IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received August 30, 2018, accepted September 19, 2018, date of publication September 28, 2018, date of current version October 19, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2872693

A Computational Framework for Automatic Online Path Generation of Robotic Inspection Tasks via Coverage Planning and Reinforcement Learning

WEI JING[1,2], CHUN FAN GOH[3], MABARAN RAJARAMAN[3], FEI GAO[1,2], SOOHO PARK[3], YONG LIU[1,2], AND KENJI SHIMADA[3]

[1]Department of Computing Science, Institute of High Performance Computing, Singapore 138632
[2]A*STAR Artificial Intelligence Initiative, Singapore 138632
[3]Department of Mechanical Engineering, Carnegie Mellon University, Pittsburgh, PA 15213, USA

Corresponding author: Wei Jing (jing_wei@ihpc.a-star.edu.sg)

This work was supported by the Agent for Science, Technology and Research (A*STAR), Singapore, through the A*STAR Human-Centric Artificial Intelligence Programme under SERC SSF Project A1718g0048.

ABSTRACT Surface/shape inspection is a common and highly repetitive task in the factory production line. Using robots to automate the inspection process could help to reduce the costs and improve the productivities. In robotized surface/shape inspection application, the planning problem is to find a near-optimal sequence of robotic actions that inspect the surface areas of the target objects in a minimum cycle time, while satisfying the coverage requirement. In this paper, we propose a novel computational framework to automatically generate efficient robotic path online for surface/shape inspection application. Within the computational framework, a Markov decision process (MDP) formulation is proposed for the coverage planning problem in the industrial surface inspection with a robotic manipulator. A reinforcement learning-based search algorithm is also proposed in the computational framework to generate planning policy online with the MDP formulation of the robotic inspection problem for robotic inspection applications. Several case studies are conducted to validate the effectiveness of the proposed method. It is observed that the proposed method could automatically generate the inspection path online for different target objects to meet the coverage requirement, with the presence of pose variation of the target object. In addition, the inspection cycle time reduction is observed to be 24% on average compared to the previous approaches during these test instances.

INDEX TERMS Robotics, planning, artificial intelligence, reinforcement learning.

I. INTRODUCTION

A. BACKGROUND

In recent years, due to the rapid advancement of technologies and the increase of labor costs, using robots with artificial intelligence algorithms in the industrial automation to save cost and improve the productivities has attracted much attention in various industrial sectors. One example of those industrial automation applications with robots is the surface/shape inspections in factory production lines [1], [2]. In many manufacturing processes, the resultant surface profile of a manufactured product may be different from its intended designed CAD profile. Thus, applying 3D surface/shape inspection is required [3], [4] to ensure the quality of the manufactured products in the factory production line.

Among the technologies involved in the robotized surface/shape inspection application, robotic path generation technology plays a crucial role for cost reduction and productivity improvement. There are several requirements to design proper robotic path generation algorithm for these inspection applications in factory production line. First, the algorithm should be able to automatically generate path for different target objects with different sizes and geometries, while explicit manual robotic programming is not required with new incoming products. Secondly, the algorithm should be able to generate efficient robotic path, because this kind of inspection task is highly repetitive and the inspection cycle time is desired to be minimized for cost reduction. Moreover, the algorithm should be able to generate path online such that

the uncertainties encountered during the inspection process can be properly handled; the sources of the uncertainties include pose variation, surface inconsistency, measurement noise and so on [5]–[7]. In this paper, the uncertainty we are mainly addressing is the pose variation of the target objects due to the errors in workpiece placement/localization process, which is one of the most important issue found on factory production line.

B. RELEVANT WORK

Automatically generating robotic path for the surface/shape inspection tasks is usually considered as a Coverage Planning Problem (CPP) [8]. For the inspection applications, the 3D CAD model of the target object is usually known prior to the planning process, although certain deviations (including the shape and pose variations) between the CAD model and actual object may exist [6]. Usually two sub-problems are involved in the CPP: the View Planning Problem (VPP) and the path planning problem [2], [9]. The former problem refers to finding a set of viewpoints that cover the required surface areas of the target object [5]; and the later refers to finding a collision-free optimal (e.g., shortest time) robotic path that visits the selected viewpoints [6]. For the inspection applications, the VPP is usually formulated as an NP-hard Set Covering Problem (SCP) or its variations [5], [7], and solved by various approximation algorithms [10]. The path planning problem is often formulated as a Traveling Salesman Problem (TSP) for the set of the selected viewpoints [11], [12].

The SCP and TSP in the model-based CPP are coupled problems [6], [13], and solving them in two sequential separate processes usually decreases the quality of the optimization results. In recent years, several improvements have been made to the CPP solutions in robotic applications, including combining the SCP and TSP to solve the two problems in one optimization process to achieve better results [2]; or using more effective sampling strategies to generate the candidate viewpoints to improve the results [10]. These CAD-based offline planning methods plan the robot inspection path only in an offline and open-loop manner, therefore, they require that the target workpieces to be placed and localized accurately in the workspace with little pose errors. However, in the industrial surface/shape inspection settings on production line, the placement/localization of the workpiece often come with non-negligible errors, then the offline methods could fail and be unable to meet the coverage requirements of the inspection applications [6], [7].

On the other hand, the online planning methods for the inspection applications could help to solve the issue by taking the feedback and compensating the errors on-the-fly during each iteration. The Next-Best-View (NBV) methods were the online planning methods proposed for view / coverage planning problem, as summarized in the surveys [5], [8], [14]. NBV-based methods select next viewpoint greedily on-the-fly, with different modeling method and selection criteria [15], [16]. The NBV-based methods are robust to handle the uncertainties in the inspection applications, since they generates the planning policy online. However, NBV-based methods usually only consider the local best move as the planning policy at each step, and could generate less efficient planning policy for the inspection application.

Along with other successful applications of Reinforcement Learning (RL) [17] as demonstrated in AlphaGo [18], robotics [19] and Atari games [20], the RL techniques have been applied to the viewpoint planning problems recently [21], which optimizes the total return of the task rather than the one-step rewards. With the Markov properties on the state space [21], RL-based method is able to handle online planning problem because the current states encode all the information. Then the Q-learning, and SARSA have been applied to the SCP from the VPP through learning a variable λ that adjusts the local search criteria. The work, however, mainly focuses on viewpoints selection problem without considering the path planning or the complexities of the integrated path-viewpoints planning problems.

C. CONTRIBUTIONS

In this paper, we propose a novel method to automatically generate efficient robotic path online in the factory surface/shape inspection application, with the presence of pose variation of the target objects. The main features of the proposed method are:

- a novel computational framework that generates the robotic inspection policy for different target workpieces with different sizes and geometries, without any explicit manual robotic programming required;
- a Markov Decision Process (MDP) formulation of coverage planning problem for the robotic surface/shape inspection application;
- a Reinforcement Learning (RL) based tree search algorithm that generates the policy on-the-fly to select the actions with the proposed MDP formulation for the robotized inspection application;

II. PROPOSED MDP FORMULATION AND SYSTEM MODELING

In robotic inspection applications, the path generation algorithm is to find the viewpoints and robot paths that complete the inspection task with required surface coverage, as illustrated in Fig. 1. Because of the coverage constraints, the path generation problem for inspection application is often referred as a Coverage Planning Problem (CPP). Many previous formulations [2], [13] consider the CPP for inspection application as an offline planning problem in a known static environment, these formulations may fail if there are certain shape discrepancies or pose variation of the target objects (workpieces).

In this paper, to address the uncertainties encountered during the inspection process on production line, especially the pose variation of the target workpieces due to the errors in workpiece placement/localization process, we propose a Markov Decision Process (MDP) formulation to model the

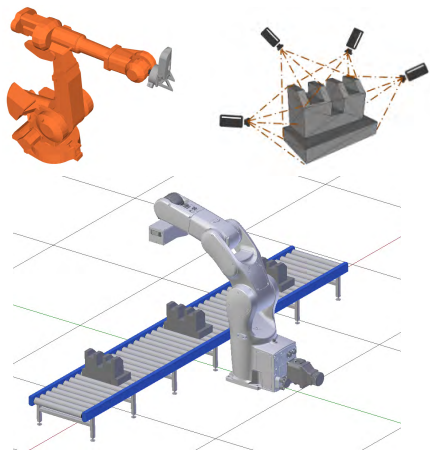


FIGURE 1. Overview: robotic surface/shape inspection in factory production line.

CPP of the robotized inspection task. With the proposed MDP formulation, efficient robotic inspection path can be automatically generated on-the-fly, which is robust and able to handle the pose variation problem.

A. ROBOTIC SURFACE/SHAPE INSPECTION AS COVERAGE PLANNING PROBLEM

We first discuss the formulation of the Coverage Planning Problem (CPP) for surface/shape inspection on production line with robot. The CPP is to generate the robot motion plan that consists of:

- a set of viewpoints that covers the required areas of the target object,
- the robotic paths to move the 3D scanner between the viewpoints,
- and the visiting sequence of the viewpoints.

The objective is to minimize the total cycle time. As discussed in [2], the CPP can be formulated as a combined View Planning Problem (VPP) and path planning problem, with the objective is to minimize the overall sum of inspection cost and traveling cost, as described in Eq. 1 below:

$$\min_{\mathbf{a}} \quad \underbrace{\sum_{a_i \in \mathbf{a}} T_{ins}(s_i)}_{\text{inspection cost}} + \underbrace{\sum_{a_i \in \mathbf{a}} T_{travel}(s_{i-1}, a_i)}_{\text{traveling cost}}, \tag{1}$$

where a_i is the action taken at i^{th} time step; s_i is the robot state at the i^{th} time step, determined by the starting state and actions taken: $s_i = f(s_0, a_0, a_1, \cdots, a_{i-1})$. An action in this paper refers to choosing a viewpoint and moving the robot to the viewpoint. The coverage constraints are required but not explicitly listed in the above formulation, as it is usually slightly varied according to the application requirements.

B. MARKOV DECISION PROCESS FORMULATION FOR COVERAGE PLANNING PROBLEM

In this paper, we propose a finite MDP formulation for the online CPP of the robotized inspection task. The finite MDP is defined as a tuple $(\mathcal{S}, \mathcal{T}, \mathcal{A}, r, \lambda)$, where $\mathcal{S}$ is the state space; $\mathcal{T}$ is the state transition model; $\mathcal{A}$ is the action space; $r : \mathcal{S} \times \mathcal{A} \rightarrow r \in \mathbb{R}$ is the rewards; and $\lambda \in [0, 1]$ is the discount factor of the rewards, which is to emphasize the current reward over the future reward.

The inspection applications in this paper follows the episodic setting, where the episode is considered as ended when the required coverage ratio is achieved. The objective is to determine the next action to take, based on the current robot pose and the current observed information of the target workpiece. The action here refers to choosing a viewpoint and moving the robot to place the 3D scanner to the viewpoint. In the proposed MDP formulation, the state space is constructed by augmenting the all previous observed information of the target workpiece to the current robot poses. With state augmentation, a state s in the state space $\mathcal{S}$ consists of two parts:

- the current pose of the robot p_s that is drawn from a finite sampled set $\mathcal{P}_s$, and
- a vector $\mathbf{m}^s_{surf}$ consisting of the coverage / observation information of the surface patches of the target object.

Then with the state formulated as $[p_s, \mathbf{m}^s_{surf}]$, all previous information is encoded in the current state, thus the requirements Markov properties are satisfied.

For the rewards and return in the MDP, because the objective of the industrial inspection applications is to minimize the total cost, which is the overall inspection and traveling cost after meeting the surface coverage requirement. Therefore, instead of maximizing the total return, the MDP formulation of CPP is to minimize the total cost.

III. THE PROPOSED COMPUTATIONAL FRAMEWORK

In this paper, we propose a novel computational framework for the online path generation problem of the robotic shape/surface inspection applications in the factory production line. The proposed framework takes the robot model, target object model, and the sensor specifications as input, automatically generates the inspection policy online. There are four individual sub-modules in this proposed framework, as listed below:

- first, the viewpoints are randomly sampled around the target object, and the robot poses to place the scanner to the viewpoints are also computed;
- then the local planning module computes the collision-free trajectories between those robot poses at the sampled viewpoints;
- after that, visibility modeling and approximation step evaluates the visibility of each viewpoint and surface patch;
- lastly the online RL based planning algorithm is applied to choose actions for inspection with the MDP formulation, until the inspection process is completed.

A brief summary of the proposed computational framework is shown in Fig. 2. The input and output of each sub-module is also described in the figure.

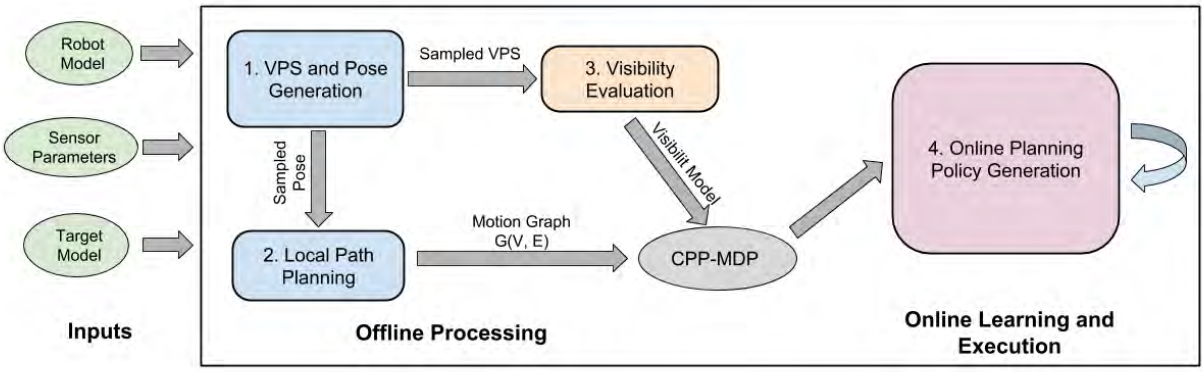


FIGURE 2. The proposed computational framework.

A. VIEWPOINTS AND POSE GENERATION

The first step of the proposed computational framework is to generate the candidate viewpoints for the inspection task. The viewpoints generation is done by randomly sampling redundant viewpoints around the target object in the Euclidean space. There are different methods that can be applied to optimize the sampling efficiency, such as voxel dilation [2], Medial Object [6], potential field [10] and so on. For this paper, in order to reuse the sampled viewpoints, the corresponding poses, as well as the local planning trajectories for target objects with different geometries, we randomly sample viewpoint positions within a ellipsoid hemisphere in the workspace. The center of the ellipsoid hemisphere used for sampling is same as the center of the desired placement position of the target object.

The viewing direction $\mathbf{z}_i$ of a sampled viewpoint is firstly generated to point to the center of the desired placement position of the target object. Then randomized variance modeled by Gaussian distribution in 3D space is added to the viewing direction $\mathbf{z}$ to improve the exploration, as shown in Eq. 2. This strategy ensures the viewpoints and local planning trajectories could be reused for different target workpieces to improve the computational efficiency.

$$\mathbf{z}' = \frac{\mathbf{z} + k_1 \mathbf{x}_\mathbf{z}}{\|(\mathbf{z} + k_1 \mathbf{x}_\mathbf{z})\|}, \tag{2}$$

where $\mathbf{x}_\mathbf{z} \sim \mathcal{N}_3(\mathbf{0}, \Sigma)$ is the multivariate Gaussian distribution, k_1 is a parameter used to adjust the scale of the randomization.

For each sampled viewpoint, a corresponding robot pose is computed through Inverse Kinematics (IK). In this paper, IK computation is implemented using the IKfast algorithm [22], which is a high performance, open source and general IK solver for serial robotic manipulators. IKfast is available as an API under Robotic Operating System (ROS) [23] environment. The computed robot poses are also validated by checking the collision with the environment through Flexible Collision Library (FCL) [24], where a slightly larger bounding cylinder is used to model the collision of target object in this paper, to ensure the collision detection results can be reused and certain safety buffer is included. If there is a failure in either the IK calculation or collision checking for a viewpoint, the viewpoint is removed from the set of candidate viewpoints. An example visualization of sampled viewpoints after validating with the environment is shown in Fig. 3.

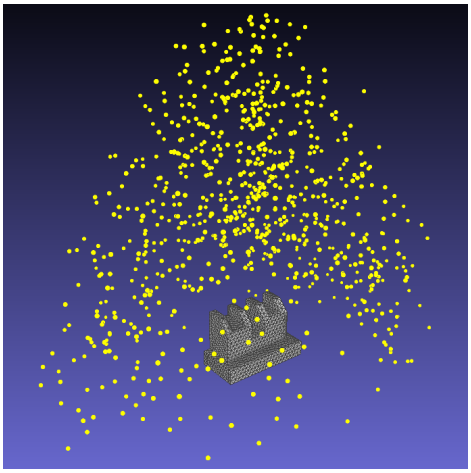


FIGURE 3. Example of sampled viewpoints.

B. LOCAL PLANNING

Local planning sub-module is to find the collision-free robot trajectories to move the 3D scanner from one viewpoint to another. In this paper, sampling-based planning method [25] is used to plan the robotic path and evaluating the local traveling cost between the candidate viewpoints. The choice of local planner can be flexible, depending on the environment and requirement of the applications. We use Rapidly-exploring Random Trees-Connect (RRT-Connect [26]) to search for the local paths between viewpoints. In this paper, MoveIt [27] with Open Motion Planning Library (OMPL) [28] is used in the implementation as a planning framework, under ROS [23] environment (ROS-Indigo, Ubuntu 14.04 LTS), where the local planning algorithm RRT-Connect can be directly called as an API function through ROS service.

The resultant local paths are parametrized with cubic spline and stored as ROS trajectory format, which can be reused in the future applications as long as the collision environment stays unchanged. The trajectory parametrization with cubic spline is done through MoveIt [27] with ROS. They can also be directly used with ROS-Industrial Package [29] for the real-world robot, similar to the previous implementation in [2]. An example of robotic path computed by the local planner with collision awareness is shown in Fig. 4.

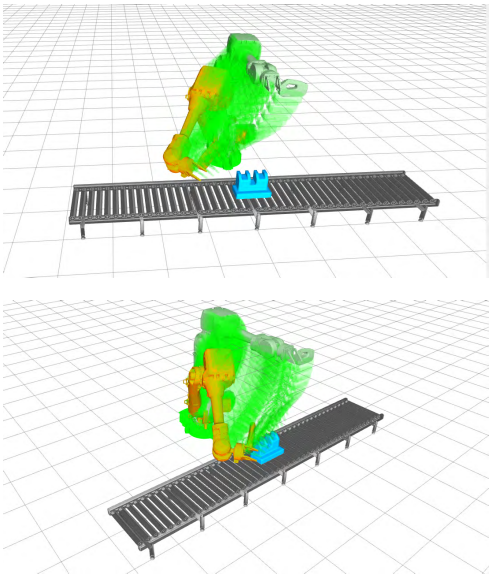


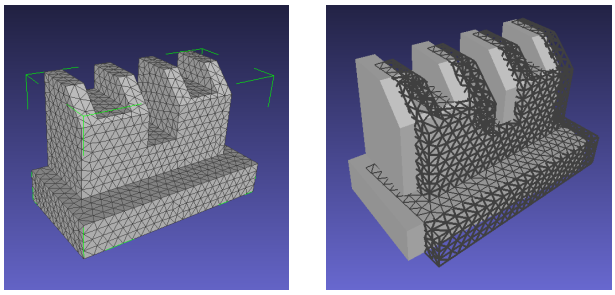
FIGURE 4. Example of collision-free robotic path computed by local planner from different viewing angle: the starting robot pose is shown in gray, the path is shown in green, and the end robot pose is shown in orange.

The resultant trajectories and traveling costs are then stored in the edge E_{vp} of a motion Graph $\mathcal{G}(V_{vp}, E_{vp})$. With the visibility matrix and motion Graph computed, the MDP for CPP of robotic inspection task discussed in this paper can be then formulated. The state transition function $\mathcal{T}(s, a)$ can be found by querying the visibility matrix to update the visibility information $\mathbf{m}^s_{surf}$, and querying the motion Graph $\mathcal{G}(V_{vp}, E_{vp})$ to update the robotic pose p_s.

C. VISIBILITY MODELING AND APPROXIMATION

For the inspection application, the model of the target object is known and in triangular mesh format. There are usually geometric deviations due to the manufacturing process, as well as the pose variations due to the workpiece placement/localization error on the production line [2]. In this paper, we mainly address the pose variation problem that is a common problem on production line. An example of pose variation is shown in Fig. 5. The surface of the target workpiece is uniformly re-sampled using Bubble Mesh [30] method. These triangular patches on the surface of the target workpiece are then used for the estimation of the visibility information.

The visibility information between the candidate viewpoints and surface patches of target object is estimated and



(a) (b)

FIGURE 5. (a). Resampled target object with 2252 triangular patches in total on the surface; the bounding box size is 187*mm* × 374*mm* × 237*mm*; (b). Pose variation of target object of 20*mm* in both x,y axis, and 5° in z-axis.

stored in a $n_{surf} \times n_{vp}$ visibility matrix, based on the specifications of the 3D scanner, as well as the poses of the viewpoints and the surface patches. n_{surf} is the number of surface patches on the target object, n_{vp} is the number of successfully sampled candidate viewpoints. The visibility model used in this paper is similar to many previous approaches [2], [7], the estimation criteria are listed as follows:

- The surface patch must be in the Field of View (FOV) of the sensor from the viewpoint.
- The surface patch must be in the viewing range of the sensor from the viewpoint.
- The viewing angle must be within a range given by the sensor specifications.
- There must be no occlusion between the viewpoint and surface patches.

The approximated visibility information will be used in the later RL-based online planning method. Note that there are uncertainties of the visibility, which is caused by the pose variation of the target object and limits the usage of the offline CAD-based planning method. The visibility matrix is pre-computed so that fast querying of visibility information between the candidate viewpoints and the surface patches in later steps to update the state variable $\mathbf{m}^s_{surf}$.

D. ONLINE COVERAGE PLANNING POLICY GENERATION

With the formulation of MDP, RL-based method can be used to compute the inspection policy online. In this paper, we propose a variation of Monte-Carlo Trees Search (MCTS) method, which we termed as ϵ-greedy Forward Tree Search (FTS) method for online planning policy search, with the MDP formulation for the robotized inspection application.

MCTS and its variations have been successfully applied to many applications in recent years, such as the game of Go [18], [31], robotics [32], optimization [33]. It is considered as an important category of RL algorithms [17], [33]. In this paper, we utilize the proposed ϵ-greedy FTS method to solve the MDP formulated for the CPP in robotized inspection application. The planning policy is computed online by iteratively building a search tree at each viewpoint during the

inspection task, the inspection task is considered as complete when the required coverage of the target surface is achieved.

Following the similar general structure of MCTS [33], four steps are involved in the ϵ-greedy FTS algorithm: *Selection*, *Expansion*, *Simulation* and *Back-propagation*, where the first two steps are usually combined as a tree policy, and a default policy is ran at the *Simulation* step. The proposed method starts by constructing a root node v_0 based on the current state s_0 (*InitializeNode*(s_0)). In this tree policy, the *Selection* steps refers to selecting a node on the tree, then *Expansion* will create a child node of the selected node. After that, the *Simulation* step will run a default policy based on the state transition model $\mathcal{T}$ of MDP, until the end of the episode. Lastly the *Back-propagation* step updates the information of the node and back trace to the root node. After building the search tree, the best child of the root node is selected as the next action, then the whole process is repeated [33], until the completion of the inspection task.

Each node v in the Tree structure stores four variables: the MDP state s_v, the count of visit N_v, the total cost c_v, and the minimum cost q_v. The choice of best child will consider the q_v, N_v and c_v. As shown in Algo.1, the difference of the ϵ-greedy FTS method is with the tree policy and simulation policy modified for the CPP for inspection applications, as described in Algo. 2 and Algo. 3. The *BackPropagation*(v_0, v, $cost_v$) is to update the visiting count and the cost of the nodes along the simulated path at each iteration. Similar to general MCTS, the action will be taken when the maximum search iteration is achieved, the process repeats until the end of the episode, when the coverage constraint is satisfied.

Algorithm 1 ϵ-greedy FTS for CPP

Input: The current state, $s_0 \in \mathcal{S}$
Output: The action to take in current state, $a_{v_0'} \in \mathcal{A}$

```
v_0 ← InitializeNode(s_0)
while within max iteration do
    v ← TreePolicy(v_0)
    cost_v ← DefaultPolicy(v)
    BackPropagation(v_0, v, cost_v)
end while
a_{v_0'} ← BestChild(v_0)
return a_{v_0'}
```

In the tree policy stage, two steps are usually involved: *Selection* and *Expansion*. In this paper, we use a ϵ-greedy approach in tree policy to balance exploration and exploitation in the proposed method, as shown in Algo. 2. When the node v is not a terminal node and within the computational budget, the algorithm may perform *TreeExpansion*(v) to choose a valid random action from the action set $\mathcal{A}$ and expands the resultant node to the search tree. The algorithm may also choose an existing child node randomly or choose the best child node of the current node, and repeat the process. The choice of best child (*BestChild*(v_0)) is based on the

Algorithm 2 TreePolicy

Input: A starting node, v; tree expansion rate ϵ_0 exploration rate ϵ_1;
Output: A resultant node, v'

```
while v is non-terminal and within search budget do
    if v is expandable and Random(0,1)< ε_0 then
        return TreeExpansion(v)
    else if Random(0,1) < ε_1 then
        v ← RandomChild(v)
    else
        v ← BestChild(v)
    end if
end while
v' ← v
return v'
```

weighted combination of average cost and best (minimum) cost of the Monte-Carlo simulation in the ϵ-greedy FTS, as shown in Eq. 3.

$$v' = \underset{v' \in \mathcal{V}}{\operatorname{argmin}}((1-\alpha)q_{v'} + \alpha \frac{c_{v'}}{N_{v'}}) \tag{3}$$

where $\mathcal{V}$ is the set of children node of v_0.

Algorithm 3 DefaultPolicy

Input: A node v with MDP state s;
Output: A simulation result, c

```
c ← 0
while coverage ratio not achieved do
    a ← ForwardGreedySearch(s)
    s, c ← UpdateByModel(s, a)
end while
return c
```

For the default policy (described in Algo. 3) used in the *Simulation* stage, instead of using random policy, we apply

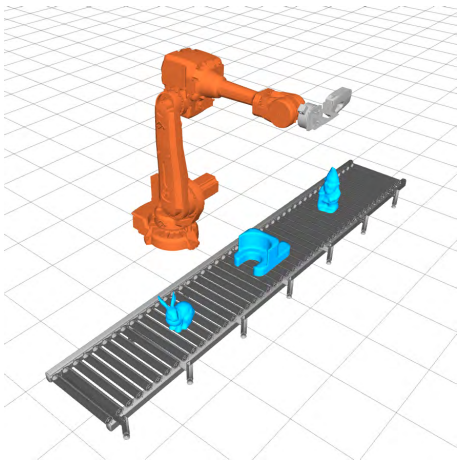


FIGURE 6. Experiment setup: the target objects (shown in light blue) are placed on the conveyor in front of a robotic manipulator, where a 3D scanner is mounted on the end-effector of the robot.

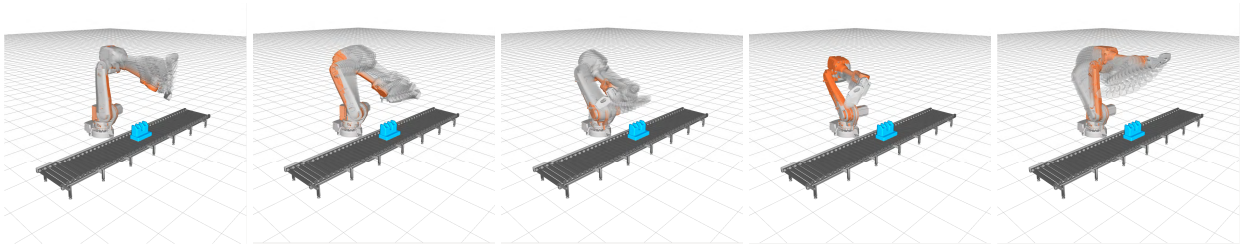
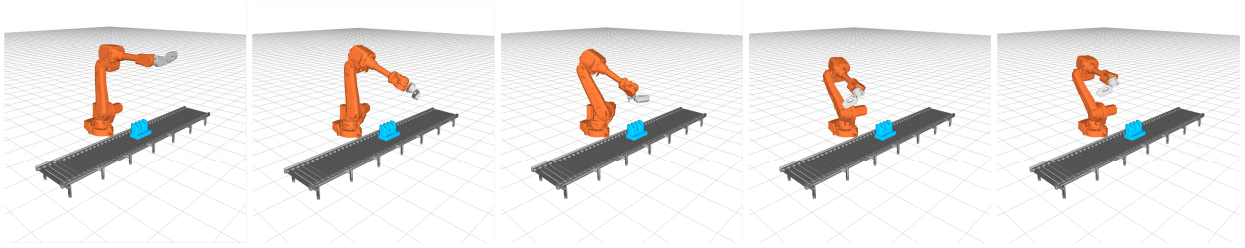


FIGURE 7. The computed inspection trajectory.



(a) (b) (c) (d) (e)

FIGURE 8. The computed inspection poses.

the greedy forward search algorithm. The method uses the visibility information of the estimated model, although the observed information might be imperfect due to positioning deviations. In the default policy, *ForwardGreedySearch(p)* returns an action that has the highest incremental coverage over the resultant cost by the action at the current state, as described in details in Eq. 4. *UpdateByModel(p, a)* is to update the coverage and state information in the *Simulation* step.

$$a = \underset{a \in \mathcal{A}}{\text{argmax}} \frac{\triangle cov(s, a)}{T_{ins}(s) + T_{travel}(s, \mathcal{T}(s, a))} \tag{4}$$

where $\triangle cov(s, a)$ is the incremental coverage by taking action a at state s, $T_{ins}(s)$ is the inspection cost and $T_{travel}(s, s')$ is the traveling cost between two states.

IV. COMPUTATIONAL EXPERIMENT

As shown in Fig. 6, the detailed experimental setting features a surface/shape inspection process on the factory production line. A robot manipulator is placed in front of the conveyor, with a 3D scanner mounted on its end-effector; the target workpiece is placed on the conveyor. The robot starts the inspection from its home pose, after completing the surface scan with a required coverage ratio, it returns back to the home pose in order to prepare for the inspection of the next target workpiece. For all the instances in the computational experiment, we sampled 500 candidate viewpoints around the target object. The computational experiments are conducted in the ROS environment [23], with an ABB IRB-4600 [34] industrial manipulator and a Artec Eva 3D scanner [35]. A discount factor $\lambda = 0.95$ is used across all the computational instances in this paper. Other sensor parameters are shown in the Table 1.

TABLE 1. Sensor parameters.

FOV	$30°, 21°$
Camera Pixels	536×371
Viewing Angle	$75°$
Viewing Range (meters)	$(0.4, 1.0)$

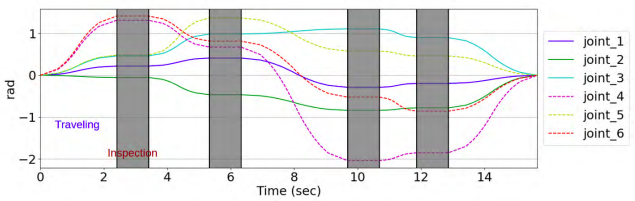


FIGURE 9. The resultant joint trajectories of the inspection process for Fig. 7: the x-axis is the time (measured in seconds), y-axis is the joint positions (measured in rads). For the colored region, the gray color is the time spent on inspection at the viewpoints, the white color part is time spent when traveling between the viewpoints.

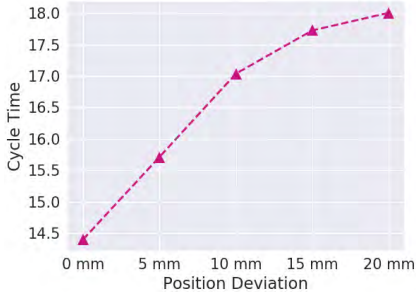
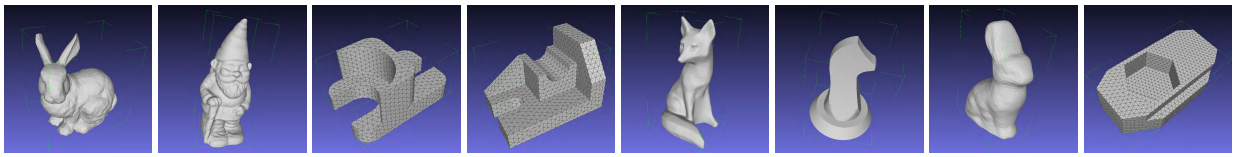


FIGURE 10. Average cycle time (measured in seconds) for different pose deviations.

We first start to apply the proposed computational framework to a mechanical target workpiece, as shown in Fig. 5a



(a) (b) (c) (d) (e) (f) (g) (h)

FIGURE 11. The polygonal geometric model of the target object used in this paper. (a) Bunny. (b) Gnome. (c) Mech. (d) Mech2. (e) Fox. (f) Chess. (g) Bunny2. (h) Mech3.

TABLE 2. Average cycle time for the planned robotic inspection task (measured in seconds).

	Bunny	Gnome	Mech	Mech2	Fox	Chess	Bunny2	Mech3
NBV-Greedy, 96%	41.5	32.6	33.3	35.0	31.6	16.6	30.0	22.7
ϵ-greedy FTS, 96%	29.3	28.8	24.4	31.6	20.4	14.4	21.0	19.3
Improvement, 96%	29.5%	11.7%	26.6%	9.8%	35.3%	13.1%	30.1%	15.0%
NBV-Greedy, 98%	56.7	44.6	39.2	46.1	31.6	25.7	32.4	30.1
ϵ-greedy FTS, 98%	34.7	33.1	29.4	36.8	26.5	14.5	27.8	21.4
Improvement, 98%	38.8%	25.9%	24.9%	20.2%	15.9%	43.5%	14.1%	29.1%

in previous section. We run the computational test under different pose variations, with a required coverage ratio of 96%, rotational deviation of 5° on z-axis. The average cycle time required (based on 10 trials) for the inspection with respect to different pose variation is shown in Fig. 10. The resultant robot inspection trajectories and poses of one inspection process are shown in Fig. 7 and Fig. 8. The robot starts the inspection from the its home pose (as shown in the Fig. 8 (a), then move to each desired inspection pose (Fig. 8 (b) to (e)) to conduct the measurement, then return back to the home pose to prepare for the next round of inspection. In this instance, four measurements are required to complete the inspection task, computed by the proposed method. The corresponding joint trajectories of the inspection process computed by the local planner are shown in Fig. 9.

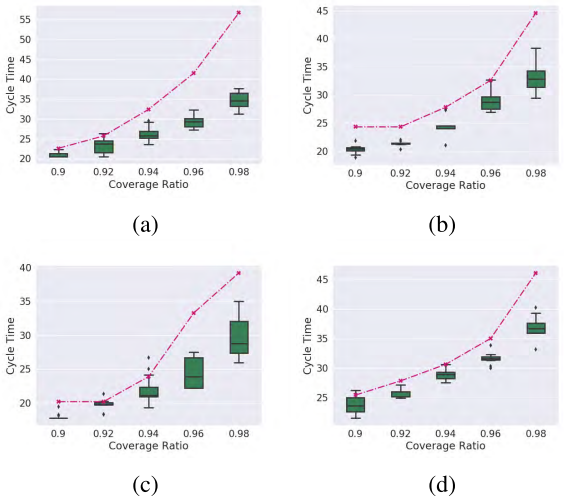


FIGURE 12. Cycle time comparison of ϵ-greedy FTS (shown in green) and NBV-greedy (shown in red) methods with different coverage ratio requirements. (a) Bunny. (b) Gnome. (c) Mech. (d) Mech2.

V. ADDTIONAL EXPERIMENTS AND RESULTS

In order to further validate the proposed framework and benchmark with the other methods, we also conduct several additional computational tests, and benchmark the proposed ϵ-greedy FTS method with the Next-Best-View (NBV) [5], [16] method, which is used as baseline in [2] and [36] with similar applications. The target workpieces with different sizes and shapes used are as shown in Fig. 11. In these computational instances, a position deviation of 15 *mm* in both x and y axis, and a rotation deviation of 5° in z-axis is used.

For the inspection application with considering uncertainties in positioning variation of the target workpiece, we use 96% and 98% as the requirements for the coverage ratios, which are similar and comparable to [6]. The actual pose of the target workpiece is assumed to be placed slightly away from its desired poses. Constant inspection time of 1.0 second is used for inspecting the target object at each viewpoints. As summarized in the Table 2, the proposed ϵ-greedy FTS method performs better than the traditional NBV-greedy framework with greedy search method in all experimental instances without any exception. The reduction of the inspection cycle time are ranged between 9.8% to 43.5%. Except the only instance of *mech* with 96% coverage requirement, double digit improvements are observed on all other experimental instances. The average cycle time reduction is 21.4% for 96% coverage ratio and 26.6% for 98% coverage ratio among all the instances.

Additionally, as shown in Fig. 12, we compare the performance of ϵ-greedy FTS method and NBV-greedy method over different coverage ratios range from 90% to 98%, on the first four target objects: *bunny*, *gnome*, *mech* and *mech2*. It is shown that the proposed ϵ-greedy FTS method (shown

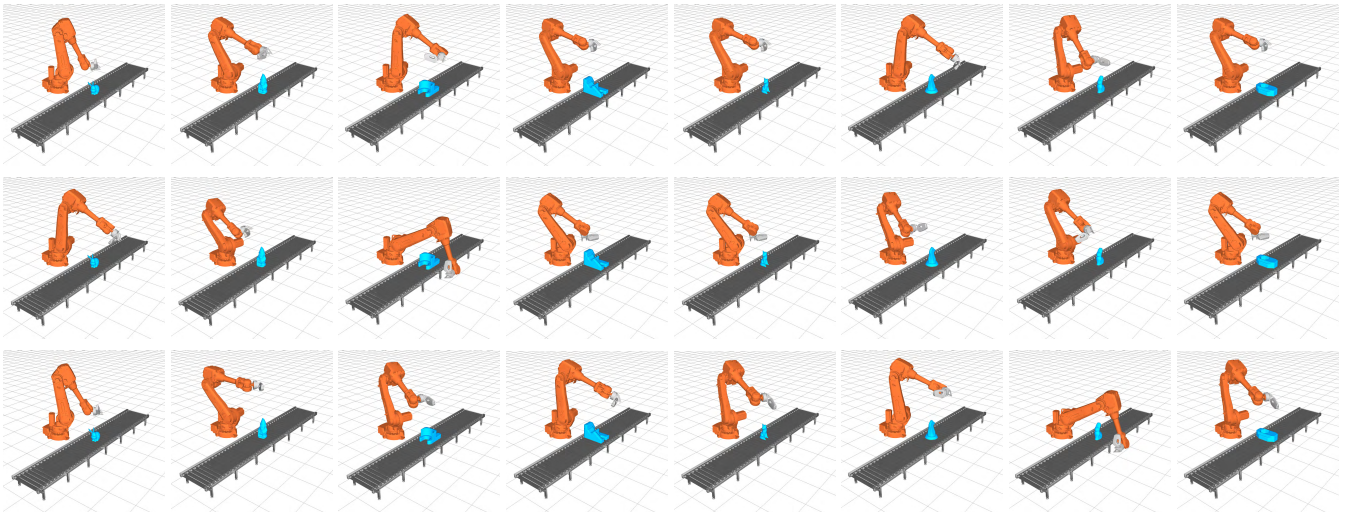


FIGURE 13. Example robotic poses for the surface/shape inspection tasks computed by the ϵ-greedy FTS method proposed in this paper, three inspection poses for each target object are randomly chosen to be visualized in this plot.

in green) constantly outperforms the NBV-greedy approach with different required coverage ratios over these models. The results of the NBV-Greedy method (shown in red) are deterministic when the environment is given. It is noticed that the improvement is higher when the required coverage ratio is higher in general, which is mainly due to the difficulties of covering the last few percentage of the surface area in the coverage planning problems, similar to the observations discovered in the pure VPP-SCP problem discussed in [21]. Some selected example robotic inspection poses of the planning results using the ϵ-greedy FTS method are shown in Fig. 13.

VI. DISCUSSIONS

The computational tests demonstrate that the proposed framework with MDP formulation and ϵ-greedy FTS search method is able to automatically generate efficient planning policy online for different target workpieces with different sizes and geometries robustly. In addition, the quantitative results show that the proposed ϵ-greedy FTS method performs better than the NBV approach, which is aligned with our expectations. There are several reasons. First, the NBV-greedy approach only considers the current best actions (similar to the local optima), but the overall actions might not be of good quality; on the contrast, the ϵ-greedy FTS performs simulations towards the end of episodes and exploring the state space to achieve better total return. Intuitively speaking, NBV-greedy approach only looks at one step ahead, while ϵ-greedy FTS method not only looks at many action steps ahead, but also tries to explore different actions based on the simulation results, in order to achieve better overall results. Moreover, another possible reason is that the model-based forward search approach helps to exclude the outliers by considering the average return, where the NBV method only chooses a local optimal viewpoint based on the estimated visibility.

VII. CONCLUSION AND FUTURE WORK

In this paper, we proposed a novel computational framework to automatically generate the inspection plan online for robotic inspection application. The proposed method utilizes a Markov Decision Process (MDP) formulation and a online policy search algorithm for Coverage Planning Problem (CPP) in the factory surface/shape inspection application with a robotic manipulator. The proposed framework is validated in several computational instances of different target workpieces with different geometries and under different conditions. In addition, the proposed framework is demonstrated to be able to handle the online coverage planning problem in the robotic inspection applications, with the presence of pose variation of the target workpiece. Moreover, within the proposed computational framework, the proposed ϵ-greedy FTS is able to outperform the previous NBV-based method for the inspection process. The maximum cycle time reduction is observed to be 43.5%, and the average cycle time reduction is observed to be 24.0% among these test instances.

For the future work, since we have proposed the overall computational framework of the online automatic path generation for robotic inspection application, we are interested in improving each sub-module of the computational framework to optimize the overall performance and robustness. More specifically, we would like to investigate different methods to better estimate the visibility model, such as conditional probabilistic visibility modeling method. Moreover, local planning and efficient viewpoints sampling methods also worth further research to improve computational efficiency and to explore the state space more efficiently. Additionally, we would also like to investigate more on the tree policy and the default simulation policy in the policy search algorithm to further improve the results, by adding search heuristics or using neural networks for value function approximation.

REFERENCES

[1] R. Raffaeli, M. Mengoni, M. Germani, and F. Mandorli, "Off-line view planning for the inspection of mechanical parts," *Int. J. Interact. Des. Manuf.*, vol. 7, no. 1, pp. 1–12, 2013.

[2] W. Jing, J. Polden, P. Y. Tao, C. F. Goh, W. Lin, and K. Shimada, "Model-based coverage motion planning for industrial 3D shape inspection applications," in *Proc. IEEE Conf. Autom. Sci. Eng.*, Aug. 2017, pp. 1293–1300.

[3] M. Mahmud, D. Joannic, and J.-F. Fontaine, "3D digitizing path planning for part inspection with laser scanning," in *Proc. 8th Int. Conf. Qual. Control Artif. Vis.* Bellingham, WA, USA: SPIE, 2007, p. 635603.

[4] A. Mavrinac, X. Chen, and J. L. Alarcon-Herrera, "Semiautomatic model-based view planning for active triangulation 3-D inspection systems," *IEEE/ASME Trans. Mechatronics*, vol. 20, no. 2, pp. 799–811, Apr. 2015.

[5] W. R. Scott, G. Roth, and J.-F. Rivest, "View planning for automated three-dimensional object reconstruction and inspection," *ACM Comput. Surv.*, vol. 35, no. 1, pp. 64–96, 2003.

[6] W. Jing, "Coverage planning for robotic vision applications in complex 3D environment," Ph.D. dissertation, Dept. Mech. Eng., Carnegie Mellon Univ., Pittsburgh, PA, USA, 2017.

[7] W. R. Scott, "Model-based view planning," *Mach. Vis. Appl.*, vol. 20, no. 1, pp. 47–69, 2009.

[8] E. Galceran and M. Carreras, "A survey on coverage path planning for robotics," *Robot. Auton. Syst.*, vol. 61, no. 12, pp. 1258–1276, 2013.

[9] P. Wang, R. Krishnamurti, and K. Gupta, "View planning problem with combined view and traveling cost," in *Proc. IEEE Int. Conf. Robot. Autom.*, Apr. 2007, pp. 711–716.

[10] W. Jing, J. Polden, W. Lin, and K. Shimada, "Sampling-based view planning for 3D visual coverage task with unmanned aerial vehicle," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2016, pp. 1808–1815.

[11] B. Englot and F. S. Hover, "Three-dimensional coverage planning for an underwater inspection robot," *Int. J. Robot. Res.*, vol. 32, nos. 9–10, pp. 1048–1073, 2013.

[12] I. Gentilini, K. Nagamatsu, and K. Shimada, "Cycle time based multi-goal path optimization for redundant robotic systems," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Nov. 2013, pp. 1786–1792.

[13] P. Wang, K. Gupta, and R. Krishnamurti, "Some complexity results for metric view planning problem with traveling cost and visibility range," *IEEE Trans. Autom. Sci. Eng.*, vol. 8, no. 3, pp. 654–659, Jul. 2011.

[14] S. Chen, Y. Li, and N. M. Kwok, "Active vision in robotic systems: A survey of recent developments," *Int. J. Robot. Res.*, vol. 30, no. 11, pp. 1343–1377, 2011.

[15] S. Isler, R. Sabzevari, J. Delmerico, and D. Scaramuzza, "An information gain formulation for active volumetric 3D reconstruction," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2016, pp. 3477–3484.

[16] C. Potthast and G. S. Sukhatme, "A probabilistic framework for next best view estimation in a cluttered environment," *J. Vis. Commun. Image Represent.*, vol. 25, no. 1, pp. 148–164, 2014.

[17] R. S. Sutton and A. G. Barto, *Reinforcement Learning—An Introduction*, vol. 1, no. 1. Cambridge, MA, USA: MIT Press, 1998.

[18] D. Silver *et al.*, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[19] S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," in *Proc. IEEE Int. Conf. Robot. Autom.*, May/Jun. 2017, pp. 3389–3396.

[20] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, Feb. 2015.

[21] M. D. Kaba, M. G. Uzunbas, and S. N. Lim, "A reinforcement learning approach to the view planning problem," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 6933–6941.

[22] R. Diankov and J. Kuffner, "OpenRAVE: A planning architecture for autonomous robotics," Robot. Inst., Carnegie Mellon Univ., Pittsburgh, PA, USA, Tech. Rep. CMU-RI-TR-08-34, 2008, vol. 79.

[23] M. Quigley *et al.*, "ROS: An open-source robot operating system," in *Proc. ICRA Workshop Open Source Softw.*, 2009, p. 5.

[24] J. Pan, S. Chitta, and D. Manocha, "FCL: A general purpose library for collision and proximity queries," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2012, pp. 3859–3866.

[25] S. M. LaValle, *Planning Algorithms*. Cambridge, U.K.: Cambridge Univ. Press, 2006.

[26] J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *Proc. IEEE Int. Conf. Robot. Autom.*, vol. 2, Apr. 2000, pp. 995–1001.

[27] I. A. Sucan and S. Chitta. (2013). *Moveit!* [Online]. Available: http://www.moveit.ros.org

[28] I. A. Sucan, M. Moll, and L. E. Kavraki, "The open motion planning library," *IEEE Robot. Autom. Mag.*, vol. 19, no. 4, pp. 72–82, Dec. 2012. [Online]. Available: http://www.ompl.kavrakilab.org

[29] S. Edwards and C. Lewis, "ROS-industrial: Applying the robot operating system (ROS) to industrial applications," in *Proc. IEEE Int. Conf. Robot. Autom., ECHORD Workshop*, May 2012.

[30] K. Shimada and D. C. Gossard, "Bubble mesh: Automated triangular meshing of non-manifold geometry by sphere packing," in *Proc. 3rd ACM Symp. Solid Modeling Appl.*, 1995, pp. 409–419.

[31] D. Silver *et al.*, "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, 2017.

[32] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *Int. J. Robot. Res.*, vol. 32, no. 11, pp. 1238–1274, 2013.

[33] C. B. Browne *et al.*, "A survey of Monte Carlo tree search methods," *IEEE Trans. Comput. Intell. AI in Games*, vol. 4, no. 1, pp. 1–43, Mar. 2012.

[34] ABB Group. *ABB IRB 4600 Industrial Robots*. Accessed: Jun. 2018. [Online]. Available: http://new.abb.com/products/robotics/industrial-robots/irb-4600

[35] Artec 3D. *Artec Eva 3D Scanner*. Accessed: Jun. 2018. [Online]. Available: https://www.artec3d.com/portable-3d-scanners/artec-eva

[36] W. Jing, J. Polden, C. F. Goh, M. Rajaraman, W. Lin, and K. Shimada, "Sampling-based coverage motion planning for industrial inspection application with redundant robotic system," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Sep. 2017, pp. 5211–5218.

WEI JING received the B.Eng. degree (Hons.) in electrical and electronic engineering from Nanyang Technological University, Singapore, in 2010, and the M.S. and Ph.D. degrees in mechanical engineering from Carnegie Mellon University, Pittsburgh, PA, USA, in 2014 and 2017, respectively. He is currently a Research Scientist with the Department of Computing Science, Institute of High Performance Computing, and also with the Artificial Intelligence Initiative, Agency for Science, Technology and Research, Singapore. His research interests include robotics, reinforcement learning, and task and motion planning.

CHUN FAN GOH received the B.Eng. degree in mechanical engineering from Nanyang Technological University, Singapore, the S.M. degree in computation for design and optimization from the Massachusetts Institute of Technology, and the M.Sc. degree from the National University of Singapore. He is currently pursuing the joint Ph.D. degree with the Department of Mechanical Engineering, Carnegie Mellon University, and the School of Mechanical and Aerospace Engineering, Nanyang Technological University. He was a Research Associate with the Centre of High-Performance Embedded System, Nanyang Technological University. He was responsible for the development of motion sensing and recognition capability of a tangible user interface. He currently focuses on applying machine learning techniques in wall-climbing robots, snore pattern detection, and control in non-stationary conditions. His research interests are in the areas of robotics, machine learning, and artificial intelligence.

MABARAN RAJARAMAN received the master's degree in mechanical engineering from Carnegie Mellon University in 2011, where he is currently pursuing the Ph.D. degree with the Mechanical Engineering Department. He focused on localization and industrial automation for known and unknown geometries of workpieces. His interests are in computer vision, machine learning, and probabilistic models.

FEI GAO received the B.S. and Ph.D. degrees in applied mathematics from Zhejiang University, China, in 2008 and 2013, respectively. He is currently a Research Scientist with the Department of Computing Science, Institute of High Performance Computing, and also with the Artificial Intelligence Initiative, Agency for Science, Technology and Research, Singapore. His research interests include cyber-physical systems, artificial intelligence, and robotics.

SOOHO PARK received the B.S. degree from the School of Mechanical and Aerospace Engineering, Seoul National University, Seoul, South Korea, in 2000, and the M.S. degree from the School of Mechanical Engineering, Carnegie Mellon University, Pittsburgh, PA, USA, in 2008, where he is currently pursuing the Ph.D. degree. He was a Control Software Engineer with automotive industry, South Korea. His research interests are in learning-based model identification, sequence learning, stochastic filtering, optimal control, and Bayesian reinforcement learning in robotic systems.

YONG LIU received the Ph.D. degree from the National University of Singapore. He was a Post-Doctoral Research Fellow with the Royal Swedish Academic of Science. He had both research and industrial experiences in machine learning, medical imaging, reinforcement learning, high performance computing, and network modeling and optimization. He was a PI for multiple projects on healthcare applications powered by artificial intelligence. He is currently a Senior Scientist with the Computing Science Department, IHPC, A*STAR. He focuses on artificial intelligence related research including deep learning and reinforcement learning, and develops new technologies and innovations to solve real world challenges, especially in the area of healthcare domain. He has co-authored two books. He was also a Team Member of the First Prize Winner of Rakuten TV Recommender Challenge in 2015. He received multiple awards and research grants from Singapore government agencies such as EDB and SPRING Singapore. One of his recent papers received the Best Paper Award at the Beyond Labeler Workshop on IJCAI 2016.

KENJI SHIMADA received the Ph.D. degree from the Massachusetts Institute of Technology in 1993. He received his Ph.D. from Massachusetts Institute of Technology in 1993, after which he returned to IBM Japan, commercialized BubbleMesh, which he invented in his Ph.D. thesis, and managed the Advanced Computer Graphics Group. He moved to the United States in 1996 to join the faculty of Carnegie Mellon University and served as an Assistant Professor, an Associate Professor, and a Professor in the Department of Mechanical Engineering, the Robotics Institute, the Department of Biomedical Engineering, and the Department of Civil and Environmental Engineering. He is currently a Theodore Ahrens Professor of engineering with Carnegie Mellon University.