Supplementary material for paper entitled: A Reinforcement Learning Approach to View Planning Problem

Mustafa Devrim Kaba* Mustafa Gokhan Uzunbas*
Ser Nam Lim
General Electric Global Research Center, 1 Research Circle, Niskayuna, NY 12309.
{mustafa.kaba, mustafa.uzunbas, limser}@ge.com

In this supplementary material, we first revisit the Remark 1 given in the paper, and give supporting illustrations about calculation of the boundary of a union of two submeshes. Second, we discuss the relationship between the model size and our new proposed score function and show that our solution to the view planning problem (VPP) is independent of the model size. Third, we provide the details of SARSA algorithm for the reader's convenience. Forth, we analyze the *next best view* (NBV) selection behavior of the methods included in the paper. Then, we show the complete set of plots for value function approximation performance of Reinforcement Learning agents (i.e. SARSA, Watkins-Q and TD). Finally, we show additional coverage results for the whole dataset.

1. Calculating the Boundary of the Covered Region on the 3D Model

In the paper, in Remark 1, we mentioned that given two submeshes $X_1, X_2 \subseteq \Omega$, the boundary $bd(X_1 \cup X_2)$ can be calculated as follows

$$bd(X_1 \cup X_2) = [bd(X_1) \setminus ed(X_2)] \cup [bd(X_2) \setminus ed(X_1)] \cup [bd(X_1) \cap bd(X_2)] \quad (1)$$

where $ed(\cdot)$ denotes the set of all edges of the submesh. This equation is crucial in calculation of the new boundary of a covered region after adding the next view to our existing coverage. This formulation can handle all possible scenarios: i) when X_1, X_2 do not intersect, ii) X_1, X_2 intersect with overlapping faces, iii) X_1, X_2 intersect with only overlapping edges. In Figure 1 we illustrate how Eq. 1 would work on two simple submeshes without loss of generality. In this figure, X_1, X_2 are represented by gray and blue colored simple triangulations in the top left image.

*These authors contributed to this paper equally.

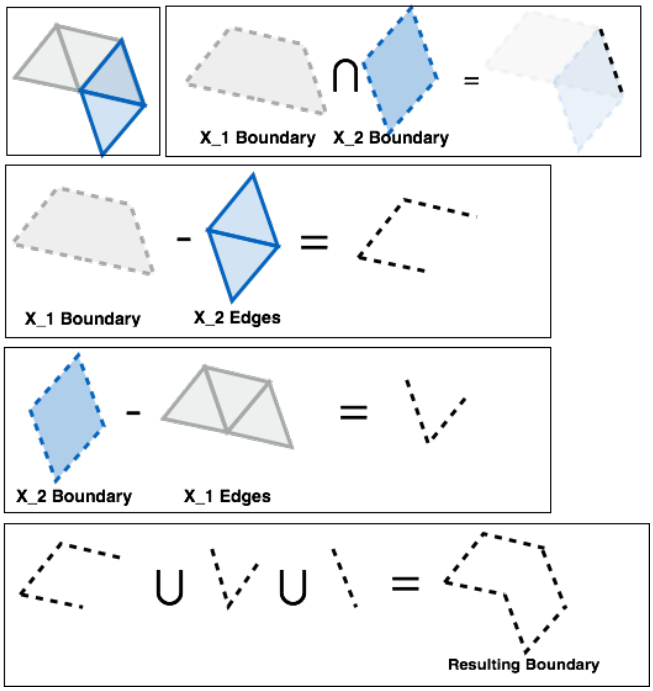

Figure 1. Illustration of calculating the new boundary of the surface area seen by all cameras after adding a new camera.

2. The Model Size and the Score Function

We find it helpful to have a brief discussion on the relationship between the model size and the score function, and hence the proposed method. Suppose that we solved a VPP for a given model, a fixed set of cameras, and a set of lambdas Λ. Then, it is natural to ask whether our method would come up with a different solution if the model size and the camera parameters were altered while keeping Λ the same. Because if this was the case, the solution would depend on the model size, which, we think, would be quite undesirable. Fortunately, this turns out to be not the case. We argue as follows: For two cameras assume that we have two associated covered submeshes M_1 and M_2. Suppose

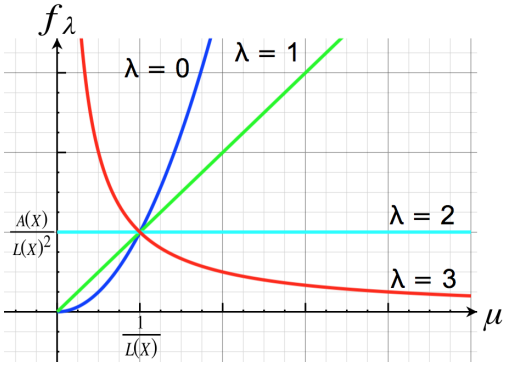

Figure 2. The relationship between the model resize factor μ and the score function.

further that when we resize the model (and the camera parameters accordingly) we find the new submeshes $\tilde{M}_1$ and $\tilde{M}_2$. Let's denote the resize factor by μ, which is assumed to be a positive number. Resizing a triangle by a factor μ multiplies its area by μ^2 and each of its edges by μ. Since our meshes are collections of triangles, the same rule applies to the submeshes. That is, the equalities

$$\mathcal{A}(\tilde{M}_i) = \mu^2 \mathcal{A}(M_i) \text{ and } \mathcal{L}(\tilde{L}_i) = \mu \mathcal{L}(M_i) \qquad (2)$$

are satisfied for $i = 1, 2$. If, for the scores of X_1 and X_2, we have $\frac{\mathcal{A}(M_1)}{\mathcal{L}(M_1)^\lambda} < \frac{\mathcal{A}(M_2)}{\mathcal{L}(M_2)^\lambda}$, then the equations (2) yield $\frac{\mathcal{A}(\tilde{M}_1)}{\mathcal{L}(\tilde{M}_1)^\lambda} < \frac{\mathcal{A}(\tilde{M}_2)}{\mathcal{L}(\tilde{M}_2)^\lambda}$. Therefore, we conclude that the ordering of the cameras would not change, even if we resized the model. Hence, the solution we find is independent of the model size, as desired. It is important to note, however, that the score function often changes with μ, even though the ordering of the cameras does not change. For an illustration of the behavior of the score function with changing μ, we refer to Figure 2.

3. SARSA Algorithm

We described our implementation of Watkins-Q and TD algorithms in the paper. For the sake of completeness, in algorithm box (1) we include the implementation of SARSA algorithm, too.

4. Analysis of NBV suggested by each Method

In Figure 3 we plot the change in coverage after camera selections for each model and each method during test. As expected, the greedy method (shown in blue dashes) tries to cover the model surface with long steps as much as possible. Notice the big jumps on the y-axis at the beginning of the planning. The Alternating- λ (shown in cyan color) method starts like the greedy method but then continues alternating between shorter and longer steps (i.e. greedy, non-greedy, greedy,... etc.). On the other hand, the Reinforcement Learning (RL) agents follow completely non-trivial strategies to finish the coverage task. For example, SARSA, TD and WQ (shown in red, green and magenta, respectively) learn to start with shorter steps at the beginning and then continue with unpredictable varying steps. These agents learn to consider future steps at the early stages of the camera selection process.

Algorithm 1 SARSA Agent

procedure TRAIN
  $\theta \leftarrow$ *random network weights*
  $\alpha \leftarrow$ *learning rate*
  $\lambda_e \leftarrow$ *eligibility* $- \lambda$
  repeat
    $c \leftarrow$ random view point
    $s \leftarrow \{c\}, e \leftarrow 0, r \leftarrow -1, \delta \leftarrow 0$
    $\lambda^* \leftarrow \arg\max_\lambda \hat{q}_\pi(\theta, s, \lambda)$
    while *true* **do**
      $e \leftarrow e + \nabla_\theta \hat{q}_\pi(\theta, s, \lambda^*)$
      $\delta \leftarrow r - \hat{q}_\pi(\theta, s, \lambda^*)$
      if *s is Terminal* **then**
        $\theta \leftarrow \theta + \alpha \cdot \delta \cdot e$
        break
      $c \leftarrow \text{NBV}(\lambda^*)$
      $s \leftarrow s \cup \{c\}$
      $\lambda^* \leftarrow \arg\max_\lambda \hat{q}_\pi(\theta, s, \lambda)$
      $\delta \leftarrow \delta + \hat{q}_\pi(\theta, s, \lambda^*)$
      $\theta \leftarrow \theta + \alpha \cdot \delta \cdot e$
      $e \leftarrow e \cdot \lambda_e$
  until Last episode

5. Value Function Approximation Performance

In the Results and Discussions section of the paper we discussed the value function approximation quality for two cases only. It was part of a much bigger experiment which examined the performance of 36 neural nets. The error plots of all 36 cases can be found in Figures 6, 7 and 8.

6. More Results

In Figure 4, on three models, we illustrate the coverage update after selecting a new camera when SARSA agent is used with *RCC* equals .92. Rows [2-4] show images seen by the selected camera at each step of the process. Coverage information is shown by red (regions observed first time by that camera) and green (regions observed by multiple cameras) colors on the images. In Figure 5, we provide the coverage results for SARSA agent for the whole dataset included in the paper.

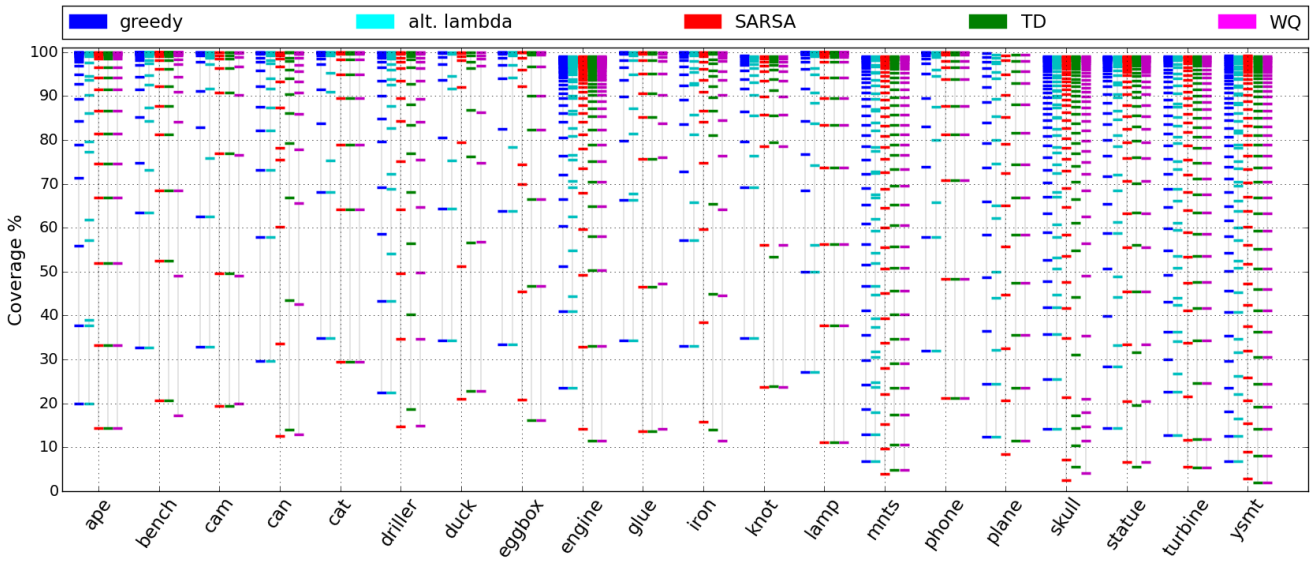

Figure 3. Visualization of the coverage process for twenty models for all methods.

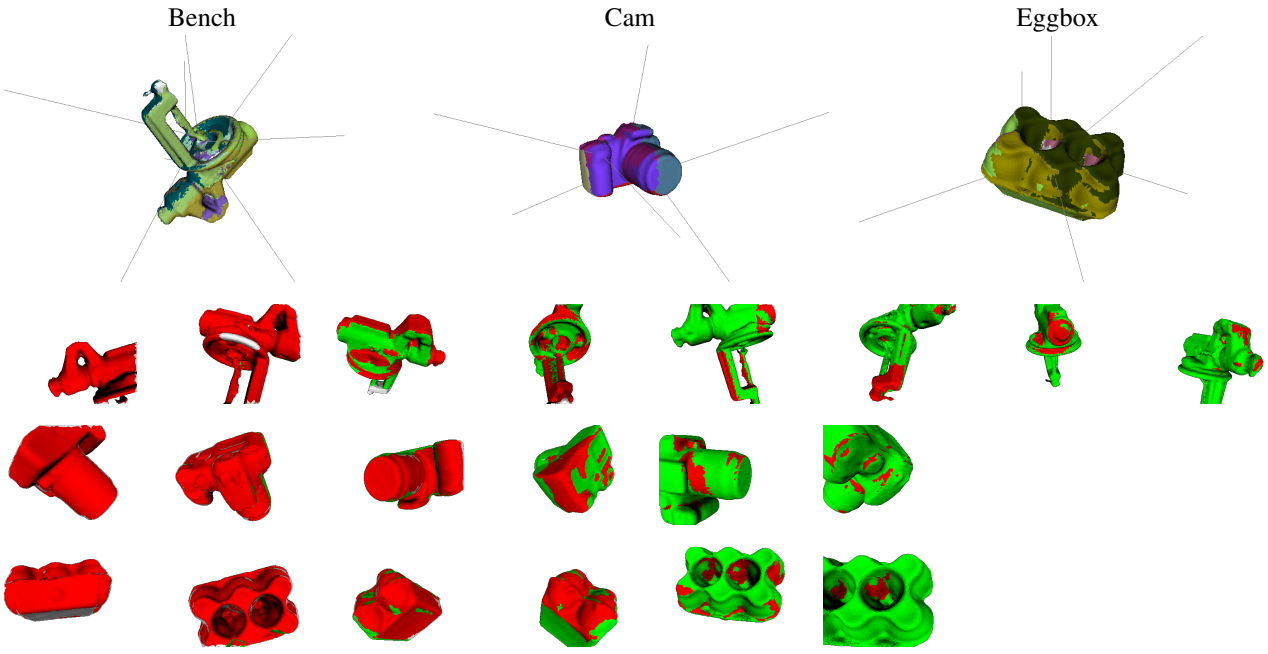

Figure 4. Top row: Coverage result for three objects from second dataset with relative coverage criteria *RCC* equals 0.92. Surface colors represent regions covered by the cameras. Rows [2-4] illustrate the update in coverage after selecting a new camera during test time by the SARSA agent. On each image, red color represents the surface seen by that camera for the first time. Green color represents surfaces that have been seen before. Colorless surfaces are unknown to the current camera meaning that they are not visible because visibility quality criteria is not satisfied. Please see definiton of RCC in the original paper.

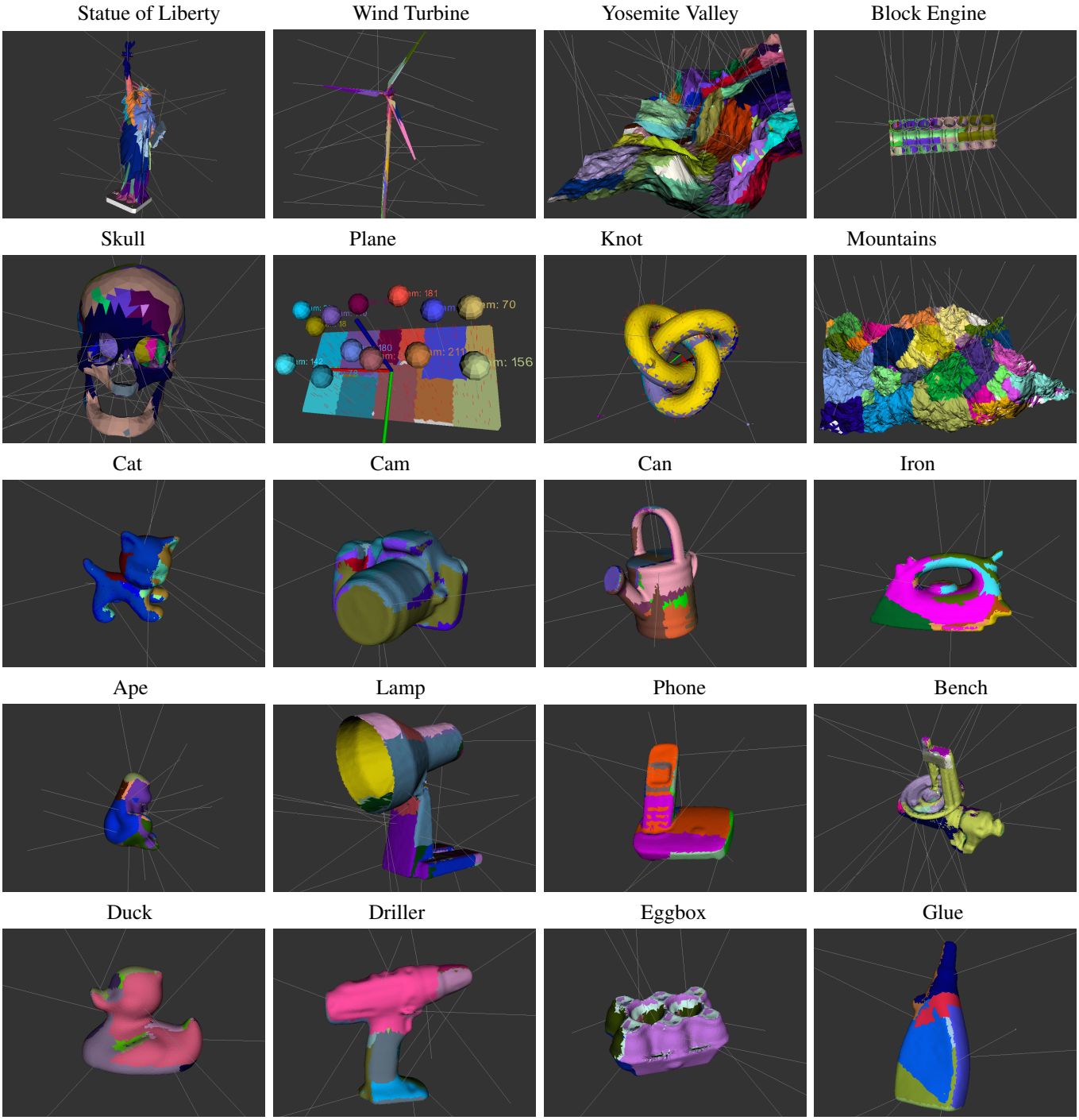

Figure 5. Visual coverage results on the models by SARSA agent.

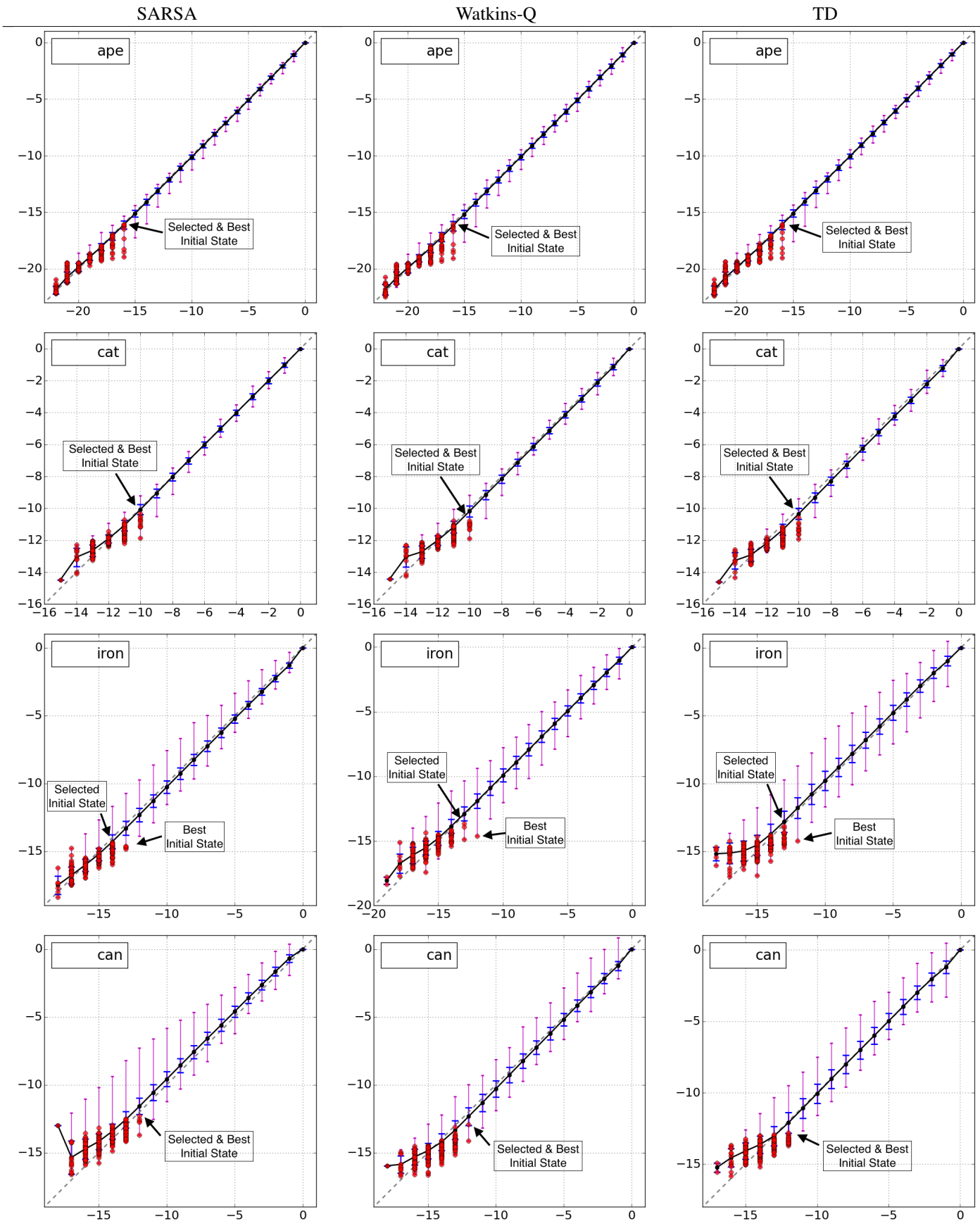

Figure 6. Value function approximation performance.

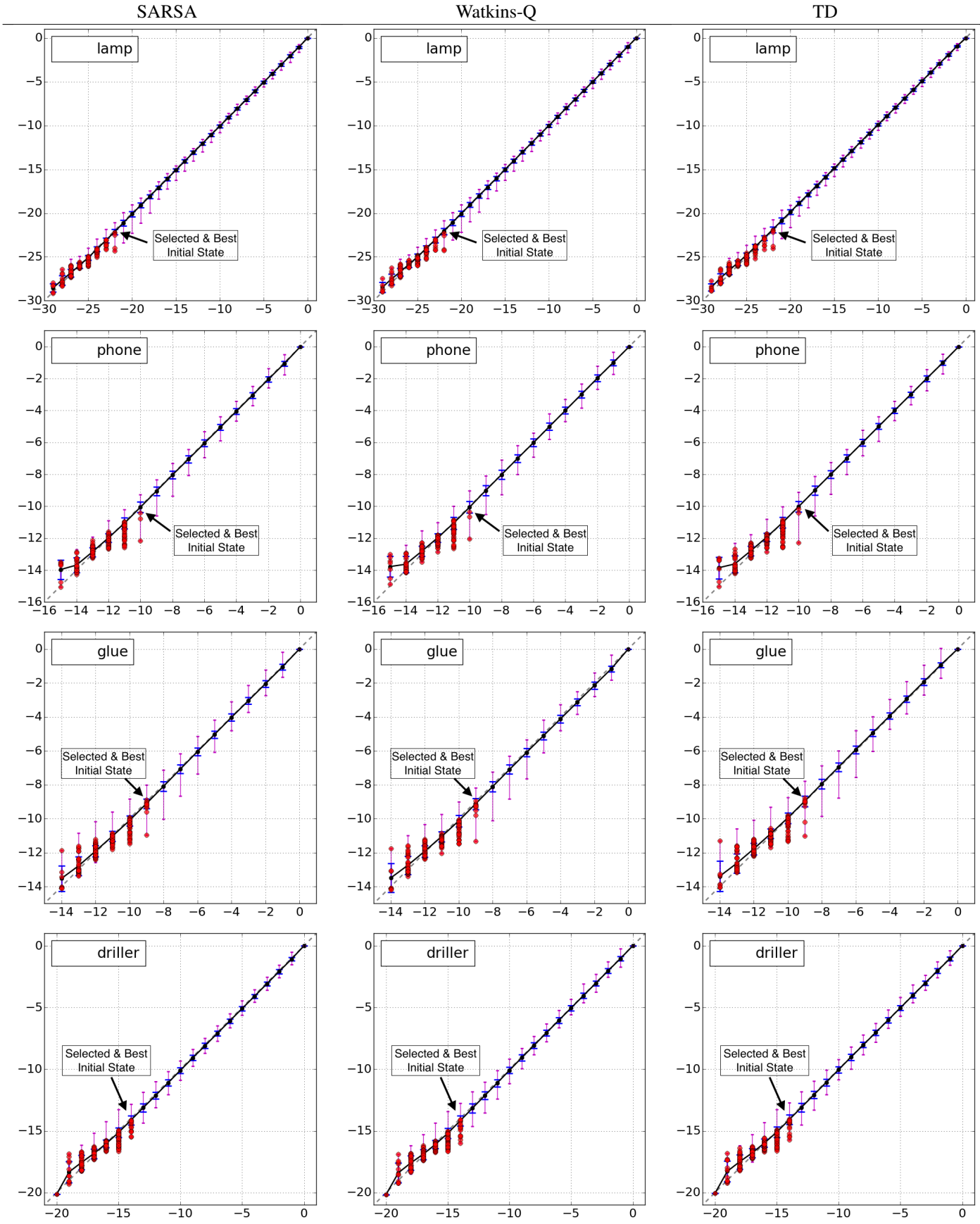

Figure 7. Value function approximation performance.

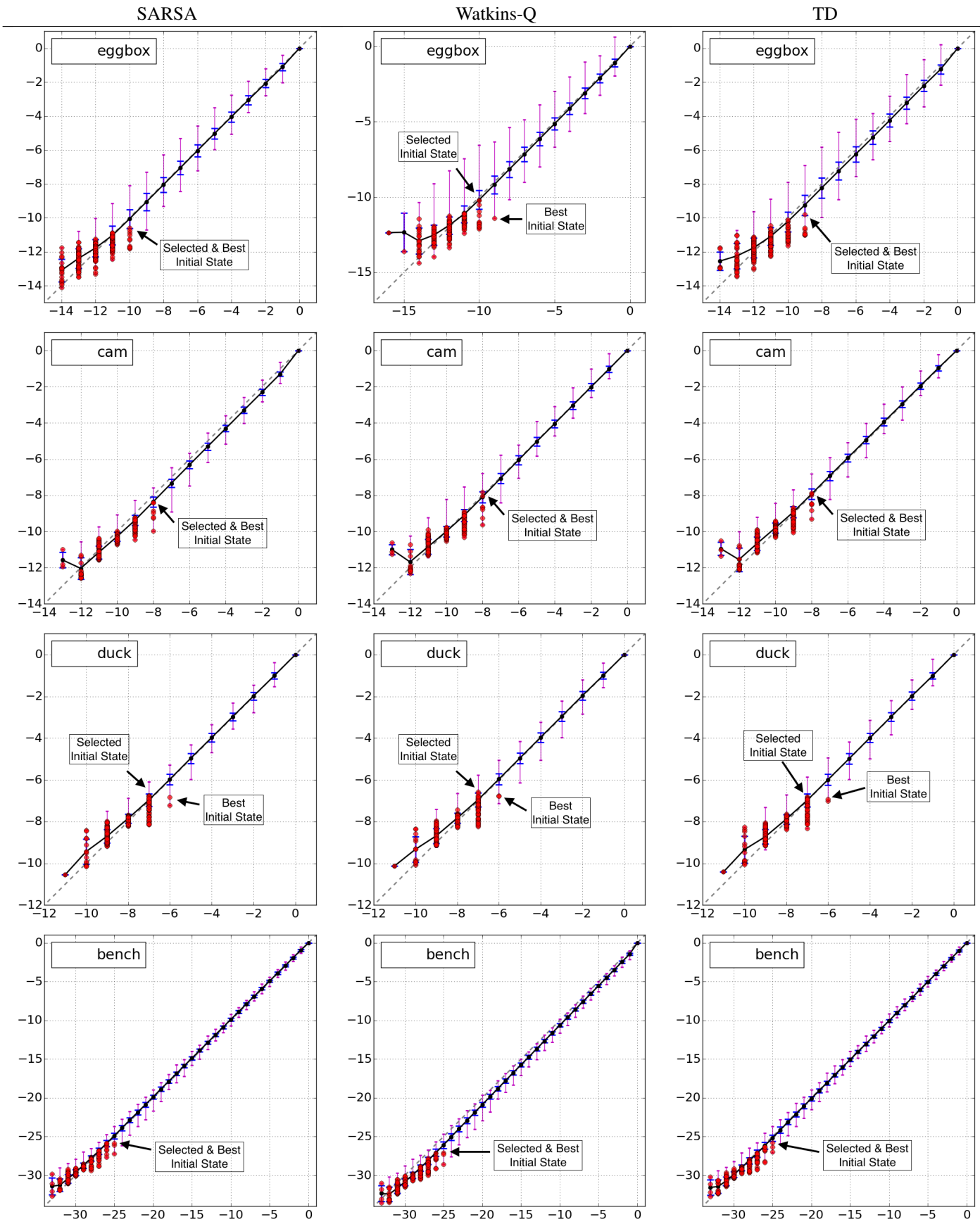

Figure 8. Value function approximation performance.