Integrating Mission, Logistics, and Task Planning for Skills-Based Robot Control in Industrial Kitting Applications

Integrating Mission, Logistics, and Task Planning for Skills-Based Robot Control in Industrial Kitting Applications

Matthew Crosby, Ronald P. A. Petrick
Department of Computer Science
Heriot-Watt University
Edinburgh EH14 4AS, Scotland, UK
{M.Crosby,R.Petrick}@hw.ac.uk

César Toscano, Rui Correia Dias
INESC TEC
Technology and Science
4200 - 465 Porto, Portugal
{ctoscano,rcdias}@inesctec.pt

Francesco Rovida, Volker Krüger
Robotics, Vision, and Machine Intelligence Lab
Aalborg University
Copenhagen 2450, Denmark
{francesco,vok}@m-tech.aau.dk

Abstract

This paper presents an integrated cognitive robotics system for industrial kitting operations in a modern factory setting. The robot system combines low-level robot control and execution monitoring with automated mission and task planning, and a logistics planner which communicates with the factory's manufacturing execution system. The system has been implemented and tested on a series of automotive kitting problems, where collections of parts are picked from a warehouse and delivered to the production line. The system has been empirically evaluated and the complete framework shown to be successful at assembling kits in a small factory environment.

Introduction

The *kitting* paradigm, where collections of parts are gathered and delivered to a production line, is becoming increasingly common in modern factories as it enables customised products to be built one after the other. Currently, kitting is a task that is usually reserved for human workers, leading to long-term injuries from repetitive stress due to the sheer number and weight of parts picked. However, automating this process means addressing a number of challenging problems including autonomous navigation in an open warehouse environment, execution of bin picking and complex part picking, planning of robot actions to complete kitting orders in the required time, and logistics planning to interface with factory systems. This paper presents an integrated cognitive robotics system that has been developed to solve the kitting problem and deal with the aforementioned problems.

The system is split into three main parts: a high-level logistics planner, a low-level robot control architecture, and the automated planning subsystems. The *logistics planner* provides an overview of the state of the entire system and includes a comprehensive *world model*. It also provides access for human workers to oversee operations and interact with planned missions and the world model. The low-level architecture, called *SkiROS* (Rovida and Krüger 2015; Pedersen et al. 2016), maintains robot-level world information for managing action execution. Finally, there are two *planning subsystems* that use standard automated planning techniques: a *task planner* that interfaces with SkiROS to find plans for individual robots, and a *mission planner* which communicates with the logistics planner to provide goal assignments for robots in a robot fleet.

Figure 1: A robot operating in a factory environment using the integrated system. The robot is executing a six-step plan to place two parts in the white kitting box it is carrying.

The robot used for this work is shown in Figure 1. The robot consists of a robotic arm mounted on an automated guided vehicle (AGV) platform. The platform has enough space for two kitting boxes to be filled concurrently. SkiROS, the task planner, and the robot-level control modules are contained on the robot itself. The logistics planner and mission planner are hosted outside of the robot and communicate via Robot Operating System (ROS) services[1] using wifi.

The rest of this paper is organised as follows. First, the related work is discussed, followed by a brief overview of the system architecture. The next three sections then introduce the main components of the integrated system: the logistics planner, the SkiROS system, and the two planning systems. Finally, we describe the results of the in-factory testing of the complete system, and conclude by discussing future work.

Related Work

Integrated robotics systems are becoming more and more complex, with many new architectures being proposed. At

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

the robot-level, hybrid systems with a deliberative high level, a reactive low level, and a synchronisation mechanism to mediate between the other two levels (Firby 1989) are common (Gat 1998; Volpe et al. 2001; Bensalem et al. 2009; Magnenat 2010), with researchers focused on finding appropriate interfaces between declarative high-level reasoning and procedural low-level control. *SkiROS* (Skills-ROS) (Rovida and Krüger 2015), the skills architecture used in this paper, is a hybrid framework following concepts from model-driven software engineering (Schlegel et al. 2015).

Other approaches that seek to bridge the gap between high-level and low-level robot actions include ROSco (Nguyen et al. 2013) and Smach (Bohren and Cousins 2010), which use Hierarchical Finite State Machines. In contrast, approaches like ROSPlan (Cashmore et al. 2015) and the work of (Vaquero et al. 2015) make use of planning. The former requires manual definition of planning domains, while the latter uses a translation approach specific to their application. In our work, the user can define and modify skills on the fly and planning domains are automatically generated.

Knowledge representation also plays a key role in cognitive robotic systems (Vernon, von Hofsten, and Fadiga 2010), especially when defining world models. A prominent example in robotics is the KnowRob system (Tenorth and Beetz 2012; 2013), which combines knowledge representation and reasoning methods for acquiring and grounding knowledge in physical systems. KnowRob uses a semantic library which facilitates loading and accessing ontologies represented in the Web Ontology Language (OWL). Semantic representations of the world scene are stored in order to reason about object positions in space and time, along with models of the robot hardware and the robot skills. A similar approach is presented in (Björkelund et al. 2012; Stenmark and Malec 2013; Bjørkelund and Edstrom 2011) as part of the Rosetta project, which focuses on how skills should be modelled for industrial assembly tasks. Another study in (Huckaby 2014) focused on identifying a precise taxonomy of skills for assembly tasks.

Automated planning has been used for robot control since Shakey (Nilsson 1984). While early approaches separated symbolic planning from other forms of planning like geometric planning, it was recognised that solutions often benefited from a hybrid approach (Cambon, Alami, and Gravot 2009). Recently, robot task planning has become an active research area, with approaches taken from diverse areas such as sampling-based motion planning (Plaku and Hager 2010; Barry 2013), integration of symbolic planning with robot-level processes (Eiter et al. 2006; Dornhege et al. 2009; Erdem et al. 2011; Gaschler et al. 2013), and probabilistic back-chaining (Kaelbling and Lozano-Pérez 2013).

System Architecture

We begin by describing the system architecture used in this work (Figure 2). The logistics planner interfaces directly with the factory's manufacturing execution systems (MES) in order to retrieve information about the current state of the factory environment and the kitting orders to be filled. This includes details of the parts to be collected for the next set of products, and also the expected locations of the parts in the warehouse. The logistics planner also interfaces with the mission planner: when kitting orders are received, they are sent to the mission planner which returns an assignment of the kitting orders (goals) to individual robots in a robot fleet.

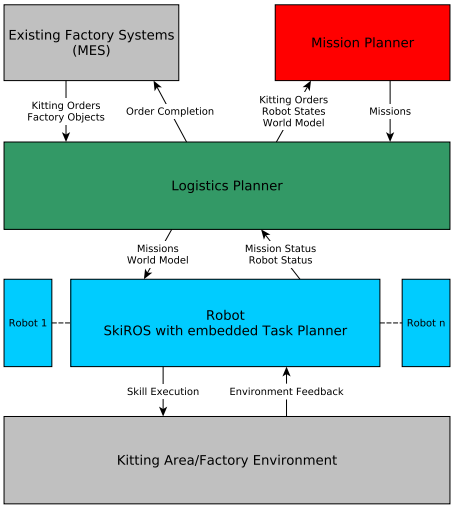

Figure 2: Overview of the system architecture.

The logistics planner also communicates with the SkiROS system on each robot. Communication is implemented via generic ROS services, except for communication between the logistics planner and MES, which must conform to the factory's system interfaces. Once a goal is sent to a specific robot, SkiROS invokes the task planner to find a sequence of skills whose execution should lead to the successful completion of the goal. During skill execution, the internal world model of the robot is constantly updated and replanning is invoked in the case of execution failure.

Both the logistics planner and SkiROS have GUIs designed for easy access to the current system state, and for factory workers to program and update the system. The logistics planner's user interface can be used to configure the warehouse environment, send and receive missions and tasks, and to monitor the system. The SkiROS user interface can be used to add custom skill sequences, monitor and update the robot's world model, and initiate the execution of skills.

The system components are described in detail below.

Logistics Planner

The logistics planner forms the bridge between the robots, the mission planner, and the MES elements. It stores data about the working environment, including identifiers for all physical objects and their three dimensional models. Information is initially compiled from the MES, manual input by a kitting technician, and the states and capabilities of the robots as defined in SkiROS (see below).

The information communicated by the logistics planner mainly concerns the physical objects that are located in the environment. In our application, this equates to shelves, small and large boxes, conveyors, kits, parts, and packaging elements. For each object, its characteristics (e.g., internal organisation, geometry, and spatial location) are modelled in

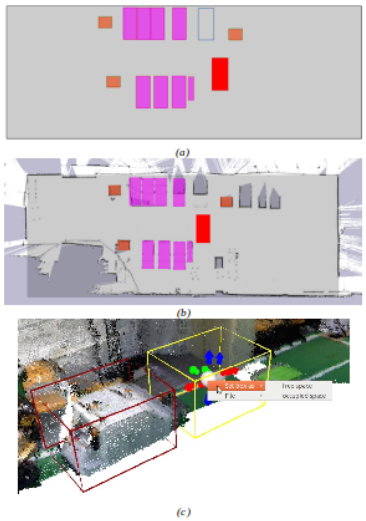

Figure 3: Logistics planner world model modes: (a) 2D CAD image, (b) 2D SLAM image, and (c) 3D point cloud image.

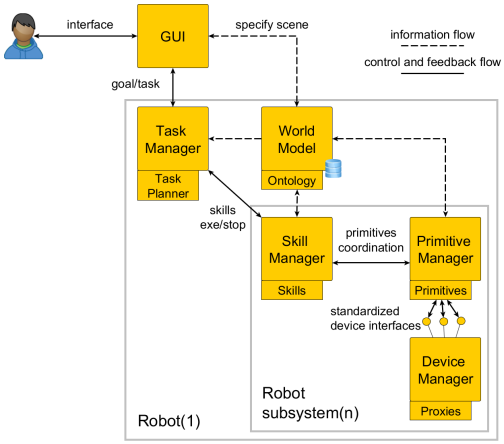

Figure 4: Overview of the SkiROS architecture.

the logistics planner so as to provide this information to the planning and execution components.

The logistics planner's world model specifies the position, orientation, and geometry (in the form of a volume) of all object instances. This is represented through an inverted hierarchical graph, where a node represents a physical object and the link between two nodes the containment relationship between the two corresponding physical objects. The node on the top of the hierarchy is the root node which models the entire space. This region is further decomposed into subregions for different categories of objects (e.g., shelves, large boxes, and conveyors) or to achieve specific purposes (e.g., robot navigation). Below these specific regions are a set of nodes used to model kits (boxes with compartments), large boxes (containers on the ground organised into layers with compartments in each layer), shelves (also structured into layers that contain small boxes), and conveyors. Parts are not normally represented as nodes in order to avoid excessive information in the graph (e.g., a large box containing four layers may have 80 parts) and also because there is no need to identify each part instance. Common characteristics of parts (e.g., the part's 3D model) are represented in the world model to support robot operations. The world model is initially constructed in two phases. First, the representation of these object categories is gathered from an external information system. Second, object instances are created and positioned in the 3D space occupied by the logistics supermarket. This is achieved by a technician using the GUI.

The logistics planner supports three methods for inputting and storing information about the environment (Figure 3). First, a 2D image is created by a CAD application to visually specify the location and orientation of each object in the logistics supermarket (Figure 3a). The red rectangle represents the object being placed whilst the remaining rectangles represent objects already in place. Second, a 2D image created by the robot through its Simultaneous Localization and Mapping (SLAM) functionality can be used to visually specify an object's location and orientation (Figure 3b). Finally, a 3D image of the supermarket is also created to visually specify object locations and orientations (Figure 3c).

SkiROS

A SkiROS component is located on each robot and controls robot-level execution and ontology management. It also communicates status updates and receives mission assignments from the logistics planner. SkiROS is organised into four layers, each of which is represented by a manager. The lowest layer is the *device manager*, which loads proxies (drivers which conform to a standard interface) and presents standard interfaces for similar devices (e.g., gripper, arm, camera, etc.). The second layer contains the *primitive manager* which contains motion primitives, software blocks that realise movement controlled with multi-sensor feedback, and services, software blocks that perform a generic computation. The modules are parameterised and contain a *parameters specification* and *execution* part.

The third layer, the *skill manager*, loads skills (see below) and provides interfaces to the layer above. It also registers the robot subsystem with the world model, specifying the hardware, available modules, and available skills. A skill's execution is usually implemented as a finite state machine which coordinates the execution of several parameterised primitives. Finally, the fourth layer of the architecture is the *task manager* which monitors the presence of subsystems via the world model and acts as a general coordinator. The task manager is the interface for external systems, designed to be connected to a GUI or the MES of a factory.

A central concept in SkiROS is the idea of *robot skills*, which can be thought of as general and robust software constructs that model self-contained, re-occurring operations that a robot might perform. Skills are intended to be designed such that they easily map to simple intuitive tasks. For example, a system might include calibration skills, manipulation skills for operations like picking and placing, as well as driving skills for mobile robots. Skills are implemented

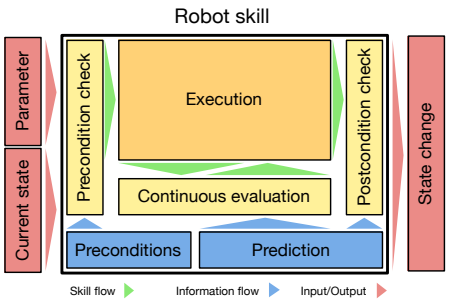

Figure 5: The conceptual model of a SkiROS skill.

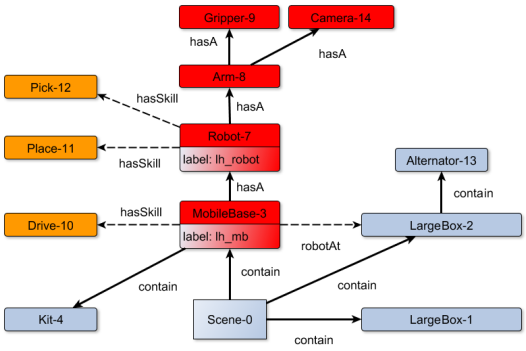

Figure 6: A simplified world model instance with hardware (red), physical (blue), and abstract (orange) objects.

by experts to contain the necessary sensing and action operations for self-contained execution on the robot platform. One benefit of a skills-based system is that non-experts can typically programme a robot task in a straightforward manner by selecting appropriate skill sequences that result in the desired state changes to the robot's environment. Skill sequences can also be constructed in a completely automated way using planning techniques, as discussed below.

The conceptual model of a robot skill is shown in Figure 5. A skill takes as input a set of parameters and a representation of the world state; it outputs a set of state changes. A skill contains both precondition and postcondition checks which monitor the environment, either through sensing or based on the world model. These checks allow the task layer to infer the likely causes of execution failures. For example, a precondition check for a pick skill might be that the item to be picked must be visible to a camera, and a postcondition check might be that the picked item must be in the gripper.

In addition to skills, a key part of SkiROS is the *world model*, which acts as a knowledge integration framework. The world model is a vertical cross-layer component which links all layers together by gathering information from every subsystem at run time, allowing the modules to maintain a shared working memory, and storing the environment and skills information that are used to create the planning domain. In terms of the architecture, the world model can be read and modified by almost every part of the system.

The world state is partially defined by a human operator in the ontology, partially abstracted by the robot using perception, and completed with the procedural knowledge embedded in the skills and primitives. It is originally populated with the robot skills knowledge, robot-specific knowledge such as grasp poses or parameter settings, and with the world model provided by the logistics planner. Each skill manager in the system is responsible for keeping the world model updated with its subsystem information (e.g., hardware, available primitives, skill state, etc.). Similarly, each primitive and skill can extend the scene information with the results of robot operation or sensing. In special cases, the ontology can be extended automatically by the robot to learn new concepts (e.g., a new grasping pose).

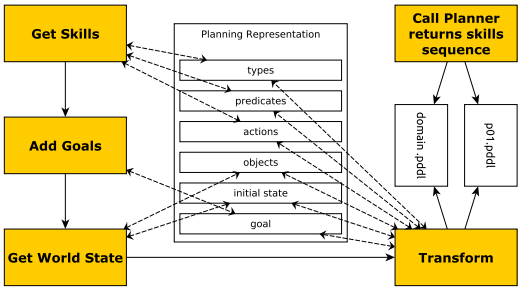

Figure 7: Overview of the task planning process and creation of the internal planning representation.

Figure 6 shows an example of a world model. The tree constructed by the spatial relations forms the scene graph, a data structure commonly used by modern computer games to arrange the logical and spatial representation of a graphical scene. In this structure, an object's pose is always defined with respect to the parent frame. The skills are connected to robot elements by the non-spatial relation *hasSkill*.

Automated Planning

Two different planning components are used in the system: a task planner and a mission planner. The task planner forms part of SkiROS and finds skill sequences for robots to complete their missions. The mission planner distributes missions to individual robots in the fleet based on input from the MES and the world model, mediated by the logistics planner.

The task planner automatically generates its domain from the world model and skill definitions in SkiROS. The task planner has three main functions: it creates a PDDL (McDermott et al. 1998) model of the skills, current state, and goals; it calls an external planner to attempt to find a plan to achieve the current goals; and, if a plan is found, it returns the plan as a sequence of skills to the task manager. The task planner is fully integrated into SkiROS and is called whenever the robot has a goal that must be achieved, either manually by an operator or automatically on mission assignment, or when skill execution fails and replanning is required.

Figure 8 shows the parameters, preconditions, and postconditions for two skills (Drive and Pick), as defined in SkiROS, based on world model constraints and updates. Relations or properties for which postcondition checks are

Drive(MobileBase, Container) :

add: RobotAt(Container, MobileBase)

Pick(Gripper, Object, Container) :

pre: empty(Gripper)
pre: robotAt(Container, Robot)
pre: objectAt(Container, Object)
del: empty(Gripper)
add: contains(Gripper, Manipulatable)

Figure 8: Skill definitions in the SkiROS ontology (modified readable version of exact specification in code).

Algorithm 1: Planning Domain Creation

Input : SkiROS World Model (wm), Goals ($goal$)
Output Initial Planning Representation
:

```
// Parse Skills
foreach Skill s : wm do
    types.addAllNewTypes(s)
    predicates.addAllNewPredicates(s)
    actions.addNewAction(s)
// Add Goal State
foreach Goal g : goal do
    goals.add(g);
// Parse World Model State
foreach Predicate p: predicates do
    initState.addAllTrueGroundings(p, wm)
    objects.addAllNewObjects(p,wm)
```

expected to be true (similarly, false) become add (similarly, delete) effects. Action parameters are created from the inputs defined for the skill's execution block.

An overview of the task planning process is shown in Figure 7. The central part shows the task planner's planning library, which contains all the structures needed to create a PDDL planning problem from the world state, skills, and goals. Specifically, it includes structures for types, predicates (both ground and unground), actions, and (typed) objects.

The process of creating the initial planning representation is given in Algorithm 1, which represents the left hand side of Figure 7 and involves three main steps. The first step is to parse the skills that exist in the world model. This involves adding all types and predicates that appear in the skills definitions to the planning library and also creating an action for each skill, which has a direct copy of the preconditions and effects. All relations, properties, and types that do not appear in a skill are therefore not included in the planning library. The process of iterating through the skills and querying the world model to find true predicates may result in a planning library with less elements and types than in the world model. The omitted data can be safely ignored (and an error given for an incorrect goal) due to the following:

Lemma 1: *Any object with a type that does not appear in a skills definition can never appear in a solution plan.*

Proof Sketch: It is impossible to change the truth value of a predicate that does not appear in an action's effects, and the truth value of a predicate that does not appear in any action's preconditions can never be required for a change in state.

Algorithm 2: Planning Domain Refinement

Input : Initial Planning Representation
Output Final Planning Representation
:

```
// Add Capabilities
foreach Action a : actions do
    predicates.add(can_a ?robot)
    a.pre.add(can_a ?robot)
    foreach Robot r : hasSkill(a, r) do
        initState.add(can_a r)
// Spatial Relation Constraints
foreach Action a do
    foreach Spatial Relation S(o,s) ∈ a.add do
        if ∄S ∈ a.pre AND ∄S ∈ a.del then
            s.params.add(x, s.type)
            s.pre.add(S(o,x))
            s.del.add(S(o,x))
        else if ∄S ∈ a.pre then
            s.pre.add(S(o,s))
        else if ∄S ∈ a.del then
            s.del.add(S(o,s))
```

```
(:action drive
   :param (?R - Agent ?T - Location
     * ?preT - Location)
   :pre (and
     * (can_drive ?R)
     * (RobotAtLocation ?R ?preT))
   :eff (and
     * (not (RobotAtLocation ?R ?preT))
       (RobotAtLocation ?R ?T)))
```

Figure 9: The Drive skill after translation to PDDL. The asterisked (*) lines are added by the translation.

The right hand side of Figure 7 encodes implicit properties of the world model and system. The first part makes sure that skills are only usable by the correct elements, by querying the **hasSkill** relation from the world model. For each action, a new predicate `(can_a ?robot)` is added to the planning description. This predicate is added as true to the initial state for each robot that can perform a particular skill, and is invariant. An extra precondition `(can_a ?robot)` is also added to each action so that it can only be instantiated by the correct robots. If the `Robot` parameter is missing from the skill definition then it is added to the action parameters.

The second part adds any preconditions and delete effects that are necessary to maintain the tree structure of the spatial relations in the world model. SkiROS contains methods for internally updating its world model so that it remains consistent, and these methods need to be modelled explicitly in the planning domain. For instance, with respect to the skills in Figure 8, the Drive skill only contains a single predicate which specifies the new location of the robot. This is because the input for the execution block of the Drive skill is only the goal location to which the robot has to move. The `drive` action must then be modified so that `robotAt` is true of only one grounding for the robot performing the Drive skill, so the

Mission type	Attempts	System successes	Execution successes	Detected fails	Undetected fails	Average time (s)
3 parts	15	15	4	8	3	310
4 parts	22	22	3	15	4	380
5 parts	2	2	0	2	0	-

Table 1: Results from testing the complete system in a factory environment.

old instantiation must be found (it becomes a precondition) and added as a delete effect of the action.

Figure 9 shows the skills from Figure 8 after translation to PDDL. Note that in terms of implementation, the parameter added to the Drive skill is removed when returning the parameterised skill to the task manager. The translation adds three new preconditions and two new delete effects over the two actions. The following lemma shows that these additions ensure the world model's tree structure is maintained:

Lemma 2: *Performing an action created by the task planner on a problem whose spatial relations form a tree will result in a state in which the spatial relations still form a tree.*

Proof Sketch: All that needs to be shown is that any deletion of a spatial relation property inserts it elsewhere with the same object (and therefore moves the whole subtree), and that every addition has a corresponding deletion. The former is a constraint on the skill definition. For the latter, every time a new spatial relation appears in the add effects then, by construction of the algorithm, a spatial relation with the same subject must appear in the delete effects. This spatial relation must match the only occurrence of that object in a spatial relation in the current state otherwise the action could not be performed as this must exist (again by construction) as a precondition to the action.

Once the translation is complete, the planning problem is written to domain and problem files in PDDL for use with an external planner. The planner's output (a sequence of instantiated actions) is parsed and converted back to parameterised skills to be sent to the task manager for robot execution.

The mission planner takes goal and world state information from the logistics planner and returns goal assignments (missions) for each robot in the fleet. The mission planner therefore needs to be able to perform planning for multi-robot domains with large numbers of agents. These types of domains are generally hard to solve for traditional classical planning approaches so a multiagent planner (Crosby 2015) is used with a custom domain to find its plans.

As the mission planner must find plans for the entire robot fleet, creating the planning domain manually is necessary in order to remain scalable as the system increases the number of robots. Some of the more fine-grained information available in the world model is abstracted away (e.g., information used by the on-board robot task planners to find the exact skill sequences), since the mission planner only needs to return a viable assignment of goals. In particular, the mission planner focuses on making sure that plans are achievable and assigned to the most appropriate robots, not with skill orderings. For specific implementations, depending on the domain sizes, complexity, and (crucially) the number of robots, it might be possible to use the task planner's automatically generated domain, or necessary to abstract even more. In either case, the interfaces to the logistics planner remain the same.

Experiments

The complete system was deployed in a real-world environment (Figure 1) at a PSA Peugeot Citroën factory in Rennes, France. The results of the experimental testing are presented in Table 1. The tests were carried out using simulated kitting orders as input. The system success rate shows the success of the system, ignoring any errors that came from robot execution modules. In all cases, the mission planner successfully found mission assignments, and the task planner found correct sequences of skills to complete the missions.

The failure points in the system came from robot execution code for individual skills, which (for this implementation) are still undergoing design change. In some failure cases, replanning was able to recover from a failure in execution, though these cases are still marked as failures in the results. Overall, the results show that the system architecture is sound, but the execution code for the skills needs to be improved. Due to the modular nature of the system design, the skill execution code can be updated without affecting the logistics planner, automated planning, or high-level SkiROS components.

Conclusions

This paper described a fully implemented and integrated system for autonomous robots in an industrial factory setting. The system uses planning and a skills model called SkiROS to bridge the gap between low-level robot control and high-level planning. A logistics planner mediates between the factory MES and the mission planning component. Skills are explicitly defined, while a task planner automatically generates the corresponding PDDL planning domain. The resulting system has been shown to operate successfully in a factory environment. From an end-user perspective, the robot is programmed to perform new tasks by specifying goal conditions; new skills are added by specifying constraints on the world model with no explicit knowledge of planning required.

Work is progressing to test more skill implementations and further explore the relationship between skills and planning. Failure handling will be improved to allow for replanning in the case of unsatisfied pre/postconditions and execution failures. To optimise cycle time, the assumption of sequential skill execution will also be relaxed, allowing parallel skill execution from temporal plans.

Acknowledgements

The research leading to these results has received funding from the European Union's Seventh Framework Programme under grant no. 610917 (STAMINA, `stamina-robot.eu`).

References

Barry, J. L. 2013. *Manipulation with Diverse Actions*. Ph.D. Dissertation, MIT, USA.

Bensalem, S.; Gallien, M.; Ingrand, F.; Kahloul, I.; and Nguyen, T.-H. 2009. Designing autonomous Robots: Toward a more dependable software architecture. *IEEE Robotics & Automation Magazine* 16(1):67–77.

Bjørkelund, A., and Edstrom, L. 2011. On the integration of skilled robot motions for productivity in manufacturing. In *IEEE International Symposium on Assembly in Manufacturing*.

Björkelund, A.; Malec, J.; Nilsson, K.; Nugues, P.; and Bruyninckx, H. 2012. Knowledge for Intelligent Industrial Robots. In *Proceedings of the AAAI Spring Symposium on Designing Intelligent Robots: Reintegrating AI*.

Bohren, J., and Cousins, S. 2010. The SMACH High-Level Executive. *IEEE Robotics & Automation Magazine* 17(4):18–20.

Cambon, S.; Alami, R.; and Gravot, F. 2009. A hybrid approach to intricate motion, manipulation and task planning. *International Journal of Robotics Research* 28(1):104–126.

Cashmore, M.; Fox, M.; Long, D.; Magazzeni, D.; Ridder, B.; Carrera, A.; Palomeras, N.; Hurtos, N.; and Carreras, M. 2015. ROSPlan: Planning in the robot operating system. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*.

Crosby, M. 2015. ADP: an Agent Decomposition Planner CoDMAP 2015. In *Proceedings of the ICAPS Competition of Distributed and Multi-Agent Planners (CoDMAP)*.

Dornhege, C.; Gissler, M.; Teschner, M.; and Nebel, B. 2009. Integrating symbolic and geometric planning for mobile manipulation. In *IEEE International Workshop on Safety, Security and Rescue Robotics (SSRR)*, 1–6.

Eiter, T.; Ianni, G.; Schindlauer, R.; and Tompits, H. 2006. Effective integration of declarative rules with external evaluations for semantic-web reasoning. In *The Semantic Web: Research and Applications*, 273–287.

Erdem, E.; Haspalamutgil, K.; Palaz, C.; Patoglu, V.; and Uras, T. 2011. Combining high-level causal reasoning with low-level geometric reasoning and motion planning for robotic manipulation. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

Firby, R. J. 1989. *Adaptive Execution in Complex Dynamic Worlds*. Ph.D. Dissertation, Yale University, USA.

Gaschler, A.; Petrick, R. P. A.; Giuliani, M.; Rickert, M.; and Knoll, A. 2013. KVP: A knowledge of volumes approach to robot task planning. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems IROS*, 202–208.

Gat, E. 1998. On three-layer architectures. In *Artificial Intelligence and Mobile Robots*. MIT Press.

Huckaby, J. 2014. *Knowledge Transfer in Robot Manipulation Tasks*. PhD thesis, Georgia Institute of Technology, USA.

Kaelbling, L. P., and Lozano-Pérez, T. 2013. Integrated task and motion planning in belief space. *International Journal of Robotics Research* 32(9–10):1194–1227.

Magnenat, S. 2010. *Software integration in mobile robotics, a science to scale up machine intelligence*. PhD thesis, École polytechnique fédérale de Lausanne, Switzerland.

McDermott, D.; Ghallab, M.; Howe, A.; Knoblock, C.; Ram, A.; Veloso, M.; Weld, D.; and Wilkins, D. 1998. PDDL – The Planning Domain Definition Language (Version 1.2). Technical Report CVC TR-98-003/DCS TR-1165, Yale Center for Computational Vision and Control.

Nguyen, H.; Ciocarlie, M.; Hsiao, K.; and Kemp, C. C. 2013. Ros commander (rosco): Behavior creation for home robots. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 467–474.

Nilsson, N. J. 1984. Shakey the robot. Technical Report 323, AI Center, SRI International.

Pedersen, M. R.; Nalpantidis, L.; Andersen, R. S.; Schou, C.; Bøgh, S.; Krüger, V.; and Madsen, O. 2016. Robot skills for manufacturing: From concept to industrial deployment. *Robotics and Computer-Integrated Manufacturing* 37:282–291.

Plaku, E., and Hager, G. D. 2010. Sampling-based motion planning with symbolic, geometric, and differential constraints. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 5002–5008.

Rovida, F., and Krüger, V. 2015. Design and development of a software architecture for autonomous mobile manipulators in industrial environments. In *IEEE International Conference on Industrial Technology (ICIT)*.

Schlegel, C.; Lotz, A.; Lutz, M.; Stampfer, D.; and Vicente-Chicote, C. 2015. Model-Driven Software Systems Engineering in Robotics: Covering the Complete Life-Cycle of a Robot. *it - Information Technology* 57(2):85–98.

Stenmark, M., and Malec, J. 2013. Knowledge-based industrial robotics. *Scandinavian Conference on Artificial Intelligence*.

Tenorth, M., and Beetz, M. 2012. Knowledge Processing for Autonomous Robot Control. *Proceedings of the AAAI Spring Symposium on Designing Intelligent Robots: Reintegrating AI*.

Tenorth, M., and Beetz, M. 2013. KnowRob: A knowledge processing infrastructure for cognition-enabled robots. *The International Journal of Robotics Research* 32(5):566–590.

Vaquero, T.; Mohamed, S. C.; Nejat, G.; and Beck, J. C. 2015. The implementation of a planning and scheduling architecture for multiple robots assisting multiple users in a retirement home setting. In *AAAI Workshop on Artificial Intelligence Applied to Assistive Technologies and Smart Environments*.

Vernon, D.; von Hofsten, C.; and Fadiga, L. 2010. *A Roadmap for Cognitive Development in Humanoid Robots*. Springer.

Volpe, R.; Nesnas, I.; Estlin, T.; Mutz, D.; Petras, R.; and Das, H. 2001. The CLARAty architecture for robotic autonomy. In *IEEE Aerospace Conference Proceedings*.