

Inductive learning of surgical task knowledge from intra-operative expert feedback

Daniele Meli, Marco Bombieri, Diego Dall’Alba and Paolo Fiorini

Department of Computer Science, University of Verona, strada Le Grazie 15, 37134, Verona, Italy

Abstract

Knowledge-based and particularly logic-based systems for task planning and execution guarantee trustability and safety of robotic systems interacting with humans. However, domain knowledge is usually incomplete. This paper proposes a novel framework for task knowledge refinement from real-time user feedback, based on inductive logic programming. The user feedback is used to generate cumulative examples which allow a learner to iteratively refine logic rules describing the task. Validated in a benchmark surgical training task, our system is also able to execute and learn unknown actions within reasonable time.

Keywords

Inductive Logic Programming, Incremental learning, Autonomous robots, Surgical robotics

1. Introduction

As robots are becoming increasingly involved in complex domains interacting with humans, safe autonomy is a key requirement. One fundamental capability of autonomous robots is *task planning*, concerning the decision-making process to select a sequence (*plan*) of elementary operations (*actions*), depending on available environmental and robotic *resources*, and *specifications* defining preconditions, effects of actions and constraints. Resources, actions, and specifications constitute *task knowledge*. In particular, the latest regulations for high-risk Artificial Intelligence (AI) systems [1] prescribe *comprehensibility*, i.e., easy readability of robot’s behavior by a human supervisor for transparency and trustability [2]. Among state-of-the-art planning formalisms, logic programming and particularly Answer Set Programming (ASP) [3] is one prominent paradigm for human-readable logic-based task knowledge representation in robotics [4]. However, complete task knowledge is usually unavailable to logic programmers in complex domains, or at least rare workflows of execution or anomalies are missing. For this reason, knowledge must often be refined and updated from intra-operative experience.


In this paper, we propose a preliminary solution based on Inductive Logic Programming (ILP) [5] under ASP semantics [6] (Section 2.2), exploiting feedback from expert users during semi-autonomous task execution. We choose the surgical robotic scenario for validation, which is growing in popularity for its potential benefits in the operating room of the future [7]. In

9th Italian Workshop on Artificial Intelligence and Robotics (AIRO 2022)

✉ daniele.meli@univr.it (D. Meli); marco.bombieri@univr.it (M. Bombieri); diego.dallalba@univr.it (D. Dall’Alba); paolo.fiorini@univr.it (P. Fiorini)



© 2022 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

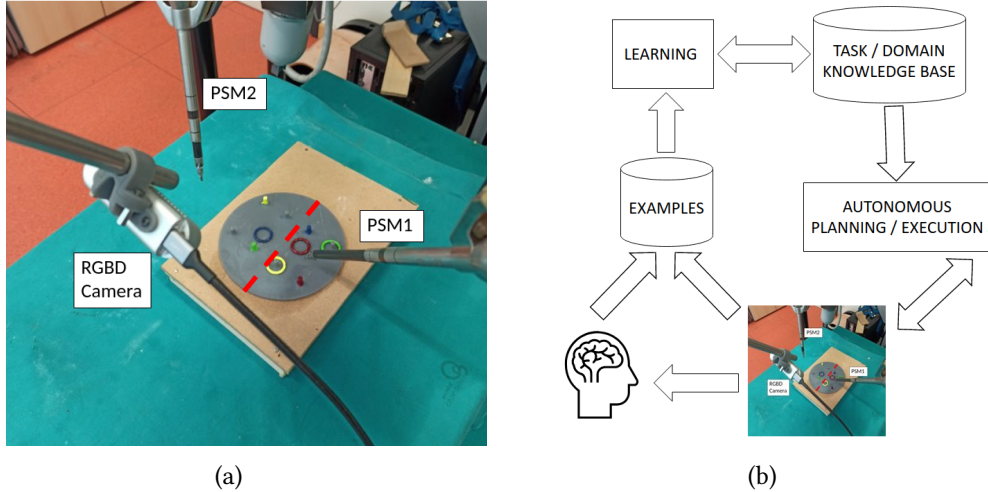


Figure 1: Left, the setup for peg transfer with dVRK; right, the proposed framework for refining task knowledge from intra-operative feedback.

particular, we consider the training task of peg transfer (described in Section 2.1) from the fundamentals of laparoscopic surgery [8], executed with da Vinci Research Kit (dVRK) [9].

ILP has been used by an international research community in different domains, particularly to learn logic programs in AI [10, 11] and surgical robotics [12, 13]. It requires few example executions, thanks to semantic generalization, so it outperforms neural networks and probabilistic approaches in safety-critic scenarios with little data. ILP is mostly used for offline learning, while online knowledge refinement is only a recent trend [14], with no applications to robotics to the best of our knowledge.

2. Background

2.1. The peg transfer task

The setup for peg transfer (Figure 1a) consists of 2 patient-side manipulators (PSMs) of dVRK equipped with graspers, and an external vision sensor (RGB-D camera). The grasping arms operate on up to four colored rings, with the goal to place them on the same-colored pegs. This task is subject to geometric constraints, e.g., each PSM can move only to *reachable* rings and pegs (depending on the relative distance among them). For instance, in Figure 1a, the red-dashed line defines regions of reachability for objects, e.g., the red ring and peg can be reached by PSM1, while PSM2 needs to pick the blue ring and transfer it to PSM1 before placement on the peg. Furthermore, rings may be initially placed on grey pegs, thus extraction may be required.

2.2. Answer Set Programming (ASP)

ASP [3] is a state-of-the-art logic programming formalism for task planning. Following standard syntax from [15], an ASP program represents a domain of interest with a *signature* and *axioms*.

The signature is the alphabet of the domain, defining *variables*, i.e., main entities of the domain (Arm, Object, Color¹ for peg transfer), and *atoms*, i.e., predicates of variables (e.g., `reachable(A, O, C)`), to represent reachability of objects in peg transfer, or actions). Values of variables are *constants* (either integers, Booleans, or strings, e.g., possible colors for C). A term whose value is assigned is *ground*, and an atom is ground if its terms are ground. Axioms define logical relations between atoms. In the peg transfer scenarios, these are task specifications. For instance,

```
grasp(A, ring, C, t) :- at(A, ring, C, t).
:- closed_gripper(A, t), grasp(A, ring, C, t).
```

state that a ring can be grasped if an arm is close to it (*normal axiom*) and that an arm with a closed gripper cannot grasp (*constraint*), respectively. t is a variable for temporal planning, representing a discrete time step. Symbol `:-` is ASP syntax for logic implication \leftarrow .

2.3. Inductive Logic Programming (ILP) for ASP

This section provides salient concepts from the Inductive Learning of Answer Set Program (ILASP), the state-of-the-art tool for learning axioms explaining answer sets [16].

A generic ILP problem \mathcal{T} under the ASP semantics is defined as the tuple $\mathcal{T} = \langle B, S_M, E \rangle$, where B is the *background knowledge*, i.e. a set of axioms, atoms and variables in ASP syntax; S_M is the *search space*, i.e. the set of candidate ASP axioms that can be learned; finally, $E = E^+ \cup E^-$ is a set of *examples*. The goal of \mathcal{T} is to find a subset $H \subseteq S_M$ such that $H \cup B \models E$. Examples are *Context-Dependent Partial Interpretations* (CDPIs), i.e., couples $\langle e, C \rangle$, being e a *partial interpretation* and C the *context*. A partial interpretation is a set of ground atoms, corresponding to actions for the purpose of this paper. Similarly, the context is a set of ground atoms representing environmental variables. Examples may be *positive* ($\in E^+$), i.e., partial interpretations *possibly generated* solving $B \cup H \cup C$, or *negative* ($\in E^-$), i.e., partial interpretations *never generated* solving $B \cup H \cup C$. This means that positive examples typically induce normal axioms (e.g., preconditions and effects of actions), while negative examples generate constraints (*cautious induction*).

ILASP finds the *minimal hypothesis*, i.e., with the least number of axioms and atoms, covering examples.

3. The framework

We propose a framework for refining (surgical) robotic task knowledge during execution, thanks to continuous feedback by human experts supervising the autonomous behavior (see Figure 1b). The basic workflow is the following:

1. a *task/domain knowledge base* B is initialized with available prior information (e.g., main task resources and specifications) in ASP formalism; for complex tasks, prior information can be extracted from domain books [17, 18, 19], or obtained with interviews to experts.

¹From now on, we will denote Arm, Object, Color as A, R, C, respectively, for brevity.

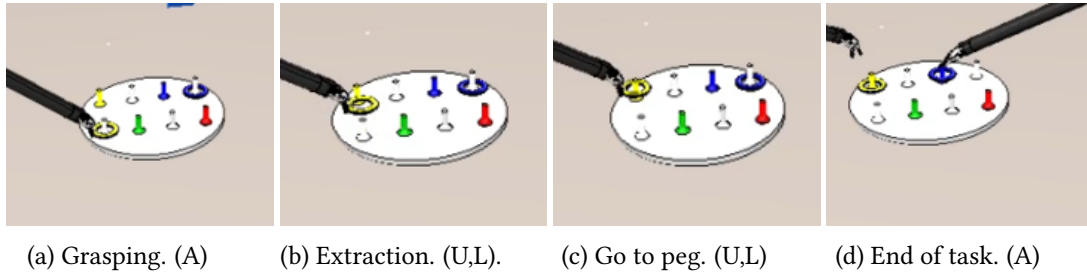


Figure 2: The validation scenario for peg transfer. U=user command; L=learning; A=autonomous.

2. an *autonomous planning/execution* module, e.g., the one proposed in [20, 21, 22], interprets sensor information in order to ground environmental atoms (*context C*) in ASP, and reason on task knowledge to generate a sequence of actions. This module is also in charge of motion planning and execution according to each specific action;
3. before executing each action planned by the reasoner, feedback is asked to an *expert user*, in order to preserve safety. The user may either confirm or reject the proposed action. The user can also propose a specific action to be executed, in case the autonomous planner is not able to find one in the current context;
4. the human feedback, the action, and the context from the planning/execution module, generate a CDPI stored in a list of *examples*. Specifically, each accepted/proposed action corresponds to is saved in E^+ , while rejected actions are in E^- . For computational convenience, we store examples for each action separately;
5. when a negative example is generated, say for action *a*, an ILASP-based *learning* module considers all examples for *a* and *B*, except for axioms including *a*. In fact, these axioms have generated a wrong plan according to the expert user, so they are probably incorrect. Hence, new axioms are generated for *a* and added to *B*. Discarding axioms for *a* does not affect the computational burden of learning since ILASP complexity depends on the size of S_M , hence the number of context atoms.

4. Experimental results

We perform a preliminary validation of our framework in a simulated environment in CoppeliaSim² replicating the dVRK setup for peg transfer and autonomous planning/execution framework with Clingo [23] for ASP solving (plan computation) as in [20]. The human feedback is given to the system via the command line, with the user typing either *yes/no* for confirmation of every single action, or a specific action to be executed as a predicate in ASP syntax. The goal is to refine axioms defining preconditions of actions and constraints.

We assume that the autonomous system is not aware of `extract(A, ring, C)` action, needed when a ring is on a peg before moving it away. We consider the initial scenario in Figure

²<https://www.coppeliarobotics.com/>

2a. After grasping, the autonomous planner proposes `move(psm2, peg, yellow)`. However, the action is infeasible, so the user forbids it and commands `extract(psm2, ring, yellow)`. At this moment, the list of examples contains only a positive example for extraction and one negative for moving to peg. Corresponding ASP axioms are removed from the task knowledge base, and ILASP runs two parallel learning tasks, leading to an empty hypothesis for moving to peg (no positive examples) and the following axiom as a precondition for extraction:

```
extract(A, ring, C, t) :- in_hand(A, ring, C, t).
```

After extraction (Figure 2b), the autonomous system should plan `move(psm2, peg, yellow)`, but axioms for this action have been removed at the previous learning stage. Hence, the user is required to input the action to be executed (Figure 2c), introducing a new positive example for it. Hence, the following axioms can be learned:

```
move(A, peg, C, t) :- reachable(A, peg, C, t).
:- in_hand(A, ring, C, t), on(ring, C, peg, C2, t), move(A, peg, C3, t).
```

with the latter constraints specifying that moving to the peg is not possible when the ring is on another peg. Finally, in Figure 2d the autonomous system can complete the task for the blue ring (executing extraction correctly) with the newly learned axioms. A video of the execution can be found at <https://bit.ly/AIRO2022>. Overall, the learner is invoked 3 times, with a maximum learning time of ≈ 6 s. For each action, S_M includes ≈ 400 axioms.

5. Conclusion

We have presented a system based on user’s feedback to refine existing task knowledge, in the form of a logic program (e.g., an ASP program). Preliminary results in a paradigmatic training task for robotic surgery show that our system is able to propose actions to an expert, execute them only in case of approval (*supervised autonomy for safety*), accept commands of possibly unknown actions from the expert and exploit them to iteratively refine task knowledge for future re-use in autonomous execution.

Future research will focus on the improvement of the existing system, in order to adopt it in a real robotic setup. In particular, learning of motion primitives associated with unknown actions shown by the user will be needed, e.g., with dynamic movement primitives [24]. Moreover, in order to increase the usability of the framework in a real surgical setup, the user should be able to provide feedback via vocal commands. For instance, with reference to Figure 2a the user could state *extract the yellow ring, then move it to the yellow peg*, which could then be interpreted with natural language processing based on ad-hoc linguistic resources (e.g., [18]), and translated to ASP syntax for knowledge refinement. Finally, we remark that learning times with ILASP can be further improved with, e.g., ad-hoc incremental solvers [25].

Acknowledgments

This project has received funding from the European Research Council (ERC) under the European Union’s Horizon 2020 research and innovation programme (grant agreement No. 742671 “ARS”).

References

- [1] Proposal for a regulation of the european parliament and of the council laying down harmonised rules on artificial intelligence (artificial intelligence act) and amending certain union legislative acts, 2021. URL: <https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:52021PC0206>.
- [2] A. B. Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. García, S. Gil-López, D. Molina, R. Benjamins, et al., Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai, *Information Fusion* 58 (2020) 82–115.
- [3] V. Lifschitz, Answer set planning, in: *International Conference on Logic Programming and Nonmonotonic Reasoning*, Springer, 1999, pp. 373–374.
- [4] D. Meli, H. Nakawala, P. Fiorini, Logic programming for deliberative robotic task planning, *Artificial Intelligence Review* (2023) 1–39.
- [5] S. Muggleton, Inductive logic programming, *New generation computing* 8 (1991) 295–318.
- [6] M. Law, A. Russo, K. Broda, The complexity and generality of learning answer set programs, *Artificial Intelligence* 259 (2018) 110–146.
- [7] G.-Z. Yang, J. Cambias, K. Cleary, E. Daimler, J. Drake, P. E. Dupont, N. Hata, P. Kazanzides, S. Martel, R. V. Patel, et al., Medical robotics—regulatory, ethical, and legal considerations for increasing levels of autonomy, *Science Robotics* 2 (2017) 8638.
- [8] N. J. Soper, G. M. Fried, The fundamentals of laparoscopic surgery: its time has come, *Bull Am Coll Surg* 93 (2008) 30–32.
- [9] P. Kazanzides, et al., An open-source research kit for the da vinci® surgical system, in: *2014 IEEE Int. Conf. Robotics and Automation (ICRA)*, IEEE, 2014, pp. 6434–6439.
- [10] A. Cropper, S. H. Muggleton, Learning efficient logic programs, *Machine Learning* 108 (2019) 1063–1083.
- [11] G. Mazzi, D. Meli, A. Castellini, A. Farinelli, Learning logic specifications for soft policy guidance in pomcp, in: *Proceedings of 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, in publication, 2023.
- [12] D. Meli, P. Fiorini, M. Sridharan, Towards inductive learning of surgical task knowledge: a preliminary case study of the peg transfer task, *Procedia Computer Science* 176 (2020) 440–449.
- [13] D. Meli, M. Sridharan, P. Fiorini, Inductive learning of answer set programs for autonomous surgical task planning, *Machine Learning* (2021) 1–25.
- [14] S. Calo, I. Manotas, G. d. Mel, D. Cunnington, M. Law, D. Verma, A. Russo, E. Bertino, Agenp: An asgrammar-based generative policy framework, in: *Policy-based autonomic data governance*, Springer, 2019, pp. 3–20.
- [15] F. Calimeri, W. Faber, M. Gebser, G. Ianni, R. Kaminski, T. Krennwallner, N. Leone, F. Ricca, T. Schaub, Asp-core-2: Input language format, *ASP Standardization Working Group* (2012).
- [16] M. Law, Inductive learning of answer set programs, Ph.D. thesis, University of London, 2018.
- [17] M. Bombieri, M. Rospocher, D. Dall’Alba, P. Fiorini, Automatic detection of procedural knowledge in robotic-assisted surgical texts, *Int. J. Comput. Assist. Radiol. Surg.* 16 (2021) 1287–1295. URL: <https://doi.org/10.1007/s11548-021-02370-9>. doi:10.1007/

s11548-021-02370-9.

- [18] M. Bombieri, M. Rospocher, S. P. Ponzetto, P. Fiorini, The robotic surgery procedural framebank, in: Proceedings of the Thirteenth Language Resources and Evaluation Conference, European Language Resources Association, Marseille, France, 2022, pp. 3950–3959. URL: <https://aclanthology.org/2022.lrec-1.420>.
- [19] M. Bombieri, M. Rospocher, S. P. Ponzetto, P. Fiorini, Machine understanding surgical actions from intervention procedure textbooks, *Comput. Biol. Medicine* 152 (2023) 106415. URL: <https://doi.org/10.1016/j.compbimed.2022.106415>. doi:10.1016/j.compbimed.2022.106415.
- [20] M. Ginesi, D. Meli, A. Roberti, N. Sansonetto, P. Fiorini, Autonomous task planning and situation awareness in robotic surgery, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2020, pp. 3144–3150.
- [21] D. Meli, et al., Autonomous tissue retraction with a biomechanically informed logic based framework, in: 2021 IEEE Int. Symp. on Medical Robotics (ISMR) [in press], IEEE, 2021, pp. 0–7.
- [22] E. Tagliabue, D. Meli, D. Dall'alba, P. Fiorini, Deliberation in autonomous robotic surgery: a framework for handling anatomical uncertainty, in: Proceedings - IEEE International Conference on Robotics and Automation, 2022, pp. 11080–11086.
- [23] M. Gebser, R. Kaminski, B. Kaufmann, M. Ostrowski, T. Schaub, P. Wanko, Theory solving made easy with clingo 5, in: Technical Communications of the 32nd International Conference on Logic Programming (ICLP 2016), Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016, pp. 2:1–2:15.
- [24] M. Ginesi, D. Meli, A. Roberti, N. Sansonetto, P. Fiorini, Dynamic movement primitives: Volumetric obstacle avoidance using dynamic potential functions, *Journal of Intelligent & Robotic Systems* 101 (2021) 1–20.
- [25] M. Law, K. Broda, A. Russo, Search space expansion for efficient incremental inductive logic programming from streamed data, *International Joint Conference on Artificial Intelligence* (2022).