Motion Descriptors for Intention Recognition in Robot Teleoperation Tasks

Mohamed Behery[1] and Gerhard Lakemeyer[1]

Abstract— **Human-robot collaboration as well as real-time robotic assistance can offer an enhanced user experience if the robot is able to recognize the operator's intention. This is particularly important in the case of robot teleoperation using non-traditional input devices such as a brain computer interface. This paper proposes augmenting the PDDL action description with motion descriptors, which are geometric functions to describe motion primitives involved in each action. Using both the symbolic and geometric world state, we are able to filter the feasible actions narrowing down the search space. We conduct a preliminary user study that shows promising results in terms of usability. Our study shows that motion descriptors can be used with ease even by inexperienced users with no programming or robotics background.**

INTRODUCTION

Human robot assistance plays a role in different applications of robotics. Ranging form performing Activities of Daily Living (ADLs) to robot teleoperation in industrial scenarios [1] or on space missions [2]. While some applications allow the robot to be fully autonomous, others require some human intervention and collaboration between a human operator and the robot [3]. The latter applications require knowledge about the operator's intention at different times according to the robot control paradigm. Such knowledge offers not only an enhanced User eXperience (UX) in those applications, but also increases the accuracy of the task execution.

In supervisory control applications an operator explicitly specifies their intention and an agent acts accordingly during operation. For instance in the case of [4], [5] operators specify an action or an affordance template and a robot performs them. In direct human control, a robot doesn't require knowledge about the operator's intention at all. The shared autonomy control paradigm requires different levels of knowledge according to the application. Flight path correction for collision avoidance only requires the intended flight direction while Brain Computer Interface (BCI)-based teleoperation for ADLs such as [6] on the other hand, requires high-level knowledge of the task the user is performing. Teleoperation assistance approaches such as [6], [7] enhance the user input depending on the activity being performed. They mostly study such assistance frameworks in isolation without considering long term plans or the possibility of different actions. On the other hand, [8] considers the intention of the user but is limited in the types of actions that can be performed.

This paper presents an approach for intention recognition aimed at robot arm teleoperation. The approach is based on a representation of actions that uses motion descriptors to describe the motion of the end-effector during the different manipulation actions. The approach uses the CLIPS [9] rule-based production system to represent the agent's knowledge, allowing us to reason about the world state to infer the set of feasible actions. Afterwards, we use the operator's commands to reason about the geometric state of the world and find the action with the highest similarity to the operator's commands. We compare the operator's commands to the motion involved in the different actions and find which actions are most similar to the user's commands. The proposed representation allows a user to describe the motion involved in an action. This provides flexibility when it comes to describing new actions.

The rest of the paper is organized as follows: the paper first discusses the background and state-of-the-art research in the topic of intention recognition. After that, we go into the detail of our proposed action description and how this is used to recognize a user's action. Next we discuss the evaluation of this approach before the paper we conclude with a discussion and an overview of future work on the topic.

BACKGROUND AND RELATED WORK

A lot of work has been done on the topic of plan or goal recognition over the past years. A wide spectrum of applications discuss intention recognition for different uses. Many medical applications can benefit from intention recognition such as elderly care [10], personal assistance robotics [11], and assisted BCI teleoperation of robot arms for ADLs [6].

Computer vision approaches focus on the analysis of videos or images and studying the body motion, poses, and gestures of a human target [12]. Probabilistic approaches exist which don't rely on plan libraries or action models such as [13] which learns vector representations of actions and represents a plan as a string and uses Expectation Maximization (EM) for plan completion, these approaches require training in different environments, which is infeasible in cases of robot teleoperation for ADLs.

Vision and gesture-based approaches, while successful, cannot be applied in situations where the human lacks the

*Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy – EXC-2023 Internet of Production – 390621612.

*This work is partially supported by the EU ICT-48 2020 project TAILOR (No. 952215).

[1]Knowledge-Based Systems Group (KBSG), RWTH Aachen University, Germany, {behery, gerhard}@kbsg.rwth-aachen.de

motor skills to perform gestures or has limited physical capabilities and/or is bound to a wheel chair or a bed which is the case in some medical applications such as [6], [14]. The lack of action models in such approaches also prevents an agent from updating the world state after an action is performed. This is not a problem when the target is action/gesture recognition itself, such as video surveillance or gesture-based camera control. But the world state needs to remain up-to-date in case the agent directly interacts with the environment.

Approaches based on symbolic representations such as [15] handle the problem of plan recognition by assuming an agent is following an optimal plan to reach a goal. They try to find an optimal plan that matches a set of observed actions to reach a goal state. This is not robust to noise in the observed actions or deviations from optimal plans. They later introduce an enhancement [16] where goals aren't eliminated but are rather ranked by probabilities given a set of observations. Even though this method is quite promising, it is only able to find the agent's intended goal, as opposed to the action being performed, i.e., it recognizes the goal $on(bottle, table)$ rather than the action $place(bottle, table)$. The latter allows us to update the agent's belief state without relying on sensor data (which can be noisy or faulty). Moreover, it allows action-specific assistance to be applied such as that presented in [6].

This paper presents an approach that uses symbolic and geometric information about the environment to infer the actions of a human operator. Our approach is aimed at the BCI-based teleoperation of a robot arm in order to perform ADLs.

The approach presented in [6] uses grasp rather than intention recognition. They find a grasp from an object's grasp-set that minimizes a cost function and start to assist the user in reaching that grasp. When the environment contains multiple objects, they use the trajectory of the user's commands along with maximum entropy [17]. One shortcoming of this approach for intention recognition is the assumption that each object can only offer one type of action after they are grasped. Unlike the approach presented in [6], ours uses the action models allowing an object to afford more than one action. Additionally, we incorporate the action model to reduce the search space as well as update the world state. Our approach is also flexible in the sense that it allows the definition of more actions to the objects without programming knowledge. This is possible by using PDDL action definitions combined with a rule-based production system. Some approaches [7] uses a tablet mounted on the robot to allow the operator to select the action before actually starting the execution, such an approach will not be feasible in the absence of traditional input devices.

Planning Domain Definition Language

The Planning Domain Definition Language (PDDL) [18] is a high level language used to describe domains in robotics applications. In these descriptions, actions are represented as a quadruple of action name, parameters, preconditions, and effects as shown in Listing 1

```
:action place
:parameters (?s ?t ?m)
:precondition (and (sponge ?s)
                   (table ?t)
                   (manipulator ?m)
                   (bound ?s ?m))
:effect ((on ?s ?t)
         (free (manip))
         not(bound ?s ?m))
```

Listing 1: PDDL definition for the picking action.

CLIPS Rule-Based System

"C" Language Production System (CLIPS) [9] is an expert system tool developed as a high performance portable production system. CLIPS relies on three building blocks: a knowledge base, a fact list, and an inference engine.

The rules of CLIPS are divided into Left Hand Side (LHS) and a Right Hand Side (RHS), the LHS represents the antecedents of each rule or conditions that control the activation of rules, the RHS is the consequent of the rule and contains a series of procedures to carry out when the rule's LHS is satisfied. In this paper we are interested in the LHS of the rules, we model the world actions as CLIPS rules where the preconditions are in LHS of the rules as used in [19]. We then use its inference engine to determine the set of feasible actions based on the information in the knowledge base. i.e., When a CLIPS rule is activated, it asserts the feasibility of an action so that we can start processing it.

In this work, we use an open-source python expert system strongly inspired by CLIPS. It provides the ability to define predicates and rules as well as the ability to do pattern matching based on a knowledge base.

MOTION DESCRIPTORS

We assume that the operator is a rational agent who knows the conditions and effects of each action. i.e., knows the actions afforded by each object and does not try actions whose preconditions aren't met, they also know how each action would change the world (causing other actions to be feasible or infeasible). We further assume that the operator knows how the end-effector should be moved to perform the different actions. These assumptions allow us to only search for the operator's actions within the symbolically feasible actions. We use a rule-based system to perform the reasoning and filter out the symbolically infeasible actions.

In this implementation we describe the actions using PDDL [18]. We then translate these definitions into CLIPS rules using the method presented in [19]. In this context, we are only interested in the preconditions of the different actions, so we transform those into the LHSs of CLIPS rules. CLIPS Matching capabilities ground these actions and the RHSs of these rules adds the respective actions to the set of feasible actions.

The proposed action description uses motion descriptors which are an abstract representation of the motion to perform actions. It defines how an end effector moves relative to

the attributes of the action's parameters (world objects). Using the motion representation, we are able to measure the similarity of the user's commands to the motions described. An example of this representation is shown in Listing 2. Which indicates that during the action of picking a sponge, the end-effector moves perpendicular to, and in the direction of the sponge's axis.

```
:action pick
:parameters (?s ?t ?m)
:precondition (and  (free ?m)
                    (sponge ?s)
                    (table ?t)
                    (manipulator ?m)
                    not(bound ?s ?_m))
:effect ((bound ?s ?m)
         not (on (?s ?t))
         not (free (?m)))
:motion ((direction–of ?s axis)
          (perpendicular–to ?s axis))
```

Listing 2: PDDL definition for the picking action including the motion extension.

Geometric Similarity Functions

We describe the motion using a set of geometric functions in terms of attributes of objects in the environment. In this paper, we use three functions: parallel-to, perpendicular-to, and direction-of.

These functions represent distance measures between the motion applied by the user and the motion involved in an action. For example, the action of picking a bottle can be described as motion perpendicular to the bottle's *axis* and in its direction. On the other hand, the action of placing a sponge on the table can be described as motion perpendicular to and in the direction of the *surface* of the table. These functions perform some computations based on the user input vector i (the line between the manipulator current position and the one resulting from the user's command) and the given object attributes as follows:

parallel-to(t): If the target t is a line -a bottle's axis, for instance- (represented here by 2 points in Cartesian space), we calculate the angle between the input line and target as seen in Equation 1.

$$\theta = \arccos(\frac{\bar{i} \cdot \bar{t}}{|\bar{i}| \cdot |\bar{t}|}) \tag{1}$$

If t is a plane -e.g., a table's surface- (represented here by 3 points in Cartesian space a, b, and c). We calculate two lines representing the plane as seen in Equations 2 and 3.

$$\bar{l_1} = a - b \tag{2}$$

$$\bar{l_2} = c - b \tag{3}$$

Then compute the normal to the plane n as seen in Equation 4. After that we can calculate the angel between t and n similar to what is seen in Equation 1. After that the distance is calculated as $||\pi - \theta||$, we only need the absolute value not the direction.

$$\bar{n} = \bar{l_1} \times \bar{l_2} \tag{4}$$

perpendicular-to(t): As seen in Equation 1, we calculate the angle between the input line and the target t. After that we can calculate the distance as $||\pi - \theta||$. In case t is a plane, θ is the angle between the normal to the plane $\bar{n}$ and the input line.

direction-of(t): If t is a point, we calculate the change of distance d between t and the manipulator given the user input. This is calculated as equation 5 where m is the manipulator position.

$$v = d(t, i) - d(m, i) \tag{5}$$

If t is a line or a plane, we project the input line on t and recursively call the `direction-of` function given the projection on t. After that we multiply the change in distance v by the euclidean distance between the manipulator position and t (or projection on t). This is done to give an advantage to actions involving closer targets over actions involving farther ones. This doesn't cause problems since the intention update is fast enough to be done with every cycle of user input.

Using this set of functions, we are able to describe multiple actions as seen in listings 2 and 3 describing the pick, wipe, place, and pour actions respectively.

```
:action wipe
:parameters (?s ?t ?m)
:precondition (and (bound ?s ?m)
                   (sponge ?s)
                   (table ?t)
                   (manipulator ?m))
:effect (clean ?t)
:motion ((parallel–to ?t roi) (direction–of ?t roi))

:action place
:parameters (?s ?t ?m)
:precondition (and (bound ?s ?m)
                   (sponge ?s)
                   (table ?t)
                   (manipulator ?m))
:effect (on ?s ?t)
:motion ((perpendicular–to ?t roi) (direction–of ?t roi))

:action pour
:parameters (?b ?c ?l ?m)
:precondition (and (bound ?b ?m)
                   (contains ?b ?l)
                   (bottle ?b)
                   (cup ?c)
                   (liquid ?l)
                   (manipulator ?m))
:effect (contains ?c ?l)
:motion (direction–of ?c pouring–frame)
```

Listing 3: The PDDL definitions used in our study for the wipe, place, and pour actions including the motion extension.

Overview

At startup time, the PDDL action descriptions are translated into CLIPS rules, where the action preconditions are the transformed into LHS of a rule representing this class of actions. At run time, using the pattern matching provided by CLIPS, we are able to ground the actions to have instances of these actions. For example, the action of picking a bottle from a table, is grounded into an instance of picking bottle b from table t. To do this, the set of actions is filtered to

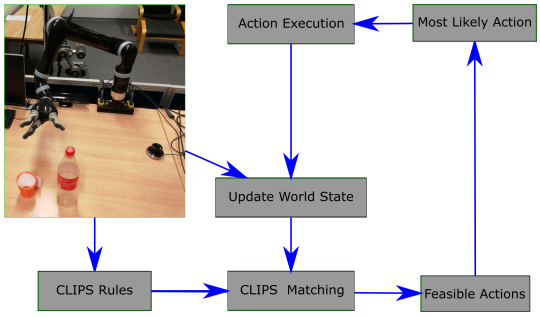

Fig. 1: The proposed framework. The domain information is used to update the world state and the action PDDL definitions are translated into CLIPS rules, then the system is able to infer the feasible actions and perform the geometric reasoning given the motion description to recognize the user's most likely action

give us the set of *symbolically feasible* actions. This filtering step can significantly reduce the search space since it can potentially limit it to a subset of actions afforded by a single object held in the manipulator, or the different instances of the pick action. An overview can be seen in Figure 1.

This framework offers a lot of flexibility due to it's modularity as well as independence from programming knowledge. An operator with no programming or robotics experience is able to describe new actions such that they are recognized without the need for further training.

Adding New Motion Descriptors

We use the three geometric functions mentioned above to evaluate the different actions, but the set of actions can be further extended and refined using more geometric functions as follows:

Adding New Geometric Similiary Function: Geometric functions can be introduced to evaluate more refined motion types such as `rotate-around` to describe actions such as stirring a pot or screwing a screw. Adding geometric functions to the action definitions requires only enough programming knowledge to evaluate the user input, on the other hand, adding new actions or fine-tuning an action using the existing functions doesn't require any programming knowledge. For example, we can define the action of flipping a page in a book to be a motion parallel to the book's surface when the manipulator is free (or holding a book in a 2 manipulator scenario).

```
:action stir
:parameters (?l ?p ?m)
:precondition (and (bound ?l ?m)
                   (pot ?p)
                   (manipulator ?m)
                   (ladle ?l))
:motion (rotate–around ?p ?frame)
```

Listing 4: PDDL definition for the stirring action.

Kinematic Reachability: We can also extend the approach by adding preconditions that are based on the robot's geometry. For instance, we can add reachability preconditions meaning we can only recognize an action if an end position is reachable. This is achievable by integrating a reachability map as mentioned in [20]. This can be useful if our method is used in combination with a BCI-controlled robot arm. On the other hand, we can prevent engaging in an action that would lead the manipulator to an unreachable pose. We can also warn the user through visual, auditory, or haptic feedback about the unreachability of the end pose.

This can be done by adding reachability preconditions to the actions. The pouring action for instance, could provide the PDDL header shown in Listing 5 containing the function `reachable` that can be evaluated using a reachability map and a frame for pouring provided by the cup `c`.

```
:action pour
:parameters (?b ?c ?l ?m)
:precondition (and (bound ?b ?m)
                   (empty ?c)
                   (bottle ?b)
                   (cup ?c)
                   (liquid ?l)
                   (manipulator ?m))
:effect (contains ?c ?l)
:motion (reachable ?c top)
```

Listing 5: PDDL definition for the pour action.

EVALUATION

To evaluate the approach, we performed testing for both functionality and usability. This was done by having users perform 2 tasks. They were asked to perform different activities by teleoperating a robot arm while we evaluated the accuracy of activity recognition. They were also asked to define actions using this approach given the geometrical functions defined above. This section gives details of the study performed.

Experimental Setup

Initially, we asked the study participants to define several ADLs in order to evaluate the usability of such a system. Additionally, A Kinova JACO robot arm[1] was used by the test participants to perform actions. This arm was chosen since it's a commercially available robot arm supporting people with conditions like tetraplegia (and others that limit limb control) in performing ADLs. Normally JACO is controlled by an accompanying 3 DoF joystick which can switch modes between either commanding the robot by moving individual joints, applying force to the end-effector/gripper, or opening and closing the fingers of the gripper. In this study a spacemouse was used to control the arm by applying the 6 DoF commands to the robot's gripper as velocity commands. A kinect sensor was used for perception in combination with AR tags were used for object recognition.

Action Definitions: We asked the participants to use the motion descriptors defined above to describe several ADLs such as flipping a page of a book, opening a drawer, as well as several pick and place tasks. After that, they were asked to fill a subset of the System Usability Scale (SUS)

[1]https://www.kinovarobotics.com/en

TABLE I: A summary the results of the SUS questionnaire used in our study. These statements were rated on a scale from 1 to 5 where 1 means "Strongly Agree", and 5 means "Strongly Disagree".

Question	1	2	3	4	5
The task was mentally demanding	0	2	3	0	1
Actions were easy to describe	4	0	1	0	1
I need the support of a technical person to use this notation	1	0	0	4	1
People would learn to use this system very quickly	0	2	3	0	1
The System was difficult to use	2	2	1	1	0
I need to learn a lot to use this system	0	0	0	2	4

questionnaire. Some questions were eliminated from the study since they are inapplicable for our participants.

Teleoperation: For the teleoperation task, we created an environment containing multiple objects: a bottle, a cup, a sponge, and a table. All the objects can be picked and placed, in addition, the sponge can be used to wipe the table. The participants were first allowed some time (5 minutes) to get used to the teleoperating the arm, they were allowed to move in all directions and test out all the rotations of the hand as well as pick, move, and place a cube using the arm. After the teleoperation tasks, our participants were asked to describe the motion of several actions using the notation introduced here and then rate the process of describing the actions.

Participants: We conducted a pilot study on 6 participants of different backgrounds between 20 and 30 years old (2 females and 4 males). They had previous experience using a joysticks in gaming. One of the participants had used a spacemouse in 3D design tasks. But non of them had experience with robot arm teleoperation in simulation or real life.

Experiment Results

During the teleoperation tasks, the system was able to detect the actions of the users 61% of the time. Which is a modest accuracy, but the errors are explainable and mostly occur in certain actions. It was observed that picking the mug was the mostly incorrectly recognized activity. it was misclassified as picking the sponge. This is due to attempting to pick it from above, i.e., perpendicular to the surface of the table and the sponge. This happened because the tasks were performed in isolation without context or long term plans (such as picking a cup to drink from it).

Other actions that involve the sponge were better recognized, this is because they are more distinctively described. The motion for picking the sponge is a vertical downward motion, as opposed to the horizontal motion involved in picking the cup or the bottle. The same distinction applies between wiping the table and placing the sponge on top of it. The system distinguished between the wiping and placing actions 19 out of 20 times. Picking the sponge was correctly recognized 9 out of 11 times, the remaining 2 times were misclassified as picking the cup. The recognition of other place actions is symbolically recognized since it was the only feasible action once an object was bound to the manipulator.

In the action description task, the users were asked to describe a set of ADLs whose performance requires different kinds of motion. The most interesting results are as follows:

- Picking a bottle from a table:
  All participants agreed on the motion being perpendicular to the axis of the bottle. Some added that it should be in the bottle's direction, and others went on to add that the motion would be parallel to the table.
- Pouring from a bottle to a cup:
  The answers all agreed on a rotation, but varied on what the center of rotation would be.
- Opening a door:
  Two participants describe the motion as a rotational motion around the door's axis of rotation (the line connecting the hinges). Others described it as a motion parallel to the floor or perpendicular to the door. One participant added that the direction should be away from the door.

We can see from the descriptions above that the same type motion can be described in different terms e.g., opening a drawer by pulling parallel to the ground is the same as pulling along its axis. Actions can also be performed differently all together, e.g., selecting the rotation anchor when pouring from a bottle. Allowing the users to describe the actions means that not only the actions would be recognized more easily, but also that we can use these descriptions as constraints while performing the actions. This way the actions would be easier to perform and would feel natural to the users.

After describing the actions, the participants were asked to rate this notation for action description using the SUS questionnaire. Four of them strongly agreed that the actions were easy to describe, the fifth participant indicated having problems describing the action in terms of object properties. It should be noted that the same participant described the actions using the keywords [perpendicular-to, parallel-to, direction-of, ... etc] without any arguments to such functions. e.g., stirring coffee and opening the door were described as a rotational motion, while the others described them as rotation around the cups axis and the door's axis respectively. All five participants strongly indicated that they wouldn't need support in using this notation. They all agreed that such a system was easy to use and that they didn't need to learn a lot of things to use that system. The results of the SUS questionnaire are summarized in Table I.

They were asked to suggest more geometrical functions that could be used to further describe motion. Suggestions indicated 'tilt', 'opposite-to', and 'angle-between'. Which suggests that even without any programming -or robotics- knowledge, one is able to describe motion using geometric functions and even suggest new ones.

CONCLUSION AND OUTLOOK

This paper proposes an action description approach aimed at action recognition for teleoperation actions. The proposed

approach describes the motion involved in the action. During run time, the operator's commands are compared to the motion of the symbolically feasible actions and the one with the most similar motion is the most likely one. In this paper we describe the motion in a human readable fashion using a set of geometric functions whose arguments are attributes of the actions' parameters. We describe 3 of these functions and how they can be used in concert to describe multiple actions. New actions can be added and described using these descriptors, the functions can also be extended with minimal effort to describe more actions or fine tune the existing ones. We also propose an extension to this approach where kinematic reachability can be involved in the decision making.

The approach we present here is aimed at robotic assistance to a human operator, specially in cases where a human controls a robot without using traditional input methods (such as a mouse, a keyboard, or a GUI). One of the advantages of this approach is the ease of adding new actions/objects to the knowledge base using predefined geometric functions, i.e., no training phase is needed to recognize the new actions. Unlike some other approaches [15], this approach doesn't perform any planning during run time and is only dependent on the number of available actions. The approach presented here follows the action models and is therefore able to update the knowledge base after each action is performed. In addition, our approach allows the actions to be customized by the operators themselves, providing each user the ability to have their own customized version of the action without any programming or robotics' knowledge.

Further testing is still planned, particularly scenarios where we allow the participants to describe and action and then test it on the robot. This would resolves the issues of defining the motion using different object attributes. The next milestone for this approach is to investigate its integration in systems such as [8], [7] where a robot arm is controlled by an operator for performing ADLs in shared autonomy where an action is represented as phases and each phase is toggled when some constraints are satisfied geometric constraints. It would be possible to use such descriptor for the different phases of the actions where and integrating intention recognition, action phase switching, and input augmentation.

REFERENCES

[1] M. Behery, M. Tschesche, F. Rudolph, G. Hirt, and G. Lakemeyer, "Action Discretization for Robot Arm Teleoperation in Open-Die Forging," in *Proceedings of the SMC 2020 conference*, 2020.

[2] N. Y. Lii, D. Leidner, A. Schiele, P. Birkenkampf, B. Pleintinger, and R. Bayer, "Command robots from orbit with supervised autonomy: An introduction to the meteron supvis-justin experiment," in *Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction Extended Abstracts*, 2015, pp. 53–54.

[3] R. Baier, H. Dammers, A. Mertens, M. Behery, S. Nouduri, L. Pelzer, A. Shahidi, M. Trinh, B. Corves, T. Gries, C. Hopmann, M. Hüsing, G. Lakemeyer, and V. Nitsch, "A Framework for the Classification of Human-Robot Interactions within the Internet of Production," ser. Lecture Notes in Computer Science. Cham: Springer, June in press.

[4] J. James, Y. Weng, S. Hart, P. Beeson, and R. Burridge, "Prophetic goal-space planning for human-in-the-loop mobile manipulation," in *IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*. IEEE, 2015, pp. 1185–1192.

[5] P. Birkenkampf, D. Leidner, and C. Borst, "A knowledge-driven shared autonomy human-robot interface for tablet computers," in *14th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. IEEE, 2014, pp. 152–159.

[6] K. Muelling, A. Venkatraman, J.-S. Valois, J. E. Downey, J. Weiss, S. Javdani, M. Hebert, A. B. Schwartz, J. L. Collinger, and J. A. Bagnell, "Autonomy infused teleoperation with application to brain computer interface controlled manipulation," *Autonomous Robots*, vol. 41, no. 6, pp. 1401–1422, 2017.

[7] G. Quere, A. Hagengruber, M. Iskandar, S. Bustamante, D. Leidner, F. Stulp, and J. Vogel, "Shared control templates for assistive robotics," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 1956–1962.

[8] M. Behery, "A Knowledge-Based Activity Representation for Shared Autonomy Teleoperation of Robotic Arms," RWTH Aachen, Tech. Rep., Aug. 2016. [Online]. Available: https://elib.dlr.de/105891/

[9] R. M. Wygant, "Clips a powerful development and delivery expert system tool," *Computers & industrial engineering*, vol. 17, no. 1-4, pp. 546–549, 1989.

[10] F. E. Martinez-Perez, J. A. G. Fraga, and M. Tentori, "Artifacts' roaming beats recognition for estimating care activities in a nursing home," in *4th International Conference on Pervasive Computing Technologies for Healthcare (PervasiveHealth)*. IEEE, 2010, pp. 1–8.

[11] J. Sung, C. Ponce, B. Selman, and A. Saxena, "Unstructured human activity detection from rgbd images," in *Robotics and Automation (ICRA)*. IEEE, 2012, pp. 842–849.

[12] R. Poppe, "A survey on vision-based human action recognition," *Image and vision computing*, vol. 28, no. 6, pp. 976–990, 2010.

[13] X. Tian, H. H. Zhuo, and S. Kambhampati, "Discovering underlying plans based on distributed representations of actions," in *Proceedings of the International Conference on Autonomous Agents & Multiagent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 2016, pp. 1135–1143.

[14] L. R. Hochberg, D. Bacher, B. Jarosiewicz, N. Y. Masse, J. D. Simeral, J. Vogel, S. Haddadin, J. Liu, S. S. Cash, P. van der Smagt, *et al.*, "Reach and grasp by people with tetraplegia using a neurally controlled robotic arm," *Nature*, vol. 485, no. 7398, p. 372, 2012.

[15] M. Ramırez and H. Geffner, "Plan recognition as planning," in *Proceedings of the 21st international joint conference on Artifical intelligence. Morgan Kaufmann Publishers Inc*, 2009, pp. 1778–1783.

[16] M. Ramirez and H. Geffner, "Probabilistic plan recognition using off-the-shelf classical planners," in *Proceedings of the Conference of the Association for the Advancement of Artificial Intelligence (AAAI)*, 2010, pp. 1121–1126.

[17] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning." in *AAAI*, vol. 8. Chicago, IL, USA, 2008, pp. 1433–1438.

[18] D. McDermott, M. Ghallab, A. Howe, C. Knoblock, A. Ram, M. Veloso, D. Weld, and D. Wilkins, "Pddl-the planning domain definition language," 1998.

[19] T. Niemuller, T. Hofmann, and G. Lakemeyer, "Clips-based execution for pddl planners," in *Proceedings of the 28th International Conference on Automated Planning and Scheduling (ICAPS), WS on Integrated Planning, Acting and Execution*, 2018.

[20] F. Zacharias, C. Borst, and G. Hirzinger, "Capturing robot workspace structure: representing robot capabilities," in *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS.* Ieee, 2007, pp. 3229–3236.