D scenes. *The International Journal of Robotics Research*, 2014.

[6] J. Carreira and C. Sminchisescu. CPMC: Automatic object segmentation using constrained parametric mincuts. *IEEE Transaction on Pattern Analysis and Machine Intelligence (PAMI)*, 2012.

[7] D. Eigen and R. Fergus. Pedicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *International Conference on Computer Vision (ICCV)*, 2015.

[8] A. Geiger, P. Lenz, C. Stiller, and R Urtasun. Vision meets robotics: The KITTI dataset. *International Journal of Robotics Research (IJRR)*, 2013.

[9] S. Gupta, P. Arbelaez, and J. Malik. Perceptual organization and recognition of indoor scenes from RGB-D images. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.

[10] S. Gupta, R. Girshick, P. Arbelaez, and J. Malik. Learning rich features from RGB-D images for object detection and segmentation. In *European Conference on Computer Vision (ECCV)*. 2014.

[11] S. Karayev, T. Baumgartner, M. Fritz, and T. Darrell. Timely object recognition. In *Advances in Neural Information Processing Systems (NIPS)*. 2012.

[12] H. Koppula, Anand A., Joachims T., and A. Saxena. Semantic labeling of 3D point clouds for indoor scenes. In *Advances in Neural Information Processing Systems (NIPS)*, 2011.

[13] C. Kwok and D. Fox. Reinforcement learning for sensing strategies. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2004.

[14] M. G. Lagoudakis and R. Parr. Least-squares policy iteration. *Journal of Machine Learning Res.*, 2003.

[15] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[16] R. Mottaghi, X. Chen, X. Liu, S. Fidler, R. Urtasun, and A. Yuille. The role of context for object detection and semantic segmentation in the wild. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[17] B. Pepik, P. Gehler, M. Stark, and B. Schiele. 3D2PM - 3D deformable part models. In *European Conference on Computer Vision (ECCV)*, 2012.

[18] X. Ren, L. Bo, and D. Fox. RGB-(D) scene labeling: Features and algorithms. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[19] M. Reza and J. Košecka. Object recognition and segmentation in indoor scenes from RGB-D images. *Robotics: Science and Systems (RSS)- Workshop on RGB-D: Advanced Reasoning with Depth Cameras*, 2014.

[20] M. Reza and J. Košecka. Reinforcement learning for semantic segmentation in indoor scenes. *CoRR*, 2016. URL http://arxiv.org/abs/1606.01178.

[21] A. Saxena, A. Jain, O. Sener, A. Jami, D. K. Misra, and H. S. Koppula. Robobrain: Large-scale knowledge engine for robots. *CoRR*, 2014.

[22] C. Seiffert, T. M. Khoshgoftaar, J. V. Hulse, and A. Napolitano. RUSBoost: A hybrid approach to alleviating class imbalance. *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, 2010.

[23] N. Silberman and R. Fergus. Indoor scene segmentation using a structured light sensor. In *International Conference on Computer Vision (ICCV) - Workshop on 3D Representation and Recognition*, 2011.

[24] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from RGB-D images. In *European Conference on Computer Vision (ECCV)*, 2012.

[25] C. J. Taylor and A. Cowley. Parsing indoor scene using RGB-D imagery. In *Proceedings of Robotics: Science and Systems (RSS)*, 2012.

[26] J. Uijlings, K. Van de Sande, T. Gevers, and A. Smeulders. Selective search for object recognition. *International Journal of Computer Vision (IJCV)*, 2013.

[27] J. Yao, S. Fidler, and R. Urtasun. Describing the scene as a whole: Joint object detection, scene classification and semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[28] E.S. Ye. Object detection in RGB-D indoor scenes. Master's thesis, EECS Department, University of California, Berkeley, 2013.