International Journal of Electrical and Computer Engineering (IJECE)
Vol. 8, No. 2, April 2018, pp. 1230 – 1237
ISSN: 2088-8708, DOI: 10.11591/ijece.v8i2.pp1230-1237

Tool Use Learning for a Real Robot

Handy Wicaksono[1,2] **and Claude Sammut**[1]
[1]School of Computer Science and Engineering, University of New South Wales; Sydney, Australia
[2]Electrical Engineering Department, Petra Christian University; Surabaya, Indonesia

Article Info

Article history:
Received July 24, 2017
Revised December 16, 2017
Accepted December 27, 2017

Keyword:
tool use by a robot
tool use learning
action learning
inductive logic programming
robot software architecture

ABSTRACT

A robot may need to use a tool to solve a complex problem. Currently, tool use must be pre-programmed by a human. However, this is a difficult task and can be helped if the robot is able to learn how to use a tool by itself. Most of the work in tool use learning by a robot is done using a feature-based representation. Despite many successful results, this representation is limited in the types of tools and tasks that can be handled. Furthermore, the complex relationship between a tool and other world objects cannot be captured easily. Relational learning methods have been proposed to overcome these weaknesses [1, 2]. However, they have only been evaluated in a sensor-less simulation to avoid the complexities and uncertainties of the real world. We present a real world implementation of a relational tool use learning system for a robot. In our experiment, a robot requires around ten examples to learn to use a hook-like tool to pull a cube from a narrow tube.

Corresponding Author:
Handy Wicaksono
School of Computer Science and Engineering, University of New South Wales
handyw@cse.unsw.edu.au

1. INTRODUCTION

Humans use tools to help them complete everyday tasks. The ability to use tools is a feature of humans intelligence [3]. Like humans, a robot also needs to be able to use a tool to solve a complex task. As an example, in the RoboCup@Home competition, a robot is asked to demonstrate several tool use abilities such as opening a bottle by using a bottle opener or watering a plant with a watering can [4]. The robot is given complete knowledge of the tools and how to use them. When such knowledge is not available, a robot must learn it. Most work in tool use learning has used a feature-based representation that is dependent on an object's primitive features. Thus, it is not flexible enough to be applied to different tools and environments. Learning is also limited to tool selection only. Brown proposed a relational approach which can overcome these limitations. Furthermore, Wicaksono & Sammut ([5]) have suggested that this representation has potential to solve a more complex problem, such as tool creation.

We define a tool as an object that is deliberately employed by an agent to help it achieve a goal, which would be too difficult to achieve without the tool. We want to learn a tool action model that explains changes in the properties of one or more objects affected by the tool, given that certain preconditions are met. Following Brown [1], learning is performed by trial and error in an Inductive Logic Programming (ILP) setting [6]. Brown [1] carried out tool use learning in a sensor-less simulation. This means that a robot has perfect knowledge of the world and uncertainties in the sensors' readings are eliminated. In this paper, we would present a complete robotic system for tool use learning following a relational learning approach. This includes three components:

1. Developing a robot software architecture that consists of primitive and abstract layers and facilitates communication between them.
2. Creating a mechanism to detect objects and generating primitive behaviors for tool use.
3. Extending relational learning methods and conducting tool use learning experiments using a real robot.

In the following sections, we describe relevant previous work, the knowledge representation formalism used, the software architecture, and the tool use learning mechanism. Finally, we perform several real world experiments and draw conclusions.

2. RELATED WORK

An intelligent robot is identified by its ability to make a decision and learn autonomously in an unstructured environment. Machine learning techniques could be useful as they can compensate for the imperfect model of a robot in a real world. For example, control of a robot manipulator can be improved by equipping a PD torque controller with Adaptive Neuro-Fuzzy Inference System [7] or combining sliding mode control with genetic algorithm [8]. In a behavior-based robotics approach, reinforcement learning can be used to learn the robot's behavior [9].

Here we describe the previous work in tool use learning by a robot. Wood et al. [10] use a neural network to learn appropriate posture of a Sony Aibo robot so it can reach an object by using a tool placed on its back. Stoytchev [11] learns to select a correct tool via its affordances which are grounded in robot behaviors. However, its result can not be generalized to other new tools. More recent work by Tikhanoff et al. [12]) combine exploratory behaviors and a geometrical feature extraction to learn affordances and tool use. Mar et al. [13] extend their work by learning a grasp configuration which influences the outcome of a tool use action.

The limitations of feature-based representations were mentioned above. Only recently have such representations been extended so that a robot can learn to grasp a tool [13]. An ILP system can overcome these limitations, as it represents the tools and other objects in a relational representation [1, 2]. However, previous work has been done previously in sensor-less simulation only. This means the complexities of acquiring perceptions, generating precise movements, and dealing with world uncertainties, are avoided. We aim to develop a complete robotic system that facilitates this learning in a real world.

3. REPRESENTING STATES AND ACTIONS

We maintain two representations of states and actions in primitive and abstract form. Primitive states are the positions of all objects in the world that are captured by vision sensors using a mechanism described in section 4.2.. As we only use simple objects, they can be detected by their two-dimensional appearance only.

Abstract states are represented as expressions in first-order logic, to be more specific as Horn clauses. Primitive states are translated to an abstract state by calling relevant Prolog programs with the primitive states as their bound values. To be classified as a tool, an object must possess particular structural properties (e.g. a particular side where a hook is attached to the handle) and spatial properties (e.g. a hook is touching the back side of a cube). We collect these properties in a hypothesis, namely `tool_pose`, which is shown below in simplified form.

```
tool_pose(Handle,Hook,Box,Tube):-
    attached_end(Handle,Hook,back), % a structural property
    attached_side(Handle,Hook,Side), % a structural property
    touching(Hook,Box,back). % a spatial property
```

We use an extended STRIPS action model [14] to describe an abstract action and how it affects the world. A primitive goal that has to be achieved by a robot is added at the end of the model.

Action name

PRE : states that must be true so that an action can be performed

ADD : conditions that become true as a result of the action

DELETE : conditions that are no longer true following the action

CONSTRAINTS : the physical limits of actuators that constrain the action

This action model is also a planning operator. Thus, action learning does not happen in isolation, as it is a part of a problem solving scenario. Every abstract action is linked to a set of primitive behaviors. This will be discussed later in section 4.3.

4. ROBOTIC SYSTEM

In this section, we explain our software architecture, objects detection method, and behavior generation mechanism.

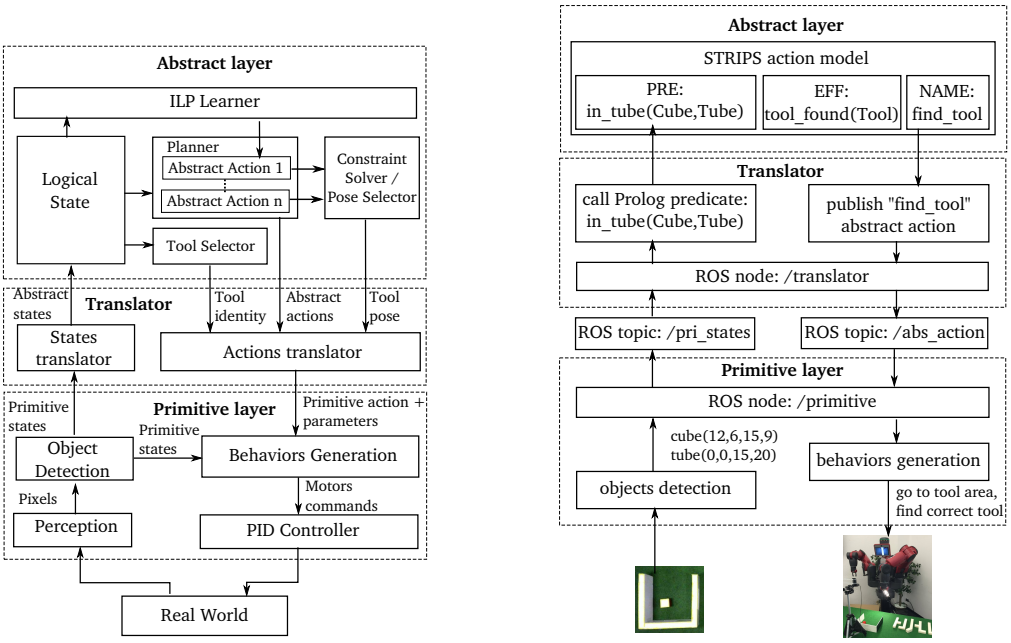
4.1. Software Architecture

We use a relational representation in the high-level layer to take advantage of the expressiveness of first-order logic (FOL) clauses, especially Horn clauses, and equip a robot with useful symbolic techniques. We implement this layer in SWI Prolog. Most modern Prolog implementations, including SWI, incorporate a constraint solver which provide advanced numerical reasoning capabilities.

In the low-level layer, a robot operates in a world of continuous, noisy and uncertain measurements. Its sensors return readings in numeric ranges, and its actuators operate based on numerical set points. As we aim to use Robot Operating System (ROS[1]) as our middleware, we implement this layer in Python. The Baxter robot, from Rethink Robotics, has its own ROS-Python API, a set of classes that provides wrappers around the ROS communication.

Another layer, namely the translator, is needed to map primitive to abstract states and to link an abstract action to corresponding primitive behaviors. It is written in C++ which has an interface to SWI Prolog as the language for the abstract layer. Communication to the primitive layer is done via ROS using a simple publish and subscribe mechanism. Our robot software architecture is shown in Fig. 1a.

We give a simple illustration of action execution, namely `find_tool`, which involves communication between layers. In the primitive layer, a cube and a tube are detected, their corners are acquired and published together by a ROS node, namely `/translator`, as a ROS topic, `/pri_states`. The translator also has a ROS node, `/translator`, that subscribes to that topic. Then it evaluates the status of the abstract state, `in_tube(Cube,Tube)`, by calling the relevant Prolog predicate. In the abstract layer, `in_tube(Cube,Tube)` is called. If it is true, as it is the only the precondition of the `find_tool` action, then the translator will publish it as the ROS topic, `/abs_action` that represents the currently active action. The primitive layer subscribes to this topic. If it recognizes `find_tool` as an active action, then it triggers the corresponding primitive behaviors. See Fig. 1b for detail.



(a) Software architecture

(b) Simplified example

Figure 1. Robot software architecture and its example

4.2. Object Detection

We detect all objects (i.e. different tools, a cube, and a tube) by their two-dimensional appearances only. We use the Baxter camera to detect objects locally, while an external web camera provides the global world state. We combine local and global images from both cameras to achieve our goal. The OpenCV library is used for object detection.

In pre-processing, Gaussian smoothing is applied. Later, the edges of objects in an image are detected with a Canny Edge detector, and their contours are found. Each contour is tested to check whether it has the properties of a particular object. The contour properties include the number of edges, the area, the angle formed by two adjacent parts, the convexity, and the perimeter. Tools are treated specially as they have more than one shape. After being detected as a tool, the object's type is determined, and its handle and hook(s) are acquired. All objects are represented

[1] http://www.ros.org

by their minimum and maximum corner points in Cartesian coordinates ($X_{min}, Y_{min}, X_{max}, Y_{max}$). The complete mechanism is shown in Algorithm 1.

Algorithm 1 Objects detection

```
Capture an image
Perform Gaussian smoothing
Detect the edges of objects using Canny detector
Find the contours of the objects
for each contour do
    Classify the contour
    if The contour is either cube, tube, or tool then
        Find its minimum & maximum corner points
return all minimum & maximum corner points
```

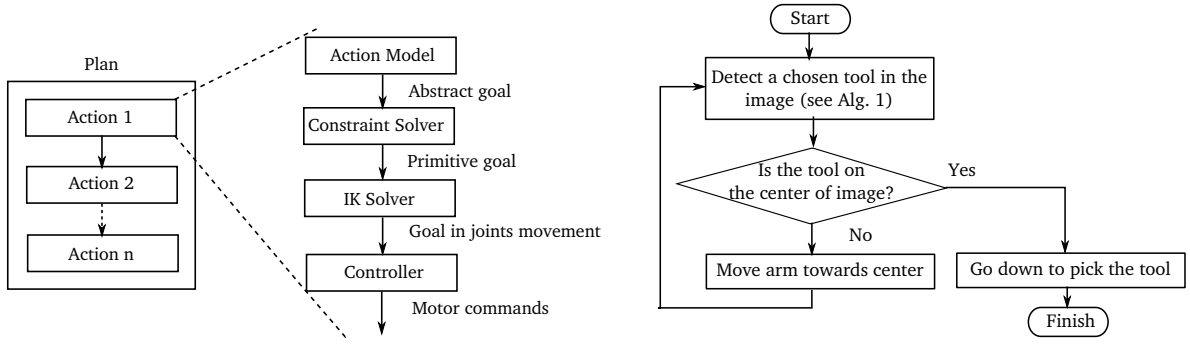
4.3. Behavior Generation

In our tool use application, a sequence of actions, or a plan, must be performed to achieve the goal, pulling a cube from a narrow tube. Each action uses different strategies (inverse kinematics solving, visual servoing, gripper activation, constraints solving) depending on whether the primitive goal is known or not, and whether the visual information is needed or not. We show an example of the plan in Table 1.

Table 1. The actions sequence for pulling a cube from a narrow tube

No	Action	Technique	Goal	Visual
1	Find the tool	Inverse Kinematics (IK) solving, visual servoing	Known	Yes
2	Grip the tool	Close the gripper	Known	No
3	Position the tool	Constraints solving, IK solving	Unknown	Yes
4	Pull the cube	IK solving	Known	No
5	Move the tool away	IK solving	Known	No
6	Ungrip the tool	Open the gripper	Known	No

An abstract action is directly linked to corresponding primitive actions. For simplicity, we only consider movement in two-dimensional Cartesian coordinates. A primitive goal is a target location in that coordinate system. When a goal is known, we use an inverse kinematics (IK) solver to compute the angles of all joints and move the arm towards that goal. However, if it is unknown, we use a constraint solver to create a numerical goal from abstract states which are the preconditions of an abstract action model (see Fig. 2a). To perform an accurate tool picking action, we use a simple version of image-based visual servoing [15], where an error signal is determined directly in terms of image feature parameters. We locate the robot arm in a position where a chosen tool is located in the center of an image captured by Baxter's wrist camera, so the arm can move downward perpendicularly to take the tool. See Fig. 2b for the detail.



(a) Constraints solving to get a primitive goal

(b) A visual servoing process

Figure 2. behaviors generation

5. TOOL USE LEARNING

In this section, we describe relational tool use learning [1, 2]. The robot must learn a tool action model that describes the properties of a tool and other objects in the world that enable a robot to use a tool successfully. Specifically, the robot learns a `tool_pose` predicate, which describes the structural and spatial conditions that must be met for the tool to be used. We maintain a version space of hypotheses, following Mitchell [16]. A version space is bounded by the most general hypothesis (h_g) and the most specific hypothesis (h_s). As we explain below, the version space is used to determine what experiments the robot should perform to generalize or specialize the tool action model.

Initially, a robot may not have a complete action model for a tool, so it cannot construct a plan. After a tutor gives a tool use example, the robot segments the observed behavior into discrete actions, matches them to actions already in its background knowledge, and constructs a new action if there is no match for a particular behavior. This construction may not be sufficient to describe the tool action in a general enough form that it can be applied in different situations. Therefore, trial and error learning is performed to refine the initial action model.

Trial and error involved performing experiments in which a different tool is selected or its pose is changed to test the current hypothesis for the tool action. In tool selection, an object that has properties that are most similar to a previously useful tool is chosen. More specifically, we test whether an object satisfies all primary structural properties, stored in h_g, and most of the secondary ones, stored in h_s, given its primitive states.

Having a correct tool is useless when it is located in an incorrect position before it trying to pull the target object. We select the pose by solving the constraints of the spatial literals in the tool pose predicate. We check whether the spatial constraints of an object can be solved or not. The unsolved constraints are ignored, and the final result is used as the numerical goal for the robot.

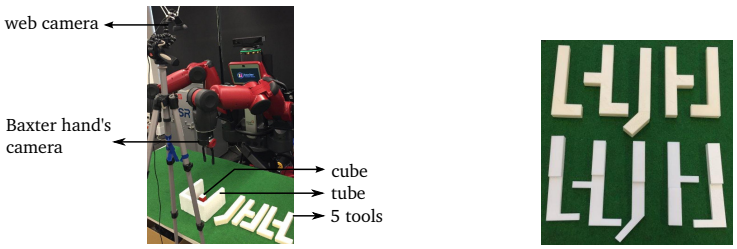
After a tool use learning experiment is performed, its result, whether successful or not, is passed to the ILP learner so the hypothesis can be refined. Our learning is derived from Golem [17], an ILP algorithm. Refinement of h_s is done by finding the constrained Least General Generalization (LGG) of a positive examples pair, and h_g is refined by performing the negative-based reduction. The learning algorithm is a modification of Haber [2] and is shown in Algorithm 2.

Algorithm 2 Tool use learning

input: new action model M, h_g = true, h_s = preconditions of M,N trials, K consecutive success
while $success < K$ or $index < N$ **do**
  $e = generate\_experiment(h_s, h_g)$
  $tool = select\_tool(h_s, h_g)$ // select a tool with highest rank
  for each e_i in e **do**
    $pose = select\_pose(e_i)$ // performing constraint solving on preconditions of the relevant action model
    if $pose = null$ **then**
      break
  $success = execute\_exp(tool, pose)$ // performing a tool use experiment
  if $success$ **then**
    label $pose$ positive
    increment $cons\_success$
    h_s = generalise h_s // via Least General Generalisation
  else
    label $pose$ negative
    $cons\_success \leftarrow 0$
    h_g = specialise h_g // via negative based reduction
  add e_i to training data
  increment $index$

6. RESULTS AND DISCUSSIONS

The Baxter research robot is used in this experiment (see Fig. 3a). The objects in the scene include a cube, a tube, and five objects of a different shape that potential tools. We also have another set of tool candidates whose hooks are narrower than their handles (see Fig. 3b). We use Baxter's wrist camera for visual servoing. We add an external web camera facing the tube to provide the global state. We divide the experiments into two stages. Initially, we evaluate the performance of our object detection algorithm and action execution. Later, we conduct tool use learning experiments.



(a) Scene details (b) All set of tools

Figure 3. Experimental scene

6.1. Object Detection and Execution Experiments

The contours of all objects (cube, tube, and tools) are detected, using Baxter's wrist camera, and their minimum and maximum points are acquired. These are marked with colored dots in the object corners in Fig. 4. Even when the tool is held by the gripper, our algorithm is still able to detect the object. However, the vision system may fail in a low-light environment. It also assumes that all objects are aligned perpendicular to each other and the camera.

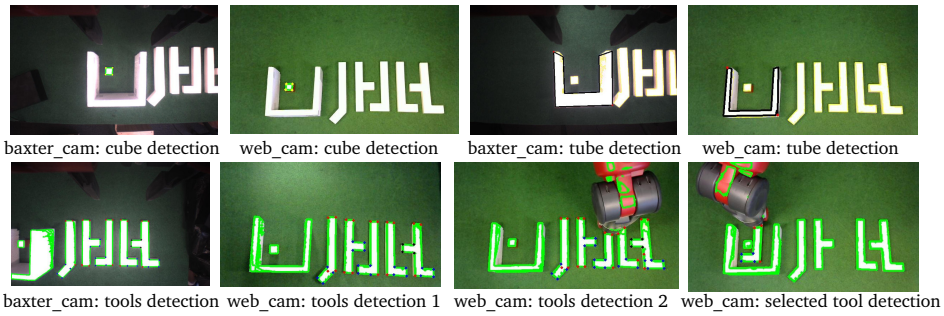


Figure 4. Objects detection in images captured by the Baxter hand camera and the web camera

We also evaluate the capability of the Baxter to move accurately to accomplish a tool use task based on the actions sequence in Table 1. In Fig. 5 we illustrate these actions: `find_tool` action which uses visual servoing, `grip_tool` action, which activates the robot's gripper, and `position_tool` actions, which tries to satisfy a primitive goal given by a constraint solver. From these trials, it was determined that the Baxter can move smoothly to complete the task. We observe that most errors are caused by the perception system, not the action mechanism.

Figure 5. Sample of the robot's actions

6.2. Learning Experiments

In this experiment, we use five different candidate tools with narrow and wide hooks. The cube, as a target object, is located at a particular position inside the tube, which is changed after the Baxter can withdraw it from the tube successfully. We adopt an initial action model which was created by Brown ([1]). Learning by trial and error is then performed to refine the `tool_pose` hypothesis. We stop learning if the robot accomplishes the task three times consecutively. The complete learning sequence is shown in Fig. 6.

We need at least ten examples to learn to perform this task. However, more experiments may be needed if any errors occur (e.g. a change in lighting or an invalid move produced by the IK solver). The first positive example is given by a teacher. In the second example, the robot attempts to imitate the tool and location of the first one, but it fails as the cube location is changed. In the third example, the robot finds that the attachment side of the hook should be the same as the location of the cube inside the tube. Later on, the robot still makes mistakes as it tends to choose a tool with a narrow hook. This `narrower` property is eliminated from h_s in the generalization process in

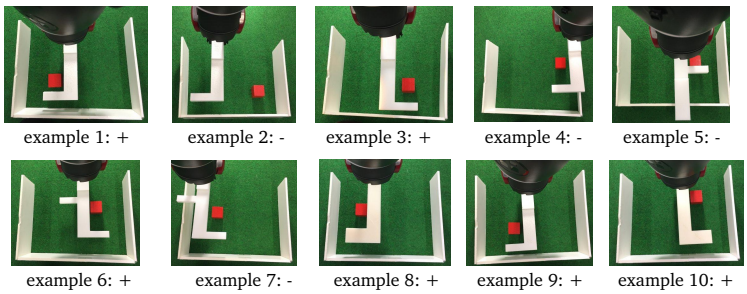


Figure 6. Tool use learning experiments in a real world

the eighth example. Finally, learning is completed in the tenth example, as it performs three consecutive successful experiments. The final hypothesis is shorter (h_s is reduced from 14 literals to 11 literals) and more general than the initial hypothesis.

6.3. Discussions

Our result is similar to the previous relational learning approach [1], they learn tool-use in 12 experiments, as we use the relatively same mechanism. However, we perform our experiment on a real robot, while the former only do it on a sensor-less simulation. This includes bridging the gap between low-level layer (in ROS environment) and high-level layer (in SWI Prolog). We also build objects detection system that has to deal with the noisy environment. Previous work did not do any detections, as it acquires perfect data from a simulator.

Compared to the line of work in feature-based representation, such as work of Tikhanoff et al. [12], our approach can learn faster (theirs needs at least 120 trials in various learning stages) as we can easily incorporate human expertise in the background knowledge. Our experiment scenario is also more complicated, as we locate the target object inside a tube. We exploit the ability of the relational representation to represent a complex relationship between objects compactly. We also learn the tool-pose, where the tool should be located to be able to pull the tool successfully, while it is predefined in their work.

There are limitations in our work, especially in the perception system which can only handle 2D images and not robust in changing environments. In this area, previous work [12, 13] is better than ours. Despite these limitations, the experiment demonstrates that the learning system is capable of relational tool use learning in the real world. In other work of us, we also use a physics simulator to assist a real robot performs tool use learning [18].

7. CONCLUSIONS AND FUTURE WORK

We have developed a complete robot system for relational tool use learning in the real world. The primitive, translator and abstract layers have been built, along with with their interfaces. We have also implemented a system to detect objects and generate primitive behaviors by inverse kinematics, a constraint solver and a visual servoing mechanism. Finally, we have extended a tool use learning system, which has been tested in experiments in the real world. For a relatively simple tool use scenario, the robot at needs at least ten examples to learn the tool's properties.

In the future, we want our robot to do tool creation when the available tools cannot be used to solve the current problem. Those tools can then be tested in the simulator, to save time and materials, and manufactured by using a 3D printer. The development of a more robust perception and a more accurate movement systems will improve our system performance. The system will be tested on a wider variety of tool use scenarios, with different tools and different tasks.

ACKNOWLEDGEMENT

Handy Wicaksono is supported by The Directorate General of Resources for Science, Technology and Higher Education (DG-RSTHE), Ministry of Research, Technology, and Higher Education of the Republic of Indonesia.

REFERENCES

[1] S. Brown and C. Sammut, "A relational approach to tool-use learning in robots," in *Inductive Logic Programming*. Springer, 2012, pp. 1–15.

[2] A. Haber, "A system architecture for learning robots," Ph.D. dissertation, School of Computer Science and Engineering, UNSW Australia, 2015.

[3] S. H. Johnson-Frey, "What's so special about human tool use?" *Neuron*, vol. 39, no. 2, pp. 201 – 204, 2003. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0896627303004240
[4] J. Stckler and S. Behnke, "Adaptive tool-use strategies for anthropomorphic service robots," in *2014 IEEE-RAS International Conference on Humanoid Robots*, Nov 2014, pp. 755–760.
[5] H. Wicaksono and C. Sammut, "A learning framework for tool creation by a robot," in *Proceedings of ACRA*, 2015.
[6] S. Muggleton, "Inductive logic programming," *New Generation Computing*, vol. 8, no. 4, pp. 295–318, 1991.
[7] O. Bachir and A.-f. Zoubir, "Adaptive neuro-fuzzy inference system based control of puma 600 robot manipulator," *International Journal of Electrical and Computer Engineering*, vol. 2, no. 1, p. 90, 2012.
[8] A. R. Firdaus and A. S. Rahman, "Genetic algorithm of sliding mode control design for manipulator robot," *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, vol. 10, no. 4, pp. 645–654, 2012.
[9] H. Wicaksono, H. Khoswanto, and S. Kuswadi, "Behaviors coordination and learning on autonomous navigation of physical robot," *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, vol. 9, no. 3, pp. 473–482, 2013.
[10] A. Wood, T. Horton, and R. Amant, "Effective tool use in a habile agent," in *Systems and Information Engineering Design Symposium, 2005 IEEE*, April 2005, pp. 75–81.
[11] A. Stoytchev, "Behavior-grounded representation of tool affordances," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, April 2005, pp. 3060–3065.
[12] V. Tikhanoff, U. Pattacini, L. Natale, and G. Metta, "Exploring affordances and tool use on the icub," in *2013 13th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, Oct 2013, pp. 130–137.
[13] T. Mar, V. Tikhanoff, G. Metta, and L. Natale, "Self-supervised learning of grasp dependent tool affordances on the icub humanoid robot," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, May 2015, pp. 3200–3206.
[14] R. Fikes and N. J. Nilsson, "STRIPS: A new approach to the application of theorem proving to problem solving," *Artif. Intell.*, vol. 2, no. 3/4, pp. 189–208, 1971.
[15] S. Hutchinson, G. D. Hager, and P. I. Corke, "A tutorial on visual servo control," *IEEE transactions on robotics and automation*, vol. 12, no. 5, pp. 651–670, 1996.
[16] T. M. Mitchell, "Version spaces: A candidate elimination approach to rule learning," in *Proceedings of the 5th international joint conference on Artificial intelligence-Volume 1*. Morgan Kaufmann Publishers Inc., 1977, pp. 305–310.
[17] S. Muggleton and C. Feng, "Efficient induction in logic programs," in *Inductive Logic Programming*, S. Muggleton, Ed. Academic Press, 1992, pp. 281–298.
[18] H. Wicaksono and C. Sammut, "Relational tool use learning by a robot in a real and simulated world," in *Proceedings of ACRA*, 2016.

BIOGRAPHY OF AUTHORS

Handy Wicaksono is a Ph.D. student at Artificial Intelligence Group, School of Computer Science and Engineering, UNSW Australia, with bachelor and master degree in Electrical Engineering from Institut Teknologi Sepuluh Nopember, Indonesia. He is also a lecturer in Electrical Engineering Department, Petra Christian University, Indonesia. His research is in the area of artificial intelligence, intelligent robot, and industrial automation.

Claude Sammut is a Professor of Computer Science and Engineering at the University of New South Wales, Head of the Artificial Intelligence Research Group and Deputy Director of the iCinema Centre for Interactive Cinema Research. Previously, he was a program manager for the Smart Internet Technology Cooperative Research Centre, the UNSW node Director of the ARC Centre of Excellence for Autonomous Systems and a member of the joint ARC/NH&MRC project on Thinking Systems. His early work on relational learning helped to the lay the foundations for the field of Inductive Logic Programming (ILP). With Donald Michie, he also did pioneering work in Behavioral Cloning. His current interests include Conversational Agents and Robotics.