Check for updates

Feature Guided Search for Creative Problem Solving Through Tool Construction

Lakshmi Nair * *and Sonia Chernova*

Georgia Institute of Technology, Atlanta, GA, United States

Robots in the real world should be able to adapt to unforeseen circumstances. Particularly in the context of tool use, robots may not have access to the tools they need for completing a task. In this paper, we focus on the problem of tool construction in the context of task planning. We seek to enable robots to construct replacements for missing tools using available objects, in order to complete the given task. We introduce the Feature Guided Search (FGS) algorithm that enables the application of existing heuristic search approaches in the context of task planning, to perform tool construction efficiently. FGS accounts for physical attributes of objects (e.g., shape, material) during the search for a valid task plan. Our results demonstrate that FGS significantly reduces the search effort over standard heuristic search approaches by $\approx 93\%$ for tool construction.

Keywords: tool construction, creative problem solving, task planning, heuristic search, adaptive robotic systems

1. INTRODUCTION

Humans often show remarkable improvisation capabilities, particularly in times of crises. The makeshift carbon dioxide filter constructed on board the Apollo 13 (Cass, 2005), and the jury-rigged ventilators built to combat equipment shortages during COVID-19 (Turner et al., 2020), are examples of human ingenuity in the face of uncertainty. In addition to humans, other primates and certain species of birds have also been shown to creatively accomplish tasks by constructing tools from available objects, such as sticks and stones (Jones and Kamil, 1973; Toth et al., 1993; Stout, 2011). While the capability to construct tools is often regarded as a hallmark of sophisticated intelligence, similar improvisation capabilities are currently beyond the scope of existing robotic systems. The ability to improvise and construct necessary tools can greatly increase robot adaptability to unforeseen circumstances, enabling robots to handle any uncertainties or equipment failures that may arise (Atkeson et al., 2018).

In this paper, we focus on the problem of tool construction in the context of task planning. Specifically, we address the scenario in which a robot is provided with a task that requires certain tools that are missing or unavailable. The robot must then derive a task plan that involves constructing an appropriate replacement tool from objects that are available to it, and use the constructed tool to accomplish the task. Existing work that addresses the problem of planning in the case of missing tools focuses on directly substituting the missing tool with available objects (Agostini et al., 2015; Boteanu et al., 2015; Nyga et al., 2018). In contrast, this is the first work to address the problem through the construction of replacement tools, by introducing a novel approach called Feature Guided Search (FGS). FGS enables efficient application of existing heuristic search algorithms in the context of task planning in order to perform tool construction by accounting for physical attributes of objects (e.g., shape, material) during the search for a valid task plan.

Heuristic search algorithms, such as A^* and enforced hill-climbing (EHC), have been successfully applied to planning problems in conjunction with heuristics such as cost-optimal landmarks (Karpas and Domshlak, 2009) and fast-forward (Hoffmann and Nebel, 2001), respectively. However, the application of heuristic search algorithms to perform tool construction in the context of task planning can be challenging. For example, consider a task where the goal of the robot is to hang a painting on the wall. In the absence of a hammer that is required for hammering a nail to complete the task, the robot may choose to construct a replacement for the hammer using the objects available to it. How does the robot know which objects should be combined to construct the replacement tool? One possible solution is for the user to manually encode the correct object combination in the goal definition, and the search procedure would find it. However, it is impractical for the user to know and encode the correct object combination to use, for all the objects that the robot could possibly encounter. Alternatively, the robot can autonomously attempt every possible object combination until it finds an appropriate tool construction for completing the task. However, this would require a prohibitive number of tool construction attempts. Further, what if the robot *cannot* construct a good replacement for a hammer using the available objects, but can instead construct a makeshift screwdriver to tighten a screw and complete the task? In this case, the task plan would also have to be adapted to appropriately use the constructed tool, i.e., "tighten" a screw with the screwdriver instead of "hammering" the nail. In order to address these challenges, FGS combines existing planning heuristics with a score that is computed from input point clouds of objects indicating the best object combination to use for constructing a replacement tool. The chosen replacement tool then in turn guides the correct action(s) to be executed for completing the task (e.g., "tighten" vs. "hammering"). Hence, our algorithm seeks to: (a) eliminate the need for the user to specify the correct object combination, thus enabling the robot to autonomously choose the right tool construction based on the available objects and the task goal, (b) minimize the number of failed tool construction attempts in finding the correct solution, and (c) adapt the task plan to appropriately use the constructed replacement tool.

Prior work by Nair et al. introduced a novel computational framework for performing tool construction, in which the approach takes an input action, e.g., "flip," in order to output a ranking of different object combinations for constructing a tool that can perform the specified action, e.g., constructing a spatula (Nair et al., 2019a,b). For performing the ranking, the approach scored object combinations based on the shape and material properties of the objects, and whether the objects could be attached appropriately to construct the desired tool. In contrast, this work focuses on the application of heuristic search algorithms such as A^*, to the problem of tool construction in the context of task planning. In this case, the robot is provided an input *task*, e.g., "make pancakes," that requires tools that are inaccessible to the robot, e.g., a missing spatula. The robot must then output a *task plan* for making pancakes, that involves constructing an appropriate replacement tool from available objects, and adapting the task plan to use the constructed tool

for completing the task. Thus, prior work takes an action as input, and outputs a ranking of object combinations. In contrast, our work takes a task as input, and outputs a task plan that involves constructing and using an appropriate replacement tool. Hence, our work relaxes a key assumption of the prior work that requires the input action to be specified. Our approach directly uses the score computation methodology described in prior work (Nair et al., 2019a,b), but combines it with planning heuristics to integrate tool construction within a task planning framework.

Our core contributions in this paper include:

- Introducing the Feature Guided Search (FGS) approach that integrates reasoning about physical attributes of objects with existing heuristic search algorithms for efficiently performing tool construction in the context of task planning.
- Improving upon prior work by enabling the robot to automatically choose the correct tool construction and the appropriate action based on the task and available objects, thus eliminating the need to explicitly specify an input action as assumed in prior work.

We evaluate our approach in comparison to standard heuristic search baselines, on the construction of six different tool types (hammer, screwdriver, ladle, spatula, rake, and squeegee), in three task domains (wood-working, cooking, and cleaning). Our results show that FGS outperforms the baselines by significantly reducing computational effort in terms of number of failed construction attempts. We also demonstrate the adaptability of the task plans generated by FGS based on the objects available in the environment, in terms of executing the correct action with the constructed tool.

2. RELATED WORK

Prior work by Sarathy and Scheutz have focused on formalizing creative problem solving in the context of planning problems (Sarathy and Scheutz, 2017, 2018). They define the notion of "Macgyver-esque" creativity as embodied agents that can "*generate, execute, and learn strategies for identifying and solving seemingly unsolvable real-world problems*" (Sarathy and Scheutz, 2017). They formalize Macgyvering problems (MGP) with respect to an agent t, as a planning problem in the agent's world $\mathbb{W}_t$, that has a goal state g currently unreachable by the agent. As described in their work, solving an MGP requires a domain extension or contraction through perceiving the agent's environment and self. Prior work by Sarathy also provide an in-depth discussion of the cognitive processes involved in creative problem solving in detail, by leveraging existing work in Neuroscience (Sarathy, 2018). Prior work by Olteţeanu and Falomir has also looked at the problem of Object Replacement and Object Composition (OROC) situated within a cognitive framework called, the Creative Cognitive Framework (CreaCogs) (Olteţeanu and Falomir, 2016). Their work utilizes a knowledge base that semantically encodes object properties and relationships in order to reason about alternative uses for objects to creatively solve problems. The semantic relationships themselves are currently encoded a-priori. Similar work by

Freedman et al. has focused on the integration of analogical reasoning and automated planning for creative problem solving by leveraging semantic relationships between objects (Freedman et al., 2020). They present the Creative Problem Solver (CPS), that uses large-scale knowledge bases to reason about alternate uses of objects for creative problem solving. In contrast to reasoning about objects, prior work by Gizzi et al. has looked at the problem of discovering new actions for creative problem solving, enabling the robot to identify previously unknown actions (Gizzi et al., 2019). Their work applies action segmentation and change-point detection to previously known actions to enable a robot to discover new actions. The authors then apply breadth-first search and depth-first search in order to derive planning solutions using the newly discovered actions.

In related work, Erdogan and Stilman (2013) described techniques for Automated Design of Functional Structures (ADFS), involving construction of navigational structures, e.g., stairs or bridges. They introduce a framework for effectively partitioning the solution space by inducing constraints on the design of the structures. Further, Tosun et al. (2018) have looked at planning for construction of functional structures by modular robots, focusing on identifying features that enable environment modification in order to make the terrain traversable. In similar work, Saboia et al. (2018) have looked at modification of unstructured environments using objects, to create ramps that enhance navigability. More recently, Choi et al. (2018) extended the cognitive architecture ICARUS to support the creation and use of functional structures such as ramps, in abstract planning scenarios. Their work focuses on using physical attributes of objects that is encoded a-priori, such as weight and size, in order to reason about the construction and stability of navigational structures. More broadly, these approaches are primarily focused on improving robot navigation through environment modification as opposed to construction of tools. Some existing research has also explored the construction of simple machines such as levers, using environmental objects (Levihn and Stilman, 2014; Stilman et al., 2014). Their work formulates the construction of simple machines as a constraint satisfaction problem where the constraints represent the relationships between the design components. The constraints in their work limit the variability of the simple machines that can be constructed, focusing only on the placement of components relative to one another, e.g., placing a plank over a stone to create a lever. Additionally, Wicaksono and Sheh (2017) have focused on using 3D printing to fabricate tools from polymers. Their work encodes the geometries of specific sub-parts of tools, and enables the robot to experiment with different configurations of the fabricated tools to evaluate their success for accomplishing a task.

The work described in this paper differs from the research described above in that we focus on creative problem solving through tool construction. Specifically, we focus on planning tasks in which the required tools need to be constructed from available objects. Two key aspects of our work that further distinguish it from existing research include: (i) sensing and reasoning about physical features of objects, such as shape, material, and the different ways in which objects can be attached, and (ii) improving the performance of heuristic search algorithms for tool construction in the context of task planning, by incorporating the physical properties of objects during the search for a task plan.

3. APPROACH

In this section, we begin by discussing some background details regarding heuristic search, followed by specific implementation details of FGS.

3.1. Heuristic Search

Heuristic search algorithms are guided by a cost function $f(s) = g(s) + h(s)$, where $g(s)$ is the best-known distance from the initial state to the state s, and $h(s)$ is a heuristic function that estimates the cost from s to the goal state. An admissible heuristic never overestimates the path cost from any state s to the goal (Hart et al., 1968; Zhang et al., 2009). A consistent heuristic holds the additional property that, if there is a path from a state x to a state y, then $h(x) \leq d(x, y) + h(y)$, where $d(x, y)$ is the distance from x to y (Hart et al., 1968). Most heuristic search algorithms, including A^*, operate by maintaining a priority queue of states to be expanded (the open list), sorted based on the cost function. At each step, the state with the least cost is chosen, expanded, and the successors are added to the open list. If a successor state is already visited, the search algorithm may choose to re-expand the state, *only if* the new path cost to the state is lesser than the previously found path cost (Bagchi and Mahanti, 1983). The search continues until the goal state is found, or the open list becomes empty, in which case no plan is returned.

3.2. Feature Guided Search

We now describe the implementation of FGS[1]. For the purposes of this explanation, we present our work in the context of A^*, though our approach can be easily extended to other heuristic search algorithms as demonstrated in our experiments. Let S denote the set of states, A denote the set of actions, γ denote state transitions, s_i denote the initial state, and s_g denote the goal state. For the planning task, we consider the problem to be specified in Planning Domain Definition Language (PDDL) (McDermott et al., 1998), consisting of a domain definition $\mathcal{P}_\mathcal{D} = (S, A, \gamma)$, and a problem/task definition $\mathcal{P}_\mathcal{T} = (\mathcal{P}_\mathcal{D}, s_i, s_g)$. Further, we use O to denote a set of n objects in the environment available for tool construction, $O = \{o_1, o_2, ...o_n\}$.

Since our work focuses on tools, we assume that some action(s) in A are parameterized by a set of object(s) $O_a \subseteq O$, that are used to perform the action. Specifically for tool construction, we explicitly define an action "join(O_a)", where $O_a = \{o_1, o_2, ...o_m\}, m \leq n$, parameterized by objects that can be joined to construct a tool for completing the task. For example, the action "join-hammer(O_a)" allows the robot to construct a hammer using the objects O_a that parameterize the action. For actions that are not parameterized by any object, $O_a = \emptyset$. Our approach seeks to assign a "feature score" to the objects in O_a,

---

[1] All source code including problem and domain definitions, are publicly available at: https://github.com/Lnair1993/Tool_Macgyvering.

Algorithm 1: Feature guided A^* search.

1 **Function** Search($\mathcal{P}_\mathcal{D}, \mathcal{P}_\mathcal{T}$, trust=true):
2 　s_i, s_g = extractStates($\mathcal{P}_\mathcal{T}$)
3 　A = extractActions($\mathcal{P}_\mathcal{D}$)
4 　O = extractObjects()
5 　O_{reject} = [ ]
6 　openList = [ ]
7 　setPathCost(s_i, 0)　　// *Set initial state's g(s) and f(s) to 0*
8 　openList. add(s_i, 0)
9 　**while** *OpenList* **not** *empty* **do**
10 　　currState = $argmin_s(f(s)) \; \forall \; s \in$ openList
11 　　openList. pop(currState)
12 　　**if** *currState* = s_g **then**
13 　　　**return** extractPlan(s_g, s_i)
14 　　nextStates = getNext(currState, A)
15 　　**for** $(s, a, O_a) \in$ *nextStates* **do**
16 　　　$g(s)$ = computePathCost(s, currState)　　// *Get current path cost*
17 　　　$c(s)$ = getPathCost(s)　　// *Get previous best known path cost*
18 　　　**if** $g(s) \geq c(s)$ **then**
19 　　　　**continue**
20 　　　**else**
21 　　　　setPathCost($s, g(s)$)　　// *Update lower costs as new paths are found*
22 　　　**end**
23 　　　$h(s)$ = computeHeuristic(s)
24 　　　$\phi(s)$ = **featureScore**(s, a, O_a, trust)　　// *Compute the feature score: Algorithm 2*
25 　　　**if** $\phi(s) = -\infty$ **then**
26 　　　　O_{reject}. add(O_a, a)　　// *Track rejected combinations*
27 　　　$f(s) = g(s) + h(s) - \phi(s)$
28 　　　**if** $f(s) = \infty$ **then**
29 　　　　**continue**
30 　　　openList. add($s, f(s)$)
31 　　**end**
32 　**end**
33 　**if** O_{reject} *not* $\emptyset$ **then**
34 　　Search($\mathcal{P}_\mathcal{D}, \mathcal{P}_\mathcal{T}$, trust = false)　　// *Re-attempt without trusting all sensors*
35 　**return** $\emptyset$　　　　　　　　// *No plan found*

---

indicating their fitness for performing the action a. Thus, given different sets of objects O_a that are valid parameterizations of a, the feature score can help guide the search to generate task plans that involve using the objects that are most appropriate for performing the action. In the context of tool construction, the feature score guides the search to generate task plans that involve joining the most appropriate objects for constructing the replacement tool, given the objects available in the environment. Feature scoring can also potentially reject objects that are unfit for tool construction.

Our approach is presented in Algorithm 1. The search algorithm extracts information regarding the initial state s_i and goal state s_g from the task definition (Line 2). The set of actions A is extracted from the domain definition $\mathcal{P}_\mathcal{D}$ (Line 3). The agent extracts the objects in its environment from an RGB-D observation of the scene through point cloud segmentation and clustering (Line 4). We initialize the open list (*openList*) as a priority queue with the initial state s_i and cost of 0 (Lines 6–8). Lines 9–32 proceed according to the standard A^* search algorithm, except for the computation of the feature score in Line 24. While the open list is not empty, we select the state with the lowest cost function (Line 10,11). If the goal is found, the plan is extracted (Lines 12–13), otherwise the successor states are generated (Line 14). For each successor state s, the algorithm computes the path cost $g(s)$ from the current state *currState* to s (Line 16). The algorithm then retrieves the best known path cost $c(s)$ for the state from its previous encounters (Line 17). If the state was not previously seen, $c(s) = \infty$. In Lines 18–22, the algorithm compares the best known path cost to the current path cost, and updates the best known path cost if $g(s) < c(s)$. The algorithm then computes the heuristic $h(s)$ (Line 23), and the feature score $\phi(s)$ (Line 24). The algorithm also maintains a list of object combinations that were rejected by feature scoring (i.e., assigned a score of $-\infty$), in O_{reject} (Line 26). The final cost is computed as $f(s) = g(s) + h(s) - \phi(s)$ (Line 27; We expand more on our choice of cost function in section 3.5). If $f(s) \neq \infty$, then the state is added to the open list, prioritized by the cost. The search continues until a plan is found, or exits if the open list becomes empty. If no plan was found, the search is reattempted (Line 34) by modifying the feature score computation (described in section 3.3). If all search attempts fail, the planner returns a failure with no plan found. In the following section we discuss the computation of the feature score in detail.

3.3. Feature Score Computation

In this section, we describe the computation of the feature score for a given set of objects O_a that parameterize an action a. Note that, in this work the feature score computation focuses on the problem of tool construction. However, FGS can potentially be extended to other problems such as tool substitution, by computing similar feature scores as described in prior work (Abelha et al., 2016; Shrivatsav et al., 2019). Given n objects, tool construction presents a challenging combinatorial problem with a state space of size ^nP_m, assuming that we wish to construct a tool with $m \leq n$ objects. Thus, $O_a = \{o_1, o_2, ...o_m\}$ denotes a specific permutation of m objects. Inspired by existing tool-making studies in humans (Beck et al., 2011), prior work introduced a multi-objective function for evaluating the fitness of objects for tool construction (Nair et al., 2019b), that we apply in this work for feature scoring. The proposed multi-objective function included three considerations: (a) shape fitness of the objects for performing the action, (b) material fitness of the objects for performing the action, and (c) evaluating whether the objects in O_a can be attached to construct the tool.

The calculation of each of the three metrics above relies on real-world sensing, which can be noisy. This can result in false negative predictions, that eliminate potentially valid object

combinations from consideration. In particular, prior work has shown that false negatives in material and attachment predictions have caused ≈ 4% of tool constructions to fail (Nair et al., 2019b). To address the problem of false negatives in material and attachment predictions, we introduce the notion of "sensor trust" in this work. Prior work that has looked at accounting for sensor trust has introduced the notion of "trust weighting" to use continuous values to appropriately weigh the sensor inputs (Pierson and Schwager, 2016). In contrast, the sensor trust parameter in our work is a *binary value* that determines whether the material and attachment predictions should be believed by the robot and included in the feature score computation. This is because material and attachment scores are hard constraints and not continuous, i.e., they are $-\infty$ for objects that are not suited for tool construction (we describe this further in later sections). Hence, a continuous weighting on the material and attachment scores is not appropriate for our work.

Our feature score computation approach is described in Algorithm 2. For actions that are not parameterized by objects, the approach returns 0 (Lines 2-3). If the trust parameter is set to *true*, the feature score computation incorporates shape, material, and attachment predictions. (Lines 5–12 of Algorithm 2; section 3.4.1 for details). If the trust parameter is set to *false*, the feature score computation only includes shape scoring (Lines 14–19 of Algorithm 2; section 3.4.2 for details). Thus, we describe two modes of feature score computation that is influenced by the sensor trust parameter. In the following sections, we briefly describe the computation of shape, material and attachment predictions, and for a more detailed implementation of each method, we refer the reader to Nair et al. (2019b) and Shrivatsav et al. (2019).

3.3.1. Shape Scoring

Shape scoring seeks to predict the shape fitness of the objects in O_a for performing the action a. This is indicated by the *ShapeFit*() function in Algorithm 2. In this work, we consider tools to have action parts and grasp parts[2]. Thus, $m = 2$ and the set of objects O_a consists of two objects, i.e., $|O_a| = 2$. Further, the ordering of objects in O_a indicates the correspondence of the objects to the action and grasp parts.

For shape scoring, we seek to train models that can predict whether an input point cloud is suited for performing a specific action. We formulate this as a binary classification problem. We represent the shape of the input object point clouds using Ensemble of Shape Functions (ESF) (Wohlkinger and Vincze, 2011) which is a 640-D vector, shown to perform well in representing object shapes for tools (Schoeler and Wörgötter, 2015; Nair et al., 2019b). We then train independent neural networks that take an input ESF feature, and output a binary label indicating whether the input shape feature is suited for performing a specific action. Thus, we train separate neural networks, one for each action[3]. More specifically, we train

---

[2]As in prior work, this covers the vast majority of tools (Myers et al., 2015; Abelha and Guerin, 2017).

[3]The advantage of the binary classification is that for new actions, additional networks can be trained independently without affecting other networks.

Algorithm 2: Feature score Computation

1 **Function** FeatureScore($s, a, O_a, trust = true$):
2   **if** O_a **is empty then**
3   **return** 0
4   **if** *trust* **then**
5   **if** $canAttach(O_a, a)$ **then**
6         $\phi^s_{shape}(O_a) = \text{ShapeFit}(O_a, a)$ // *Sensors are fully trusted - section 3.4.1*
7         $\phi^s_{mat}(O_a) = \text{MaterialFit}(O_a, a)$
8         $\phi(s) = \lambda_1 * \phi^s_{shape}(O_a) + \lambda_2 * \phi^s_{mat}(O_a)$ // *The weighted sum is assigned to s*
9   **return** $\phi(s)$
10   **else**
11   **return** $-\infty$
12   **end**
13   **else**
14   **if** $(O_a, a) \in O_{reject}$ **then**
15        $\phi^s_{shape}(O_a) = \text{ShapeFit}(O_a, a)$ // *Not fully trust sensors - section 3.4.2*
16   **return** $\phi^s_{shape}(O_a)$ // *Evaluate objects that were previously rejected*
17   **else**
18   **return** $-\infty$
19   **end**
20   **end**

---

separate networks for the tools' action parts, e.g., the head of a hammer or the flat head of a spatula, and for a supporting function: "*Handle*," which refers to the tools' grasp part, e.g., hammer handle.

For the score prediction, given a set of objects O_a to be used for constructing the tool, let $\mathcal{K}$ denote the set of objects in O_a that are candidates for the action parts of the final tool, and let $O_a - \mathcal{K}$ be the set of candidate grasp parts. Then the shape score $\phi^s_{shape}(O_a)$ is computed by using the trained networks as:

$$\phi^s_{shape}(O_a) = \prod_{o_i \in \mathcal{K}} p(action|o_i) \prod_{o_i \in O_a - \mathcal{K}} p(handle|o_i) \quad (1)$$

Where, p is the prediction confidence of the corresponding network. Thus, we combine prediction confidences for all action parts and grasp parts. For example, for the action "join-hammer(O_a)" where O_a consists of two objects (o_1, o_2), the shape score $\phi^s_{shape}(O_a) = p(hammer\_head|o_1) * p(handle|o_2)$.

3.3.2. Material Scoring

Material scoring seeks to predict the material fitness of the objects in O_a for performing the action a. This is indicated by the *MaterialFit*() function in Algorithm 2. In this work, we make three simplifying assumptions. Firstly, we consider the construction of rigid tools which covers a vast range of real-world examples (Myers et al., 2015; Abelha et al., 2016). Secondly, we consider the material properties of the action parts of the

TABLE 1 | Table indicating appropriate materials for action parts of different tools.

Tool	Material (Action part)
Hammer	Metal, Wood
Screwdriver	Plastic, Metal
Ladle	Plastic, Wood, Metal
Spatula	Plastic, Wood, Metal
Rake	Plastic, Wood, Metal
Squeegee	Foam

tool since the action parts are more critical to performing the action with the tool (Shrivatsav et al., 2019). Lastly, we assume that the materials that are appropriate for different tools is provided a-priori, e.g., hammer heads are made of wood or metal (Shown in **Table 1**). Note that this information can also be obtained using common knowledge bases such as RoboCSE (Daruna et al., 2019).

For material scoring, we seek to train models that can predict whether an input material is suited for performing a specific action. We represent the material properties of the object using spectral readings, since it has been shown to work well for material classification problems in prior work (Erickson et al., 2019, 2020; Shrivatsav et al., 2019). For extracting the spectral readings, the robot uses a commercially available handheld spectrometer[4], called SCiO, to measure the reflected intensities of different wavelengths, in order to profile and classify object materials. The spectrometer generates a 331-D real-valued vector of spectral intensities. Then given a dataset of SCiO measurements from an assortment of objects, we train a model through supervised learning to output a class label indicating the material of the object.

For the material score prediction, given the spectral readings for the action parts in O_a denoted by $\mathcal{K}$, we map the predicted class label to values in **Table 1** to compute the material score using the prediction confidence of the model. Let $T(a)$ denote the set of appropriate materials for performing an action a. Then the material score is computed as:

$$\phi^s_{mat}(O_a) = \begin{cases} z = \prod_{o_i \in \mathcal{K}} \max_{c_i \in T(a)} p(c_i|o_i), & \text{if } z \geq t \\ -\infty, & \text{otherwise} \end{cases} \quad (2)$$

Where, p is the prediction confidence of the network regarding the class c_i. We compute the max prediction confidence across all the appropriate classes $c_i \in T(a)$, and their product over the action parts in $\mathcal{K}$. For example, for the action "join-hammer(O_a)," where O_a consists of two objects (o_1, o_2), the material score $\phi^s_{mat}(O_a) = \max(p(metal|o_1), p(wood|o_1))$. If the max value exceeds some threshold[5] denoted by t, then the corresponding value is returned. Otherwise, the model returns $-\infty$. Hence, note that material prediction acts as a hard constraint, by directly

---

[4]https://www.consumerphysics.com/ - Note that SCiO can be controlled via an app that enables easy scanning of objects. The robot simply moves the scanner over the object, and the user presses a key within the app to scan the object.
[5]We empirically determined a threshold of 0.6 to work well.

eliminating any objects that are made of inappropriate materials, thus reducing the potential search effort.

3.3.3. Attachment Prediction
Given a set of objects, we seek to predict whether the objects can be attached to construct a tool. This is indicated by the *canAttach*() function in Algorithm 2. In order to attach the objects, we consider three attachment types for creating fixed attachments, namely, *pierce attachment* (piercing one object with another, e.g., foam pierced with a screwdriver), *grasp attachment* (grasping one object with another, e.g., a coin grasped with pliers), and *magnetic attachment* (attaching objects via magnets on them). For magnetic attachments, we manually specify whether magnets are present on the objects, enabling them to be attached. For pierce and grasp attachment, we check whether the attachments are possible as described below. If no attachments are possible for the given set of objects, the feature score returns $-\infty$, indicating that the objects are not a viable combination. Thus, the search eliminates any objects that cannot be attached.

3.3.3.1. Pierce attachment
Similar to material reasoning, we use the SCiO sensor to reason about material pierceability. We assume homogeneity of materials, i.e., if an object is pierceable, it is uniformly pierceable throughout the object. We train a neural network to output a binary label indicating pierceability of the input spectral reading (Nair et al., 2019b). If the model outputs 0, the objects cannot be attached via piercing.

3.3.3.2. Grasp attachment
To predict grasp attachment, we model the grasping tool (pliers or tongs) as an extended robot gripper. This allows the use of existing robot grasp sampling approaches (Zech and Piater, 2016; ten Pas et al., 2017; Levine et al., 2018), for computing locations where a given object can be grasped. In particular, we use the approach discussed by ten Pas et al., that outputs a set of grasp locations given the input parameters reflecting the attributes of the pliers or tongs used for grasping (ten Pas et al., 2017). If the approach could not sample any grasp locations, the objects cannot be attached via grasping.

3.4. Incorporating the Sensor Trust Parameter
In this section, we describe how the sensor trust parameter (Line 4, Algorithm 2) is incorporated to compute the feature score in two different ways. The first approach includes trusting the shape, material, and attachment predictions of the models described above. The second approach allows the robot to deal with possible false negatives in material and attachment predictions, by only incorporating the shape score into the feature score computation.

3.4.1. Fully Trust Sensors
In the case that the robot fully trusts the material and attachment predictions, the trust parameter is set to *true* (Line 4, Algorithm 2). The final feature score is then computed as a weighted sum of the shape and material scores, if the objects can be attached (Algorithm 2, Lines 5–8). We found uniform weights of $\lambda_1 = 1, \lambda_2 = 1$, to work well for tool constructions. If the objects cannot be attached, then $\phi(s) = -\infty$, indicating that the

objects in O_a do not form a valid combination. Otherwise, using Equations (1) and (2):

$$score(s, O_a) = \lambda_1 * \phi_{shape}^s(O_a) + \lambda_2 * \phi_{mat}^s(O_a) \qquad (3)$$

Since both material and attachment predictions are hard constraints, certain object combinations can be assigned a score of $-\infty$, indicating that the robot does not attempt these constructions. As described before, this can lead to cases of false negatives where the robot is unable to find the correct construction due to incorrect material or attachment predictions. In these cases, the algorithm tracks the rejected object combinations in O_{reject} (Algorithm 1, Line 26), and repeats the search as described below, by switching trust to *false* (Algorithm 1, Lines 33–34).

3.4.2. Not Fully Trust Sensors
In case of false negatives, the robot can choose to eliminate the hard constraints of material and attachment prediction from the feature score computation, thus allowing the robot to explore the initially rejected object combinations by using only the shape score. This is achieved by setting the trust flag to *false* in our implementation (Lines 14–15, Algorithm 2). In this case, we attempt to re-plan using the feature score as:

$$\phi(s) = \begin{cases} \phi_{shape}^s(O_a), & \text{if } O_a \subseteq O_{reject} \\ -\infty, & \text{otherwise} \end{cases} \qquad (4)$$

Here, O_{reject} indicates the set of objects that were initially rejected by the material and/or the attachment predictions. Since, shape score is a soft constraint, i.e., it does not eliminate any object combinations completely, we use the shape score to guide the search in case of the rejected objects. In the worst case, this causes the robot to explore all ^nP_m permutations of objects. However, as shown in our results, shape score can serve as a useful guide for improving tool construction performance in practice, when compared to naively exploring all possible object combinations. The final feature score computation, influenced by attachments and the trust parameter, can be summarized as follows from Equations (3), (4):

$$\phi(s) = \begin{cases} score(s, O_a), & \text{if attachable \& trust} \\ \phi_{shape}^s(O_a), & \text{if } \textbf{not} \text{ trust \& } O_a \subseteq O_{reject}. \\ -\infty, & \text{otherwise} \end{cases}$$

3.5. Final Cost Computation
Once the feature score is computed, the final cost function is computed as $f(s) = g(s) + h(s) - \lambda * \phi(s)$. Interestingly, we found that $\lambda = 1$, thus $f(s) = g(s) + h(s) - \phi(s)$, performs very well with the choice of search algorithms and heuristics in this work for the problem of tool construction. In this case, the higher the feature score $\phi(s)$, the lower the cost $f(s)$, in turn guiding the search to choose nodes with higher feature score (lower $f(s)$ values). Additionally, the values of the feature score are within the range $0 \leq \phi(s) \leq 2$. Since we use existing planning heuristics that have been shown to work well, and the task plans generated have $\gg 2$ steps involved, $g(s) + h(s) \gg 2$ and thus, $f(s) > 0$. Thus, $\lambda = 1$

works well for the problems described in this work. However, this presents an interesting research question for our future work in terms of an in-depth analysis of the choice of heuristic and feature score values, and its influence on the guarantees of the search.

3.6. Implementation Details
In this section, we describe additional details regarding the implementation of the work, both in terms of the algorithm, as well as the physical implementation on the robot.

3.6.1. Algorithm Implementation
In terms of implementation, the process begins with the trust parameter set to *True*. FGS generates a task plan that involves combining objects to construct the required tool. Once a task plan is successfully found, the robot can proceed with executing the task plan and joining the parts indicated by O_a as described in Nair et al. (2019b), to construct the required tool for completing the task. If the tool could not be successfully constructed or used, the plan execution is said to have failed, and the robot re-plans to generate a new task plan with a different object combination, since the algorithm tracks the attempted object combinations. Note that the approach also keeps track of object combinations rejected by material and attachment predictions in O_{reject}. If no solution could be found with trust set to *True*, and $O_{reject} \neq \emptyset$, then the robot switches trust to *false*, and FGS explores the object combinations within O_{reject} (Lines 33-34 of Algorithm 1). If no solution could be found with either trust setting, FGS returns a complete failure and terminates.

Further note that in this work, we do not explicitly deal with symbol grounding (Harnad, 1990) and symbol anchoring (Coradeschi and Saffiotti, 2003) problems. We overcome these issues by manually mapping the object point clouds to their specific symbols within the planning domain definition. Once the task plan is generated, the mapping is then used to match the symbols within the task plan to their corresponding objects in the physical world, via their point clouds. However, existing approaches can potentially be adapted in order to refine the symbol grounding functions (Hiraoka et al., 2018), or to enable the robot to automatically extract the relationships between the object point clouds and their abstract symbolic representations (Konidaris et al., 2018).

3.6.2. Physical Implementation
The spectrometer used in this work can be activated either using a physical button located on the sensor, or through an app that is provided with the sensor. However, pressing the physical button requires precision and careful application of the correct amount of force, which can be challenging for the robot since it may potentially damage the sensor if the applied force exceeds a certain threshold. To prevent this, in our implementation, the robot simply moves the scanner over the objects, and the user then manually presses a key within the app to activate the sensor. Additionally, the rate of scanning is also limited by the speed of the robot arm itself. Since the robot arm used in this work moves rather slowly, it took about ≈ 1.7 min on average to scan 10 objects, while this would take <30 s for a human. Overcoming these issues and several other manipulation

challenges are essential to ensure practical applicability of this work. We discuss this in more detail in section 5.

4. EXPERIMENTAL VALIDATION AND RESULTS

In this section, we describe our experimental setup and present our results alongside each evaluation. We validate our approach on three diverse types of tasks involving tool construction in a household domain, namely, wood-working, cooking, and cleaning. For wood-working tasks, the tools to be constructed include hammer and screwdriver; for cooking tasks the tools include spatula and ladle; and lastly for cleaning tasks the tools include rake and squeegee. Each tool is constructed from two parts ($m = 2$) corresponding to the action and grasp parts of the tool[6]. Our experiments seek to validate the following three aspects:

1. **Performance of feature guided A^* against baselines:** In order to investigate the informativeness of including feature score in heuristic search, we evaluate the feature guided A^* approach against three baselines. We also evaluate our approach in terms of the two different settings of the sensor trust parameter to investigate the benefits of introducing sensor trust.

2. **Combining feature scoring with other heuristic search algorithms:** To investigate whether feature scoring can generalize to other search approaches, we integrate feature scoring with two additional heuristic search algorithms. Specifically, we present results combining feature scoring with weighted A^* and enforced hill-climbing with the fast-forward heuristic (Hoffmann and Nebel, 2001).

3. **Adaptability of task plans to objects in the robot's environment:** We evaluate whether the robot can adapt its task plans to appropriately use the constructed tool, as the objects available to the robot for tool construction are modified. This measures whether the robot can flexibly generate task plans in response to the objects in the environment.

For all our experiments, we use a test set consisting of 58 previously unseen candidate objects for tool construction (shown in **Figure 1**). These objects consist of metal (11/58), wood (12/58), plastic (19/58), paper (2/58), and foam (14/58) objects. Only the foam and paper objects are pierceable. Prior to planning, the robot scans the materials of the objects for material scoring and attachment predictions. For our results, we evaluate the statistical significance where it is applicable, using repeated measures ANOVA with *post-hoc* Tukey's test. We discuss each experiment in more detail below, along with the results for each.

4.1. Performance of Feature Guided A*

In this section, we evaluate the performance of feature guided A^* against three baselines: (i) standard A^*, where $f(s) = g(s) + h(s)$, (ii) feature guided uniform cost search, where $f(s) = g(s) + 2.0 -$

$\phi(s)$, and (iii) standard uniform cost search, where $f(s) = g(s)$. In (ii), we use $2.0 - \phi(s)$ to add a positive value to $g(s)$ since, $0 \leq \phi(s) \leq 2$. As a heuristic with A^*, we use the cost optimal landmark heuristic (Karpas and Domshlak, 2009). We also vary the sensor trust parameter, and present results for the two cases where the robot is not allowed to change the trust parameter (trust always set to *true*, i.e., lines 33–34 of Algorithm 1 not executed), and for the case where the robot is allowed to change it to *false* when no plan is found.

For the evaluation, we create six different tasks, two tasks each for wood-working, cooking and cleaning. Each task requires the construction of *one* specific tool for its completion, e.g., one of the tasks in wood-working requires construction of a hammer, and the other requires construction of a screwdriver. For each task we created 10 test cases, where each test case consisted of 10 objects chosen from the 58 in **Figure 1**, that could potentially be combined to construct the required tool. We report the average results across the test cases for each task type (total 10 $\times$ 2 cases per task type with 10 candidate objects per case). We create each test set by choosing a random set of objects, ensuring that only one "correct" combination of objects exists per set. The correct combinations are determined based on human assessment of the objects. For each task, we instantiate the corresponding domain and problem definitions in PDDL[7].

The metrics used in this experiment include (i) the *number of nodes expanded* during search as a measure of computational resources consumed, (ii) the *number of failed construction attempts* before a working tool was found (also referred to as "attempts" in this paper), and (iii) the *success rate* indicating the number of times the robot successfully found a working tool. Ideally, we would like the number of nodes expanded and the number of failed construction attempts to be as low as possible. Note that the brute force number of failed construction attempts for 10 objects is 89, since there are $^{10}P_2$ possible object permutations for $m = 2$, with 89 incorrect possibilities. Ideally, we would like the number of failed construction attempts to be 0. The success rate should be as high as possible, ideally equal to 100%.

Table 2 shows the performance of feature guided A^* (where $f(s) = g(s) + h(s) - \phi(s)$, denoted by "FS+H") compared to the different baselines: "H" denotes standard A^* (where $f(s) = g(s) + h(s)$), "FS" denotes feature guided uniform cost search (where $f(s) = g(s) + 2.0 - \phi(s)$), and "UCS" denotes standard uniform cost search (where $f(s) = g(s)$). The values reported per task are the average performances across the test cases where tool constructions were successful. As shown in **Table 2**, incorporating feature scoring (FS, FS+H) helps significantly reduce the number of failed construction attempts compared to the baselines without feature scoring (H, UCS), with $p < 0.001$. Since heuristics can help reduce the search effort in terms of number of nodes expanded, we see that approaches that do not use heuristics (FS and UCS) expand significantly more

---

[6]In this work, we pre-specify the trajectories to be followed when combining the objects to construct the tool.

[7]In the planning problem definition, the objects are instantiated numerically through "obj0" to "obj9," where each literal is manually assigned to one of the 10 objects during planning time. Our planning and domain definitions are available at: https://github.com/Lnair1993/Tool_Macgyvering.

FIGURE 1 | Dataset of 58 objects used for the experiments, made of different materials.

TABLE 2 | Table comparing feature guided *A** ("FS+H") with baselines.

	Cleaning				Cooking				Wood-working			
	FS+H	**H**	**FS**	**UCS**	**FS+H**	**H**	**FS**	**UCS**	**FS+H**	**H**	**FS**	**UCS**
# Nodes	**5187**	**5187**	9061	9061	**329**	604	36237	36213	7264	**6936**	28606	28734
# Failed attempts	**2**	46	3	49	**3**	48	4	40	**2**	45	**2**	37

The other notations: "H"—standard A*; "FS"— feature guided uniform cost search; "UCS"—standard uniform cost search. This table reports the average number of failed attempts per task, across test cases where tool construction was successful. Note that the max number of failed attempts possible is 89 (brute force). Bold highlights the best performance values for the different metrics.
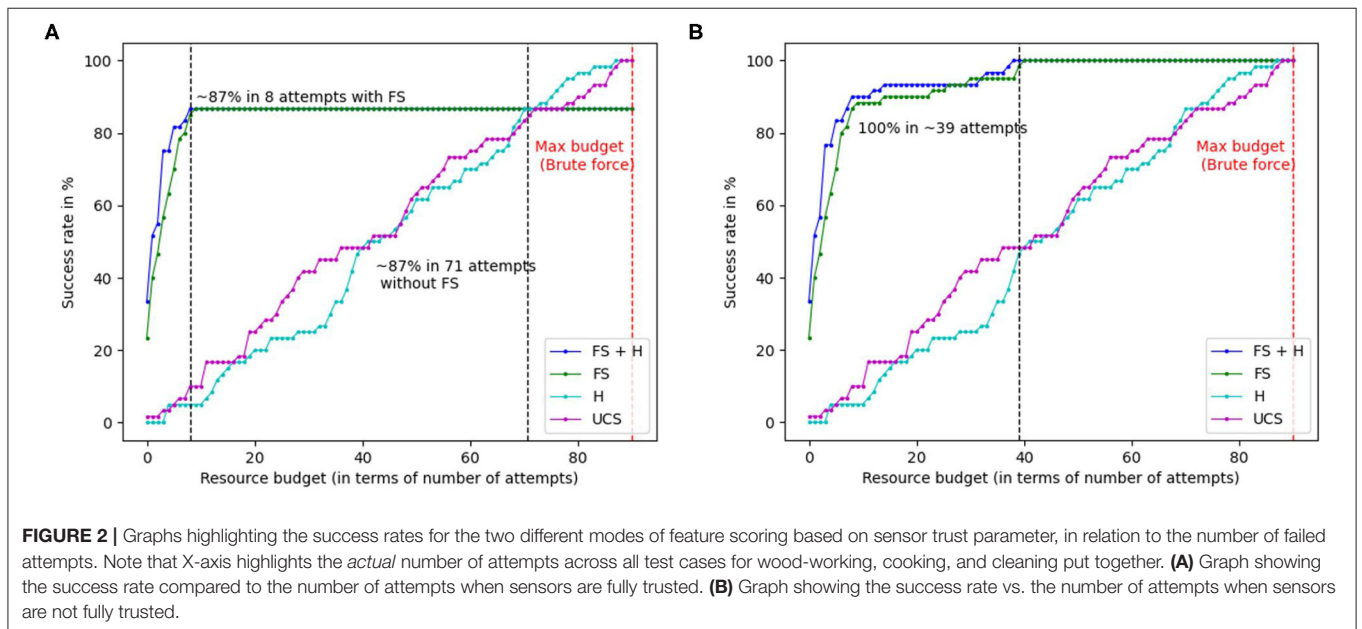
FIGURE 2 | Graphs highlighting the success rates for the two different modes of feature scoring based on sensor trust parameter, in relation to the number of failed attempts. Note that X-axis highlights the *actual* number of attempts across all test cases for wood-working, cooking, and cleaning put together. **(A)** Graph showing the success rate compared to the number of attempts when sensors are fully trusted. **(B)** Graph showing the success rate vs. the number of attempts when sensors are not fully trusted.

nodes than FS+H and H, with $p < 0.001$. Note that there is no statistically significant difference in the number of nodes expanded between H and FS+H. Thus, using feature scoring with heuristics (FS+H) yields the best performance in terms of *both* number of nodes expanded, and the number of failed construction attempts. To summarize, these results show that feature scoring is informative to heuristic search by significantly reducing the average number of failed construction attempts to ≈ 2 compared to ≈ 46 without it (brute force number of failed attempts is 89).

Further, in **Figures 2A,B**, we plot the success rate vs. the resource budget of the robot in terms of the *permissible* number of failed attempts. That is, the robot is not allowed to try any more than a fixed number of attempts, indicated by the resource budget. **Figure 2A** considers the case where the sensor trust parameter is always set to *true*, and **Figure 2B** considers the case when the robot is allowed to switch the trust to *false*, if a solution was not found. Note that in contrast to **Table 2**, the graphs report *actual* number of failed attempts, *across all tasks*, whereas **Table 2** reports the *average* number of failed attempts across the test cases per task, for tool constructions that were successful. In **Figure 2A**, we see that FGS (FS+H and FS) achieves a success rate of 86.67% (52/60 constructions) within a resource budget of ≈ 8 failed attempts to do so. This indicates that 13.33% of the valid constructions were treated as false negatives by material and attachment predictions, and were completely removed from consideration (unattempted). Thus, increasing the permissible resource budget beyond 8, does not make any difference. Without feature scoring, H and UCS achieve a success rate of 87% with a budget of 71 attempts, and 100% after exploring nearly every possible construction (max resource budget of 89 failed attempts). In contrast, when the robot is allowed to switch the trust parameter, the robot uses shape scoring alone to continue guiding the search. As shown in **Figure 2B**, FGS (FS+H and FS) achieves 100% success rate within a budget of ≈ 39 attempts, since the robot does not eliminate any object combinations from consideration. The performance is also significantly better than the baselines that do not use feature scoring. This is because shape scoring guides the search through the space of object combinations based on the objects' shape fitness, compared to H and UCS that do not have any measure of the fitness of the objects for tool construction. To summarize, feature scoring enables the robot to successfully construct tools by leveraging the sensor trust parameter, while significantly outperforming the baselines in terms of the resource budget required.

In order to understand which tools were more challenging for feature scoring, **Table 3** shows a tool-wise breakdown in performance for feature guided A^* for the two different sensor trust values. The notation "trust" denotes the case where sensors are fully trusted, and "$\sim$trust" denotes case where they are not fully trusted. When the sensors are fully trusted, rakes were a particularly challenging test case, as indicated by the lowest success rate of 7/10. In contrast, hammers and ladles have a success rate of 10/10. The failure cases for each tool arises from incorrect material and attachment predictions. While not fully trusting the sensors ($\sim$trust) leads to a 100% success rate (60/60 cases), using shape score alone leads to more failed construction

attempts when compared to combining shape with material and attachment predictions since shape alone is less informative (e.g., for rake, $\sim$trust has 7 failed attempts vs. 3 failed attempts for trust).

Figure 3 shows sample task plans generated by the robot in cooking and cleaning tasks. In the case of cooking, the robot needed a spatula to flip the eggs, and used a flat piece (obj4) with tongs (obj5) to construct the spatula via grasp attachment. For cleaning, the robot needed a squeegee to clean the window, and used a foam block (obj1) and screwdriver (obj6) to construct the squeegee via pierce attachment. Without the constructed tools, the actions highlighted in red would fail, i.e., the "flip" action would fail without the constructed spatula. Hence, FGS enables the robot to replace missing tools through construction. To summarize, the key findings of this experiment indicate that feature scoring is highly informative for heuristic search by reducing the number of nodes expanded by $\approx 82\%$, and the number of failed construction attempts by $\approx 93\%$, compared to the baselines. Further, allowing the robot to switch the trust parameter when a plan is not found, helps achieve a success rate of 100% within a budget of ≈ 39 attempts, significantly outperforming baselines that do not use feature scoring.

4.2. Feature Scoring With Other Heuristic Search Algorithms

To demonstrate that feature scoring generalizes to other search approaches, in this section we present results for combining feature scoring with weighted A^* (Pohl, 1970), and enforced hill-climbing using the fast-forward heuristic (Hoffmann and Nebel, 2001). We use the same experimental setup and metrics as described in section 4.1. In addition, we also measure the output plan length to investigate the optimality of the different approaches. For weighted A^*, feature scoring is incorporated as $f(s) = g(s) + w * (h(s) - \phi(s))$, where w indicates a weight parameter[8]. For enforced hill-climbing, the cost function is computed as $f(s) = h(s) - \phi(s)$. For both weighted A^* and enforced hill-climbing, we use the fast-forward heuristic, which has been shown to be successful for planning tasks in prior work (Hoffmann and Nebel, 2001).
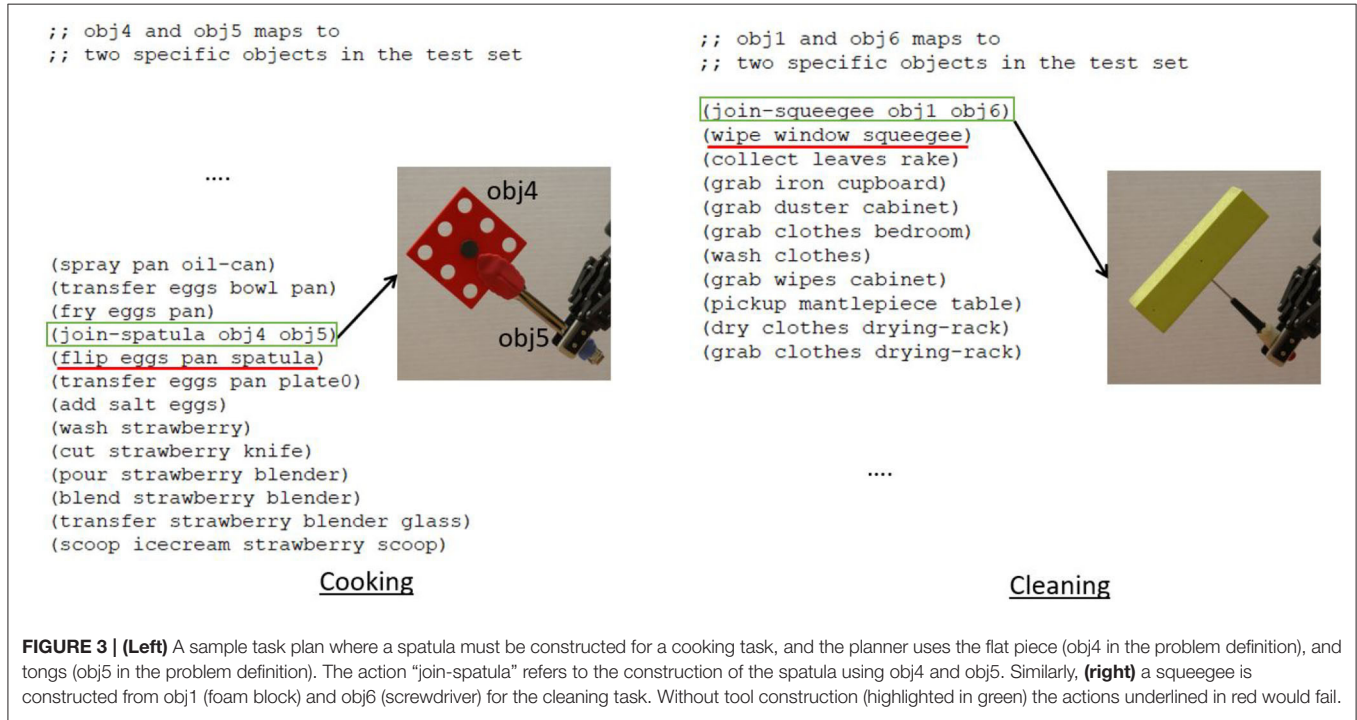
In **Table 4**, we present the results for feature scoring combined with A^* and the cost-optimal landmark heuristic ("A^*+LM"), weighted A^* with fast-forward heuristic ("wA^*+FF"), and enforced hill-climbing with fast forward heuristic ("eHC+FF"). Compared to A^*+LM, wA^*+FF and eHC+FF reduce the computational effort (fewer nodes expanded) in return for sub-optimal solutions (longer plan lengths). This is expected of weighted A^* and enforced hill-climbing since they are inadmissible algorithms. There is no statistically significant difference between # failed construction attempts in each case. To summarize, the key finding of this experiment is that feature scoring can be applied to other planning heuristics such as fast-forward, and other heuristic search algorithms like weighted A^* and enforced hill-climbing, to further reduce computational effort, albeit at the cost of optimality in terms of plan length.

---

[8]Weight was set to 5.0.

TABLE 3 | Table showing tool-wise breakdown in performance for feature guided *A**.

	Cleaning				Wood-working				Cooking			
	Squeegee		Rake		Hammer		Screwdriver		Spatula		Ladle	
	Trust	~Trust	Trust	~Trust	Trust	~Trust	Trust	~Trust	Trust	~Trust	Trust	~Trust
# Failed attempts	0	1	3	7	2	2	2	8	3	7	2	2
# Success	9/10	10/10	7/10	10/10	10/10	10/10	8/10	10/10	8/10	10/10	10/10	10/10

This table reports the average number of failed attempts per tool, across cases where tool construction was successful. The notation ~trust indicates cases where sensors are not fully trusted. Note that max # failed attempts is 89.


```
;; obj4 and obj5 maps to
;; two specific objects in the test set



    ....



(spray pan oil-can)
(transfer eggs bowl pan)
(fry eggs pan)
(join-spatula obj4 obj5)
(flip eggs pan spatula)
(transfer eggs pan plate0)
(add salt eggs)
(wash strawberry)
(cut strawberry knife)
(pour strawberry blender)
(blend strawberry blender)
(transfer strawberry blender glass)
(scoop icecream strawberry scoop)
```

Cooking

```
;; obj1 and obj6 maps to
;; two specific objects in the test set

(join-squeegee obj1 obj6)
(wipe window squeegee)
(collect leaves rake)
(grab iron cupboard)
(grab duster cabinet)
(grab clothes bedroom)
(wash clothes)
(grab wipes cabinet)
(pickup mantlepiece table)
(dry clothes drying-rack)
(grab clothes drying-rack)


    ....
```

Cleaning

FIGURE 3 | (Left) A sample task plan where a spatula must be constructed for a cooking task, and the planner uses the flat piece (obj4 in the problem definition), and tongs (obj5 in the problem definition). The action "join-spatula" refers to the construction of the spatula using obj4 and obj5. Similarly, **(right)** a squeegee is constructed from obj1 (foam block) and obj6 (screwdriver) for the cleaning task. Without tool construction (highlighted in green) the actions underlined in red would fail.

TABLE 4 | Table showing performance of feature guided Weighted *A** (w*A**) and feature guided Enforced Hill-Climbing (eHC) with the fast-forward heuristic (FF).

|	Cleaning			Cooking			Wood-working		
	A*+LM	w*A**+FF	eHC+FF	*A**+LM	w*A**+FF	eHC+FF	*A**+LM	w*A**+FF	eHC+FF
# Nodes	5187	21	21	329	23	35	7264	25	38
# Failed attempts	2	2	4	3	3	4	2	1	2
Plan length	20	22	22	19	19	19	11	15	15

4.3. Adaptability of Task Plans

In this section we evaluate the adaptability of our FGS approach to generate task plans based on objects in the environment, to appropriately use the constructed tool. We create three tasks, one task each for wood-working, cooking, and cleaning. In each of the tasks, *either* of two tools can be constructed to successfully complete the task, but there is only one ground truth depending on the objects available for construction. That is, the available objects only enable the construction of one of the two tools. Thus, the robot has to *correctly choose the tool to be constructed*. In addition, the robot must adapt the task plan to appropriately use the constructed tool. For the wood-working task either a hammer (with action "hit") or a screwdriver (with action "tighten") can be used to attach two pieces of wood; for the cooking task either a spatula (with action "flip") or a ladle (with action "scoop") can be used to flip eggs; and for the cleaning task, either a squeegee (with action "reach") or a rake (with action "collect") can be used to collect garbage.

For the evaluation, we create three different tasks, one each in wood-working, cooking, and cleaning. For each task, either
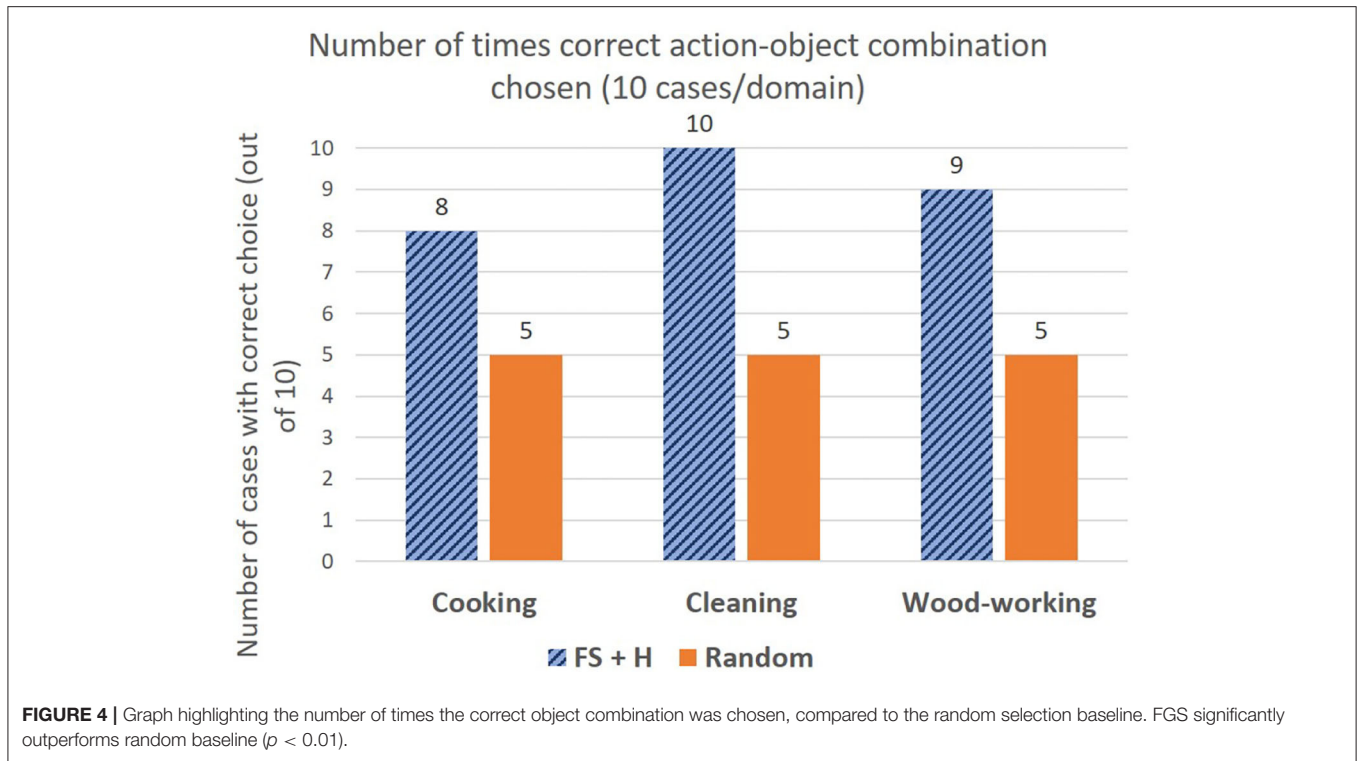
FIGURE 4 | Graph highlighting the number of times the correct object combination was chosen, compared to the random selection baseline. FGS significantly outperforms random baseline ($p < 0.01$).
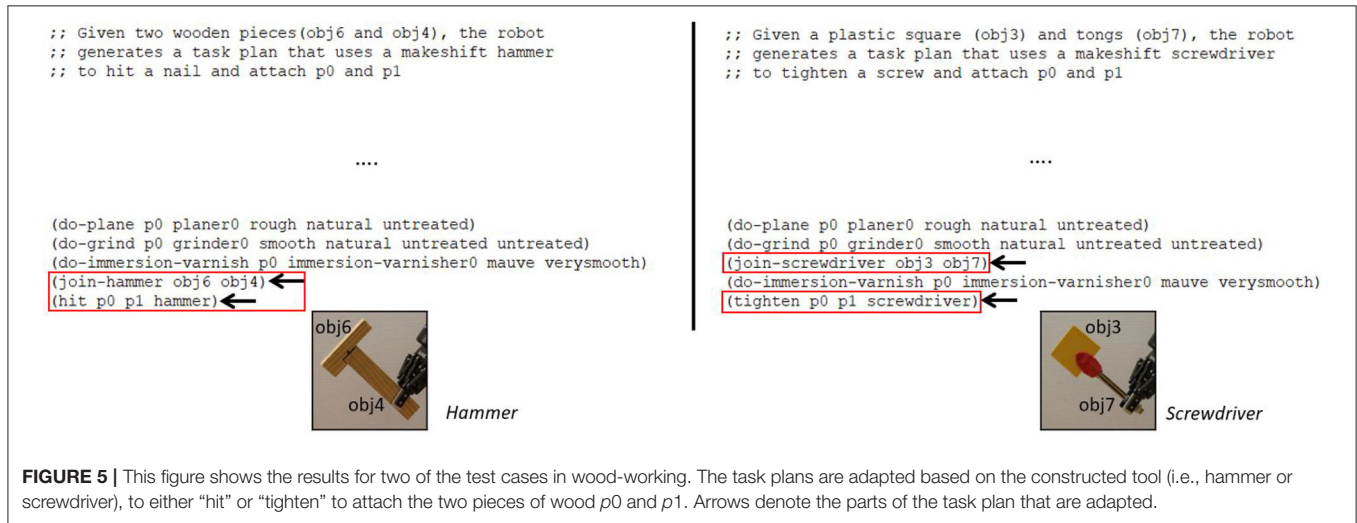
FIGURE 5 | This figure shows the results for two of the test cases in wood-working. The task plans are adapted based on the constructed tool (i.e., hammer or screwdriver), to either "hit" or "tighten" to attach the two pieces of wood $p0$ and $p1$. Arrows denote the parts of the task plan that are adapted.

one of two tools can be used to complete the task as described above. For each task, we created 10 different test sets of random objects, similar to the experiment described in section 4.1. In each case, only one "correct" combination exists. Thus, the robot has to correctly identify which of the two tools can be constructed for accomplishing the task, given the set of objects. We evaluate the performance of feature guided A^* in each case alongside a random selection baseline to demonstrate the difficulty of the problem. The random selection baseline randomly chooses one of the two tool construction options for each task. Note that for each task, the domain and problem definitions are unchanged

across the 10 test cases of objects. This indicates that the task plan adaptability does not require any manual modifications by the user, instead is the direct result of the sensor inputs received by the robot.

The key metric used in this experiment includes the number of times the robot chose the correct tool to construct for each task. Thus, if the robot chose to construct a hammer, when the correct combination was to construct a screwdriver, the attempt is considered to have failed. We also present qualitative results showing some of the sample task plans and tools constructed by the robot for different sets of objects.
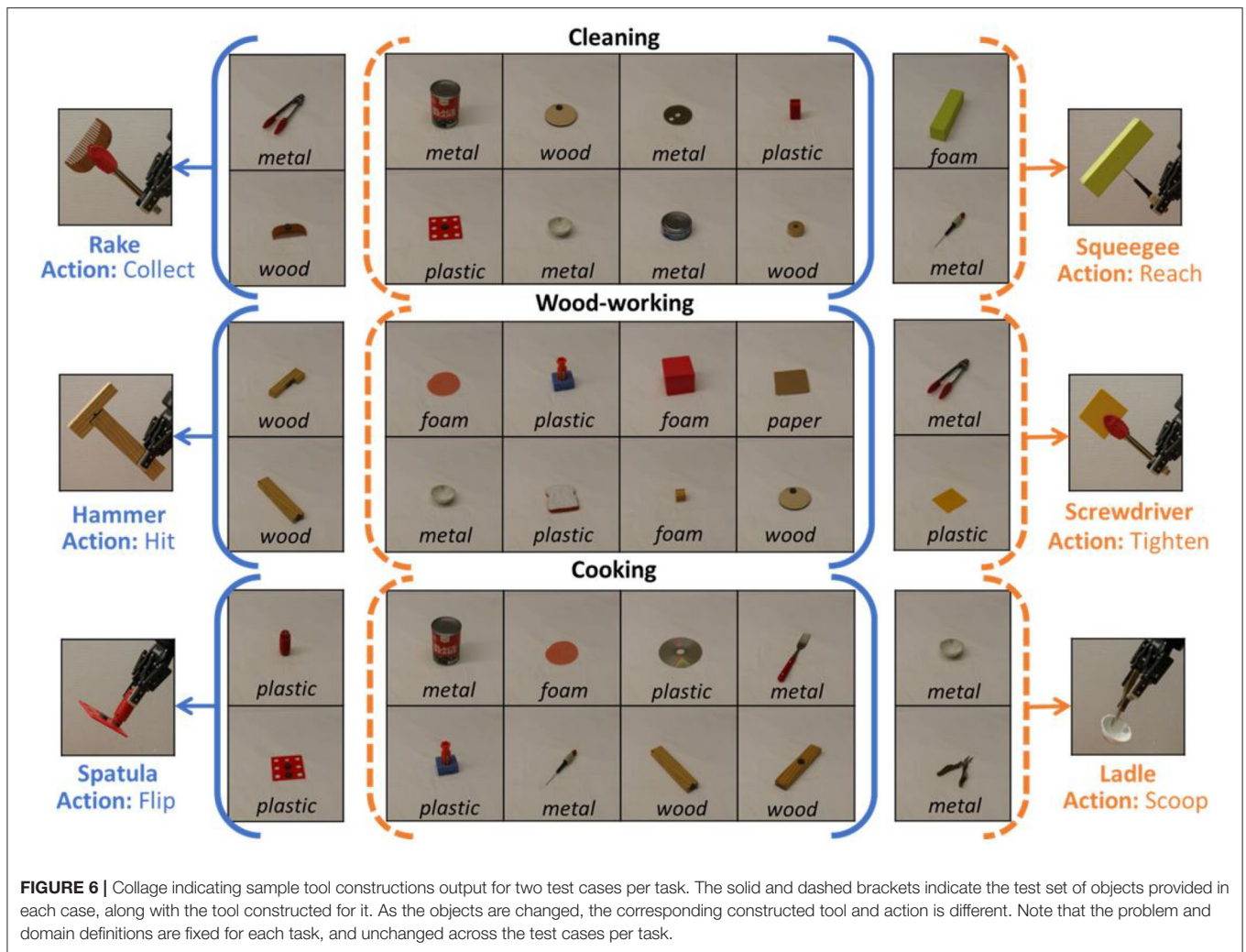
FIGURE 6 | Collage indicating sample tool constructions output for two test cases per task. The solid and dashed brackets indicate the test set of objects provided in each case, along with the tool constructed for it. As the objects are changed, the corresponding constructed tool and action is different. Note that the problem and domain definitions are fixed for each task, and unchanged across the test cases per task.

Figure 4 shows the performance of feature guided A^* compared to the random selection baseline. We see that feature guided A^* chooses the correct tool for 27/30 cases, and significantly outperforms the random selection baseline ($p < 0.01$). The failure cases in the wood-working task arise due to noisy material detection. In the case of cooking task, the noisy point clouds sensed by the RGBD camera leads to incorrect choices, e.g., the concavity of bowls was not correctly detected for some ladles.

In **Figure 5**, we show two task plans that are generated within the task of wood-working. For the same task, either a hammer or a screwdriver can be used to attach two pieces of wood $p0$ and $p1$. Depending, on the objects available in the environment, the robot chooses to construct one of the two tools and adapts the task plan to use the corresponding tool for completing the task. As shown in the left of **Figure 5**, the robot chose to construct a hammer to "hit" and attach the two pieces of wood. Whereas, shown in the right of **Figure 5**, the robot chose to construct a screwdriver to "tighten" and attach the two pieces of wood. Similar adaptations are observed for the remaining two tasks as well: "scoop" with

ladles vs. "flip" with spatulas in the cooking task, and "reach" with squeegees vs. "collect" with rakes in the cleaning task. Thus, the constructed tool depends on the objects in the environment, which in turn adapts the generated task plan to appropriately use the constructed replacement tool.

In **Figure 6**, we present some qualitative results for six different tools constructed by the robot for six of the test cases. The solid and dashed parentheses highlight the input test set. For example, given the metal bowl and metal pliers, the robot chooses to construct a ladle (and use the "scoop" action in the task plan). In contrast, when the pliers and bowl are replaced with a plastic handle and a flat plastic piece, the robot chooses to construct a spatula instead (and use the "flip" action in the task plan). Given that the problem and domain definitions are unchanged for the two cases, this shows that the robot is able to adapt the task plan in response to the objects in the environment. To summarize, the key finding of this experiment is that the robot is able to successfully adapt the task plan to construct and use the appropriate tool depending on the objects available for construction, with an accuracy of 90% (27/30 cases).

5. CONCLUSION AND FUTURE WORK

In this work, we presented the Feature Guided Search (FGS) approach that allows existing heuristic search algorithms to be efficiently applied to the problem of tool construction in the context of task planning. Our approach enables the robot to effectively construct and use tools in cases where the required tools for performing the task are unavailable. We relaxed key assumptions of the prior work in terms of eliminating the need to specify an input action, instead integrating tool construction within a task planning framework. Our key findings can be summarized as follows:

- FGS significantly reduces the number of nodes expanded by $\approx$ 82%, and the number of construction attempts by $\approx$ 93%, compared to standard heuristic search baselines.
- The approach achieves a success rate of 87% within a resource budget of 8 attempts when sensors are fully trusted, and 100% within a budget of 39 attempts, when the sensors are not fully trusted.
- FGS enables flexible generation of task plans based on objects in the environment, by adapting the task plan to appropriately use the constructed tool.
- Feature scoring can also be effectively combined with other heuristic search algorithms such as weighted A^* and enforced hill-climbing.

Our work is one of the first to integrate tool construction within a task planning framework, but there remain many unaddressed manipulation challenges in tool construction that are beyond the scope of this paper. Tool construction is a challenging manipulation problem that involves appropriately grasping and combining the objects to successfully construct the tool. That is, once the robot has correctly identified the objects that need to be combined (focus of this paper), the robot would then have to physically combine the objects, and use the constructed tool for the task. Currently, our work pre-specifies the trajectories to be followed for tool construction, although existing research in robot assembly can be leveraged to potentially accomplish this (Thomas et al., 2018). Further, a key question to be addressed is, *how can the robot learn to appropriately use the constructed tool?* Future work could address this problem by leveraging existing research in tool use (Stoytchev, 2005; Sinapov and Stoytchev, 2007, 2008), and trajectory-based skill adaptation (Fitzgerald et al., 2014; Gajewski et al., 2019). Upon successful construction of the tool, the research problem reduces to that of using the tool appropriately. In this case, the robot can either learn how to use the tool as described in Stoytchev (2005), Sinapov and Stoytchev (2008, 2007) or, the robot can adapt previously known tool manipulation skills to the newly constructed tool as described in Fitzgerald et al. (2014) and Gajewski et al. (2019). Addressing these challenges is important to further ensure practical applicability of tool construction.

Additionally, creation of tools through the attachment types discussed in this work is currently restricted to a limited number of use cases, in which two objects that have the specific attachment capabilities already exist, and are available to the robot. In the future, we seek to expand to more diverse types of attachments, including gluing or welding the objects together, as well as creation of tools from deformable materials, in order to improve the usability of our work. We further seek to expand on this work by investigating the application of feature scoring to domains other than tool construction. In particular, we seek to investigate the different ways in which feature score can be effectively combined with the cost function for other domains involving tool-use such as tool substitution. While our proposed cost function is dependent on the values of the feature score and is shown to perform well for tool construction, it is important to further investigate the cost function and its influence on the guarantees of the search to allow for a more generalized application of FGS.

FGS enables the robot to perform high-level decision making with respect to the objects that must be combined in order to construct a required tool. In this work, we use physical sensors (RGBD sensors and SCiO spectrometer) that produce partial point clouds and noisy spectral scans, leading to some challenges that commonly arise in the real world. Nevertheless, there are several open research questions that need to be addressed before this work can be deployed in a real setting. Thus, FGS is the first step within a larger pipeline, and we envision this work to be complementary to existing frameworks that are aimed at resilient and creative task execution, such as Antunes et al. (2016) and Stückler et al. (2016). In summary, FGS presents a promising direction for dealing with tool-based problems in the area of creative problem solving.

DATA AVAILABILITY STATEMENT

The datasets presented in this study can be found in online repositories. The names of the repository/repositories and accession number(s) can be found at: https://github.com/Lnair1993/Tool_Macgyvering.

AUTHOR CONTRIBUTIONS

LN and SC conceived of the presented idea. LN developed the theory, performed the computations, and carried out the experiment. LN wrote the manuscript with support from SC. All authors contributed to the article and approved the submitted version.

FUNDING

ACKNOWLEDGMENTS

REFERENCES

Abelha, P., and Guerin, F. (2017). "Learning how a tool affords by simulating 3D models from the web," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (Vancouver, BC), 4923–4929.

Abelha, P., Guerin, F., and Schoeler, M. (2016). "A model-based approach to finding substitute tools in 3d vision data," in *2016 IEEE International Conference on Robotics and Automation (ICRA)* (Stockholm), 2471–2478.

Agostini, A., Aein, M. J., Szedmak, S., Aksoy, E. E., Piater, J., and Würgütter, F. (2015). "Using structural bootstrapping for object substitution in robotic executions of human-like manipulation tasks," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (Hamburg), 6479–6486.

Antunes, A., Jamone, L., Saponaro, G., Bernardino, A., and Ventura, R. (2016). "From human instructions to robot actions: formulation of goals, affordances and probabilistic planning," in *2016 IEEE International Conference on Robotics and Automation (ICRA)* (Stockholm), 5449–5454.

Atkeson, C. G., Benzun, P. B., Banerjee, N., Berenson, D., Bove, C. P., Cui, X., et al. (2018). "What happened at the Darpa robotics challenge finals," in *The DARPA Robotics Challenge Finals: Humanoid Robots to the Rescue* (Pittsburgh, PA: Springer), 667–684.

Bagchi, A., and Mahanti, A. (1983). Search algorithms under different kinds of heuristics-a comparative study. *J. ACM* 30, 1–21.

Beck, S. R., Apperly, I. A., Chappell, J., Guthrie, C., and Cutting, N. (2011). Making tools isn't child's play. *Cognition* 119, 301–306. doi: 10.1016/j.cognition.2011.01.003

Boteanu, A., Kent, D., Mohseni-Kabir, A., Rich, C., and Chernova, S. (2015). "Towards robot adaptability in new situations," in *2015 AAAI Fall Symposium Series* (Arlington, VA: AAAI Press).

Cass, S. (2005). Apollo 13, we have a solution. *IEEE Spectr. On-line* 4:1. Available online at: https://spectrum.ieee.org/tech-history/space-age/apollo-13-we-have-a-solution

Choi, D., Langley, P., and To, S. T. (2018). "Creating and using tools in a hybrid cognitive architecture," in *2018 AAAI Spring Symposium Series* (Arlington, VA).

Coradeschi, S., and Saffiotti, A. (2003). An introduction to the anchoring problem. *Robot. Auton. Syst.* 43, 85–96. doi: 10.1016/S0921-8890(03)00021-6

Daruna, A., Liu, W., Kira, Z., and Chetnova, S. (2019). "Robocse: Robot common sense embedding," in *2019 International Conference on Robotics and Automation (ICRA)* (Montreal, QC), 9777–9783.

Erdogan, C., and Stilman, M. (2013). "Planning in constraint space: automated design of functional structures," in *2013 IEEE International Conference on Robotics and Automation (ICRA)* (Karlsruhe), 1807–1812.

Erickson, Z., Luskey, N., Chernova, S., and Kemp, C. (2019). Classification of household materials via spectroscopy. *IEEE Robot. Autom. Lett.* 4, 700–707. doi: 10.1109/LRA.2019.2892593

Erickson, Z., Xing, E., Srirangam, B., Chernova, S., and Kemp, C. C. (2020). Multimodal material classification for robots using spectroscopy and high resolution texture imaging. *arXiv [Preprint]*. arXiv:2004.01160v2. Available online at: https://arxiv.org/abs/2004.01160

Fitzgerald, T., Goel, A. K., and Thomaz, A. L. (2014). "Representing skill demonstrations for adaptation and transfer," in *AAAI Symposium on Knowledge, Skill, and Behavior Transfer in Autonomous Robots* (Pao Alto, CA).

Freedman, R., Friedman, S., Musliner, D., and Pelican, M. (2020). "Creative problem solving through automated planning and analogy," in *AAAI 2020 Workshop on Generalization in Planning (GenPlan 20)* (New York, NY).

Gajewski, P., Ferreira, P., Bartels, G., Wang, C., Guerin, F., Indurkhya, B., et al. (2019). "Adapting everyday manipulation skills to varied scenarios," in *2019 International Conference on Robotics and Automation (ICRA)* (Montreal, QC), 1345–1351.

Gizzi, E., Castro, M. G., and Sinapov, J. (2019). "Creative problem solving by robots using action primitive discovery," in *2019 Joint IEEE 9th International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob)* (Oslo), 228–233.

Harnad, S. (1990). The symbol grounding problem. *Phys. D* 42, 335–346.

Hart, P. E., Nilsson, N. J., and Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans. Syst. Sci. Cybern.* 4, 100–107.

Hiraoka, T., Onishi, T., Imagawa, T., and Tsuruoka, Y. (2018). Refining manually-designed symbol grounding and high-level planning by policy gradients. *arXiv [Preprint]*. arXiv:1810.00177. Available online at: https://arxiv.org/abs/1810.00177

Hoffmann, J., and Nebel, B. (2001). The FF planning system: fast plan generation through heuristic search. *J. Artif. Intell. Res.* 14, 253–302. doi: 10.1613/jair.855

Jones, T. B., and Kamil, A. C. (1973). Tool-making and tool-using in the northern blue jay. *Science* 180, 1076–1078.

Karpas, E., and Domshlak, C. (2009). "Cost-optimal planning with landmarks," in *IJCAI* (Pasadena, CA), 1728–1733.

Konidaris, G., Kaelbling, L. P., and Lozano-Perez, T. (2018). From skills to symbols: learning symbolic representations for abstract high-level planning. *J. Artif. Intell. Res.* 61, 215–289. doi: 10.1613/jair.5575

Levihn, M., and Stilman, M. (2014). "Using environment objects as tools: unconventional door opening," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2014)* (Chicago, IL), 2502–2508.

Levine, S., Pastor, P., Krizhevsky, A., Ibarz, J., and Quillen, D. (2018). Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *Int. J. Robot. Res.* 37, 421–436. doi: 10.1177/0278364917710318

McDermott, D., Ghallab, M., Howe, A., Knoblock, C., Ram, A., Veloso, M., et al. (1998). *PDDL- The Planning Domain Definition Language.* Technical Report CVC TR-98-003/DCS TR-1165, Yale Center for Computational Vision and Control.

Myers, A., Teo, C. L., Fermüller, C., and Aloimonos, Y. (2015). "Affordance detection of tool parts from geometric features," in *International Conference on Robotics and Automation (ICRA)* (Seattle, WA), 1374–1381.

Nair, L., Balloch, J., and Chernova, S. (2019a). "Tool macgyvering: tool construction using geometric reasoning," in *International Conference on Robotics and Automation (ICRA)* (Montreal, QC), 5837–5843.

Nair, L., and Chernova, S. (2020). Feature guided search for creative problem solving through tool construction. *arXiv [Preprint]*. arXiv:2008.10685. Available online at: https://arxiv.org/abs/2008.10685

Nair, L., Srikanth, N., Erikson, Z., and Chernova, S. (2019b). "Autonomous tool construction using part shape and attachment prediction," in *Proceedings of Robotics: Science and Systems* (Messe Frieburg), 1–10.

Nyga, D., Roy, S., Paul, R., Park, D., Pomarlan, M., Beetz, M., et al. (2018). "Grounding robot plans from natural language instructions with incomplete world knowledge," in *Conference on Robot Learning* (Zurich), 714–723.

Olteteanu, A.-M., and Falomir, Z. (2016). Object replacement and object composition in a creative cognitive system. Towards a computational solver of the alternative uses test. *Cogn. Syst. Res.* 39, 15–32. doi: 10.1016/j.cogsys.2015.12.011

Pierson, A., and Schwager, M. (2016). "Adaptive inter-robot trust for robust multi-robot sensor coverage," in *Robotics Research* (Springer), 167–183. Available online at: https://link.springer.com/chapter/10.1007/978-3-319-28872-7_10

Pohl, I. (1970). Heuristic search viewed as path finding in a graph. *Artif. Intell.* 1, 193–204.

Saboia, M., Thangavelu, V., Gosrich, W., and Napp, N. (2018). Autonomous adaptive modification of unstructured environments. *Robot. Sci. Syst.* 14, 70–78. doi: 10.15607/RSS.2018.XIV.070

Sarathy, V. (2018). Real world problem-solving. *Front. Hum. Neurosci.* 12:261. doi: 10.3389/fnhum.2018.00261

Sarathy, V., and Scheutz, M. (2017). The MacGyver test-a framework for evaluating machine resourcefulness and creative problem solving. *arXiv [Preprint]*. arXiv:1704.08350. Available online at: https://arxiv.org/abs/1704.08350

Sarathy, V., and Scheutz, M. (2018). MacGyver problems: AI challenges for testing resourcefulness and creativity. *Adv. Cogn. Syst.* 6.

Schoeler, M., and Wörgötter, F. (2015). Bootstrapping the semantics of tools: affordance analysis of real world objects on a per-part basis. *IEEE Trans. Cogn. Dev. Syst.* 8, 84–98. doi: 10.1109/TAMD.2015.2488284

Shrivatsav, N., Nair, L., and Chernova, S. (2019). Tool substitution with shape and material reasoning using dual neural networks. *arXiv [Preprint]*. arXiv:1911.04521.

Sinapov, J., and Stoytchev, A. (2007). "Learning and generalization of behavior-grounded tool affordances," in *2007 IEEE 6th International Conference on Development and Learning* (London), 19–24.

Sinapov, J., and Stoytchev, A. (2008). "Detecting the functional similarities between tools using a hierarchical representation of outcomes," in *2008 7th IEEE International Conference on Development and Learning* (Monterey, CA), 91–96.

Stilman, M., Zafar, M., Erdogan, C., Hou, P., Reynolds-Haertle, S., and Tracy, G. (2014). Robots using environment objects as tools the 'MacGyver' paradigm for mobile manipulation," in *2014 IEEE International Conference on Robotics and Automation (ICRA)* (Hong Kong), 2568–2568.

Stout, D. (2011). Stone toolmaking and the evolution of human culture and cognition. *Philos. Trans. R. Soc. Lond. B Biol. Sci.* 366, 1050–1059. doi: 10.1098/rstb.2010.0369

Stoytchev, A. (2005). "Behavior-grounded representation of tool affordances," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation* (Barcelona), 3060–3065.

Stückler, J., Schwarz, M., and Behnke, S. (2016). Mobile manipulation, tool use, and intuitive interaction for cognitive service robot cosero. *Front. Robot. AI* 3:58. doi: 10.3389/frobt.2016.00058

ten Pas, A., Gualtieri, M., Saenko, K., and Platt, R. (2017). Grasp pose detection in point clouds. *Int. J. Robot. Res.* 36, 1455–1473. doi: 10.1177/0278364917735594

Thomas, G., Chien, M., Tamar, A., Ojea, J. A., and Abbeel, P. (2018). "Learning robotic assembly from cad," in *2018 IEEE International Conference on Robotics and Automation (ICRA)* (Brisbane), 1–9.

Tosun, T., Daudelin, J., Jing, G., Kress-Gazit, H., Campbell, M., and Yim, M. (2018). "Perception-informed autonomous environment augmentation with modular robots," in *2018 IEEE International Conference on Robotics and Automation (ICRA)* (Brisbane), 6818–6824.

Toth, N., Schick, K. D., Savage-Rumbaugh, E. S., Sevcik, R. A., and Rumbaugh, D. M. (1993). Pan the tool-maker: Investigations into the stone tool-making and tool-using capabilities of a bonobo (*Pan paniscus*). *J. Archaeol. Sci.* 20, 81–91.

Turner, M., Duggan, L., Glezerson, B., and Marshall, S. (2020). Thinking outside the (acrylic) box: a framework for the local use of custom-made medical devices. *Anaesthesia* 75, 1566–1569. doi: 10.1111/anae. 15152

Wicaksono, H., and Sheh, C. S. R. (2017). "Towards explainable tool creation by a robot," in *IJCAI-17 Workshop on Explainable AI (XAI)* (Melbourne), 63.

Wohlkinger, W., and Vincze, M. (2011). "Ensemble of shape functions for 3d object classification," in *2011 IEEE International Conference on Robotics and Biomimetics (ROBIO)* (St. Paul, MN), 2987–2992.

Zech, P., and Piater, J. (2016). Grasp learning by sampling from demonstration. *arXiv [Preprint].* arXiv:1611.06366. Available online at: https://arxiv.org/abs/1611.06366

Zhang, Z., Sturtevant, N. R., Holte, R. C., Schaeffer, J., and Felner, A. (2009). "A* search with inconsistent heuristics," in *IJCAI* (Pasadena, CA), 634–639.