A Layered Control Approach to Human-Aware Task and Motion Planning for Human-Robot Collaboration

Marco Faroni[1], Manuel Beschi[2], Stefano Ghidini[1,2], Nicola Pedrocchi[1],
Alessandro Umbrico[3], Andrea Orlandini[3], Amedeo Cesta[3]

Abstract— **Combining task and motion planning efficiently in human-robot collaboration (HRC) entails several challenges because of the uncertainty conveyed by the human behavior. Tasks plan execution should be continuously monitored and updated based on the actual behavior of the human and the robot to maintain productivity and safety. We propose control-based approach based on two layers, *i.e.*, task planning and action planning. Each layer reasons at a different level of abstraction: task planning considers high-level operations without taking into account their motion properties; action planning optimizes the execution of high-level operations based on current human state and geometric reasoning. The result is a hierarchical framework where the bottom layer gives feedback to top layer about the feasibility of each task, and the top layer uses this feedback to (re)optimize the process plan. The method is applied to an industrial case study in which a robot and a human worker cooperate to assemble a mosaic.**

I. Introduction

Robotized assembly and manufacturing are evolving towards a human-centered paradigm [1] but, even if humans improve the flexibility and performance of the system, the complexity underlying task and motion planning activities increases [2]. In Human-Robot Collaboration (HRC), the problem complexity is high even for a task composed of a few activities: a simple task can be performed through many robot trajectories and each of them could be performed simultaneously with different human tasks. Also, the robot's execution time may be different than expected, as the robot may slow down for safety reasons [2]. Therefore, HRC makes the integration of task and motion planning difficult without relying on limiting hypothesis and applicability contexts [3]. To cope with such complexity, many works focus on the identification of a feasible solution and not the optimal one. Such methods are mainly structured into three groups depending on the task planning strategy: PDDL [4], hierarchical task networks [5], [6], and constraint satisfaction problems (CSP) [7]. These methodologies, however, may achieve low performance in industrial application, where at least a sub-optimal feasible solution is necessary. Few works have also addressed the identification of an optimal task and motion planning via, e.g., a logic-geometric programming approach [8]. Such approach, however, does not scale to large number of tasks, and it cannot manage the complexity of industrial tasks. Except for [2], [3], all the approaches in literature do not deal with time-variability and constraints characterizing HRC tasks. On the one hand, there are many constraints to be considered in order to safely perform robot tasks. Specifically, the execution of robot tasks must take into account simultaneous tasks the human may perform to avoid collisions and risks. On the other hand, a coordinated task plan should be generated and concurrently performed by humans and robots with the aim of increasing the production efficiency and safely supporting the human. An interesting hierarchical approach that could be extended to HRC is presented in [9]. A co-optimization approach explicitly models the robot transition cost from one task to another, and splits the optimization problem over three layers: task planning, action planning and motion planning. The methodology allows the optimization of the plans to reach the robot final configuration and exploits the information coming from the low-level motion planner. However, [9] does not consider temporal information and the likely concurrent execution of human and robot tasks, and the task planner uses a Travel Salesman Problem formulation that does not fit the specification of a generic industrial task. An underrated issue is the integration of the feedback coming from the field for the update of the motion and task planning and execution [3]. Indeed, task and motion planning architectures should also consider to dynamically change the motion plan to be compliant with human movements.

This work aims to extend [3] investigating a control-based approach to the optimization of the task and motion planning in HRC. Starting from the formalization of the problem as in [9], we propose a layered control approach for HRC at different levels of granularity that maximizes throughput in terms of process execution time. The work is also motivated by the H2020 project ShareWork (http://www.sharework-project.eu), which aims at developing safe and effective innovative HRC solutions in manufacturing.

The paper is organized as follow: Section II deals with the description of the methodology, Section III describes the simulation of a case study, and Section IV reports some points of discussion and the future work description.

II. Human-Aware Control Approach

Fig. 1 shows an overview of the proposed control architecture, split in two main modules: a task planner and an action planner (which, in turn, integrates the motion planner).

[1] STIIMA-CNR - Institute of Intelligent Industrial Technologies and Systems, National Research Council of Italy {name.surname}@stiima.cnr.it
[2] Dipartimento di Ingegneria Meccanica e Industriale, University of Brescia {name.surname}@unibs.it
[3] ISTC-CNR - Institute of Cognitive Sciences and Technologies, National Research Council of Italy {name.surname}@istc.cnr.it

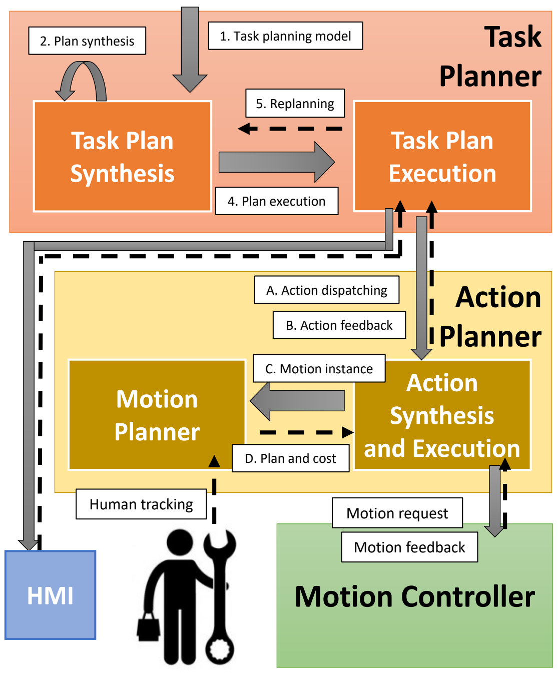


Fig. 1: Task and motion planning integration overview.

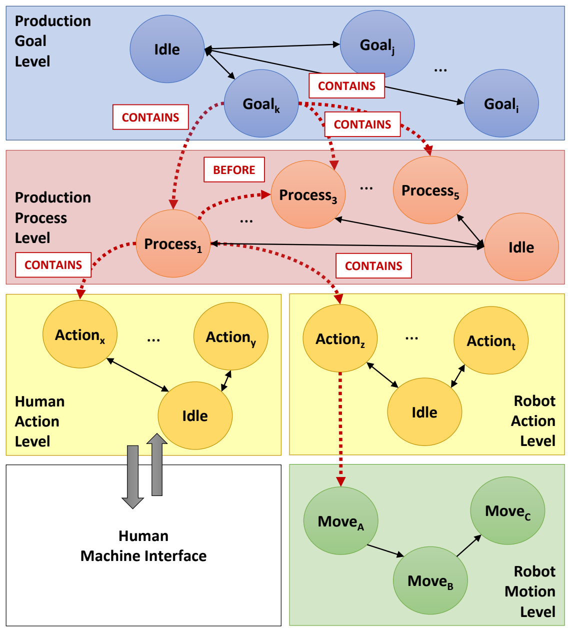


Fig. 2: Hierarchical model of the case study.

The Task Planner addresses human-robot coordination and task sequencing and assignment. It generates a suitable task sequence dealing with the temporal variance entailed by the human presence. The Task Planner sends the goals to a human operator, for example through a Human-Machine Interface (HMI). In addition, it receives feedback from the HMI, as operators can accept or discard the incoming tasks, and also inform the system about the outcome of the performed task (successfully finished or terminated with errors). This feedback is fundamental as the controller cannot assume the duration of human tasks but it needs to wait for a feedback. Pursuing a Sense-Plan-Act approach, the task planner is capable of dynamically (re)planning the sequence of tasks according to the feedback and the (possible) refusal of task from the operator. The robot Action Planner computes the optimal sequence of motion plans that satisfies the operational constraint. Remarkably, the actual human position is considered as feedback when the motion plans are computed. Once the action is defined, the Action Planner sends the trajectory to be executed to the robot motion controller. The combination of these modules realizes flexible robot behaviors that can also be dynamically adapted according to the observed behaviors of a human operator, limiting negative effects on the production flow, when unforeseen events occur.

A. Task Planner

The Task Planner relies on Timeline-based Planning and Scheduling (P&S) [10], a temporal planning paradigm that takes inspiration from the classical Control Theory and has been successfully applied in many real-world applications (*e.g.*, [11]). A timeline-based model is composed of a set of *state variables* describing the possible temporal behaviours of domain features. Each state variable specifies a set of *values* that represent the states or actions the related domain feature may assume or perform over time. Each *value* is also associated with a *flexible duration* and a *controllability tag* which specifies whether the value is controllable or not (from the "controller perspective"). A *state transition function* characterizes the allowed sequences of values of a state variable, i.e., the set of temporal behaviors of a domain feature that are valid with respect to the domain specification (*i.e.*, a *timeline*). In order to coordinate different domain features and realize complex functionalities or achieve complex goals, a set of rules called *synchronizations* model such "global" constraints that must be satisfied to build valid plans. A timeline represents an *envelope* of valid temporal behaviors of a particular domain features. A plan is constituted by a set of timelines (one for each state variable) and represents a set of temporal behaviors (*i.e.*, *solutions*) that can be adapted according to the observations/feedback received from the environment during execution. The execution of a timeline-based plan consists in selecting the particular temporal instance of a plan which best fits the observed state of the environment.

The Task Planner is implemented through PLATINUm [12], a timeline-based software framework for P&S able to deal with *temporal uncertainty* and already successfully deployed in real-world manufacturing scenarios [2]. A planning synthesis module (*Task Plan Synthesis* in Fig. 1) synthesizes flexible behaviors of domain features (i.e., the timelines). An executive module (*Task Plan Execution* in Fig. 1) dispatches the tasks composing the timelines and checks the consistency of a plan with respect to the actual state of the environment. PLATINUm pursues a general solving behavior can be tailored to the particular features of a planning problem through the use of specifically designed *heuristics* and *search strategies*. The former supports the solving algorithm in selecting the flaw to solve at each iteration. This choice does not affect the completeness of algorithm but it is particularly crucial for its efficiency. The latter supports the solving algorithm in selecting which partial plan to refine at next iteration among the ones available in the search space (*search space expansion*). This choice may affect the completeness of the algorithm and the *qualities* of the synthesized plans. The design of P&S applications for HRC also entails design and verification issues [13]. Here, we focus on the approach feasibility and these issues are considered out of the scope of the paper.

B. *Action Planning Module*

The Action Planner is in charge of converting high-level actions from the task planner into actual robot movements. Its general structure is represented in Fig. 1. It is equipped with a motion planning module used to solve motion planning queries during the action planning. It encodes a set of actions such as *pick_from*, *place_in*, *assemble*, which are automatically parsed into a sequence of motion instances and basic operations (*e.g.*, *move_to*, *grasp*, *release*, *wait*). These basic operations represent the smallest level of granularity in the task decomposition hierarchy described in Fig. 2. For example, consider the case of a pick and place skill. The robot should perform the following operations: move to the picking station; approach the object; grasp it; move to the placing spot; approach the final pose; and release the object. These operations shall be executed in the aforementioned order. From the task planner point of view, it is pointless to subdivide the skill in single operations, given that their order is already known. On the contrary, gathering all the operation in a higher-level action reduces the complexity of the task planning problem because a large number of sub-tasks is removed. For this reason, all these operations are grouped in a single action (yellow nodes in Fig. 2) that corresponds to a planning instance for the Action Planner.

An action planning query finds the optimal plan to execute the sequence of basic operations (green nodes in Fig. 2. Although the order of the operations is already defined, the Action Planner has a certain level of autonomy in the way such operations are performed. Referring to the pick and place example, multiple objects of the same type might be eligible for the same pick and place task. The Action Planner runs a motion planning query from the robot current state to each eligible picking spot. In this way, it is possible to select the most suitable object. The selection criterion is the cost of the path planning query computed by means of the metric function used by the motion planning algorithm. Path length is the most common index, but indices based on the level of interference between the robot and the human's motion are also possible by using a human-occupancy costmap. Similarly, in the placing operation, if multiple placing spots are available, the Action Planner selects the most suitable with respect to the planning metrics. It is worth noticing that the path planning problems are solved online and therefore take into account the current state of the human co-worker and the environment. This means that the same motion planning query gives different solutions based on the current state of the workspace and the human behavior. Once an optimal plan to execute the action is found, the trajectories are sent to the robot controller, which is in charge of executing and monitoring the actual motion of the robot.

Other complex skills can be encoded in the motion planning module by following a similar approach. It is clear that reasoning at skill level is a great advantage for the task planner, which can now focus on high-added-value operations, while the motion planning module automatically manages complex but structured operations.

III. Case Study - Collaborative Mosaic Assembly

We analyze a case study in which an operator and a robot shall assemble the mosaic shown in Fig. 3. It reproduces the acronym *SW* (from the name of the project *ShareWork*) and is composed of 50 cubes arranged in 5 rows and 10 columns. Each slot of the mosaic is identified by its column letter and its row number (*e.g.*, A1, A2, etc.). Letter 'S' is made up of orange cubes; letter 'W' is made up of white cubes; the background is made up of blue cubes. The mosaic shall be assembled according to the following constraints: i) orange cubes moved only by the robot; ii) white cubes moved only by the operator; iii) blue cubes moved by both the robot and the operator; iv) Row 3 shall begin after the end of Row 1; v) Row 4 shall begin after the end of Rows 1 and 2; vi) Row 5 shall begin after the end of Rows 1, 2, and 3.

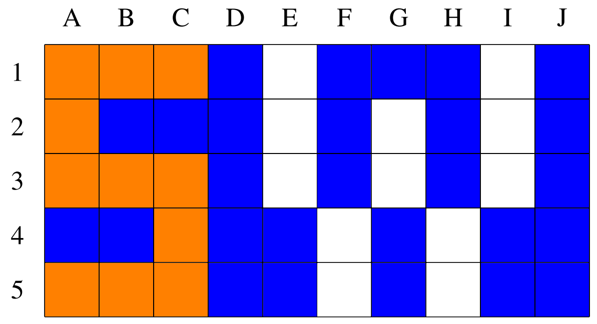


Fig. 3: A collaborative mosaic to be assembled. Orange tiles can be handled only by the operator, white tiles can be moved only by the robot, and blue tiles can be moved by both.

The following subsections describe the model considered in the proposed framework for the collaborative process and the ROS-based simulation environment developed for validating the proposed methodology.

A. *Dynamic Task Planning Module Implementation*

We define a timeline-based representation of the task planning problem for the case study. Fig. 2 shows an overview of the task planning model for the collaborative process modeled according to a Hierarchical Task Analysis [14]. A *production* *goal level* represents the high-level goals "triggering" the execution of a collaborative production process. A *production process level* represents the different collaborative processes that can achieve production goals. This level models the tasks to perform and the operational constraints to satisfy in order to successfully achieve a particular production goal. Finally, a *human/robot behavior layer* models the low-level tasks the two actors can simultaneously perform to actually realize the production tasks.

According to [15], the dynamics of these levels are describes by means of a number of *state variables*. A state variable $SV_G = \langle V_G, T_G, D_G, \gamma_G \rangle$ models the production goal level by representing production goals that can be performed into a production environment (e.g., the value $v_i = DoMosaic \in V_G$). A state variable $SV_M = \langle V_M, T_M, D_M, \gamma_M \rangle$ models the high-level tasks that should be performed to carry out a particular production goal of the goal level (i.e., values $v_i \in V_G$). Each value represents a (collaborative) high-level task to be executed in order to successfully carry out $v_j \in V_M = \{DoRow_1, ..., DoRow_5\}$ representing the tasks of assembling the rows of the mosaic). *Synchronization rules* link production goals (e.g., $v_i = DoMosaic \in V_G$) to the underlying production tasks ($v_j = DoRow_j \in V_M$) and specify a number of *temporal relations* between these values, representing desired operational constraints. Equation (1) shows an example of such a rule specifying a number of *precedence relations* between high-level high-level tasks $v_j = DoRow_J \in V_M$:

$$a_0[SV_G = DoMosaic] \rightarrow \quad \exists\, a_1[SV_M = DoRow_1], \; ..., \; a_5[SV_M = DoRow_5].\mathcal{C} \tag{1}$$

where $\mathcal{C} = DoRow_1 \prec DoRow_3 \wedge \ldots \wedge DoRow_4 \prec DoRow_5$.

Then, two additional state variables $SV_H = \langle V_H, T_H, D_H, \gamma_H \rangle$ and $SV_R = \langle V_R, T_R, D_R, \gamma_R \rangle$ characterize the behavior level and model respectively the low-level tasks the human and the robot can perform to actually carry out a collaborative process. In this specific case, the human and the robot can perform the same type of task $PickPlace_x$. A task $PickPlace_x$ consists in *picking* a tile x of the mosaic from a box (*i.e.*, *white_box*, *blue_box* and *orange_box*) and *place* the tile x in the correct position (*e.g.*, *A1*, *B2* or *G3*). Values $v_k = PickPlace_k \in V_H$ and $v_t = PickPlace_t \in V_R$ represent the $PickPlace_x$ low-level tasks the human and the robot can actually perform, according to specific production needs of the case study. The state variable of the robot behavior SV_R would not contain $PickPlace_x$ tasks concerning white boxes. Vice versa, the state variable of the human behavior SV_H would not contain $PickPlace_x$ tasks concerning orange boxes. For example the task $PickPlace_x$ with $x = A1$ can be performed only by the robot as $A1$ is an "orange tile", while the task $PickPlace_x$ with $x = I3$ can be performed only by the human as $I3$ is "white tile". Tasks concerning blue tiles instead (*e.g.*, $PickPlace_{D3}$) can be performed by both and thus are contained in both state variables ($PickPlace_{D3} \in V_R \cap V_H$).

Another set of synchronization rules links high-level tasks in SV_M to *low-level* tasks in SV_H and SV_R. These rules model possible allocations of low-level tasks (i.e., $PickPlace_x$ concerning blue tiles) and thus define different ways of performing high-level tasks. Equation (2) and Equation (3) show two examples of synchronization rules describing two alternative ways of performing high-level task $v_j = DoRow_1 \in V_M$. Specifically these rules show two alternative assignments of low-level tasks $PickPlace_x$ to the human and to the robot (Equation (2) assigns the low-level task $PickPlace_{J1}$ to the robot while Equation (3) assigns the same task to the human).

$$a_0[SV_M = DoRow_1] \rightarrow \quad \exists\, a_1[SV_R = PickPlace_{A1}], \; ..., a_5[SV_H = PickPlace_{E1}], \; ..., a_{10}[SV_R = PickPlace_{J1}].\mathcal{C}_{r1,a} \tag{2}$$

$$a_0[SV_M = DoRow_1] \rightarrow \quad \exists\, a_1[SV_R = PickPlace_{A1}], \; ..., a_5[SV_H = PickPlace_{E1}], \; ..., a_{10}[SV_H = PickPlace_{J1}].\mathcal{C}_{r1,b} \tag{3}$$

In general rules with the same *head* (*i.e.*, the left side of the rule) represent alternative ways of decomposing a task and are therefore treated as disjunctions during plan synthesis. In this case, they are used to model possible assignment of $PickPlace_x$ concerning blue tiles.

Controllability information characterizes task execution with respect to the uncontrollable dynamics of the environment. The controllability tagging function of a state variable SV_i specifies whether a value $v_j \in V_i$ is *controllable* or not. We distinguish among *controllable*, *partially-controllable* and *uncontrollable* values ($\gamma_i : V_i \rightarrow \{c, pc, u\}$). Controllable values are completely under the control of the system (it can decide both the start and the end of their execution). Partially-controllable values can only be started by the system while their end (*i.e.*, the actual duration of their execution) can only be observed during execution. Uncontrollable values can neither be started nor ended by the system and their actual behavior can only be observed.

The behavior of the human is modeled as *uncontrollable*. Values of $v_k \in V_H$ are tagged as uncontrollable ($\gamma_H(v_k) = u, \forall\, v_k \in V_H$). Robot behavior is modeled as *partially controllable*. Values $v_t \in V_R$ are thus tagged as *partially-controllable* ($\gamma_R(v_t) = pc, \forall\, v_t \in V_R$). Values of other state variables are modeled as *controllable*. Controllability information is used to analyze and evaluate *execution properties* of plans [16]. Therefore, PLATINUm synthesizes flexible plans that do not make assumptions on the actual duration of not controllable values/tasks (*i.e.*, *pseudo-controllable plans* [16]) assuring a good level of robustness and limiting the need for *task re-planning* at execution time [17].

B. Action Planning Module Implementation

The general structure of the Action Planner module described in Section II-B is implemented here for the case study. The general description of Fig. 2 is therefore tailored to the case study tasks and actions. Each pick-place operation is an action planning instance (yellow nodes in Fig. 2). A pick-place is then composed of several motion instances (green nodes in Fig. 2), namely: *move_to_X*, *approach_object*, *grasp_object*, *move_to_Y*, *approach_slot*, *release_object*. *X* and *Y* are symbolic labels that identify the color (*orange, white, blue*) of the picking cube and the coordinates of placing spots (*A1, A2, etc.*) respectively.

When an action from the task planning arrives (*e.g.*, $PickPlace_{J1}$), the motion planning module decodes the desired color and chooses the most suitable cube from the available ones. Then, it places the cube in the corresponding slot. To select the best action plan, a picking server and a packaging server are defined for each robot. The picking server manages multiple inbound stations, each station is a logical box containing objects of multiple types (*i.e.*, colors). When the picking server receives the request to pick an object of a certain type, it checks how many inbound stations contain the requested object type. These stations are the goals of the path planning queries. The picking server executes the movement to the closest inbound station and, simultaneously, it runs another path planning query by setting as goals the poses corresponding to the current placing slot. After the execution of the approaching movement to the selected object, the picking server executes the grasp action, such as closing the fingers or activating the suction cup. Finally, the picking server plans and executes the removal movement. The packaging server manages the filling of the mosaic. It initially plans a path to the approaching position of the specified slot. During the execution of the movement, the algorithm plan the movement to the placing slot. After that, the release action is executed, for example opening the gripper fingers or switching off the vacuum generator. It is worth noticing that both the picking server and the packaging server can manage multiple robots acting simultaneously in the cell.

C. *Software Implementation*

The mock-up scenario shown in Fig. 4 is a simulation environment built in ROS. It is composed of an Universal Robot UR5 mounted on an linear axis. The mosaic should be assembled on a worktable parallel to the linear axis. The blue pieces are on a worktable (accessible to both the robot and the operator), the orange ones behind the robot, and white ones on a table near the operator. A human mannequin is integrated in the simulator[1].

The Action Planner is implemented as a set of ROS action servers dedicated to pick or place tasks. Each action server implements a *MoveIt!* planner pipeline, allowing the use of all path planning algorithms available in ROS (*e.g.*, OMPL [18] and STOMP [19]). The Task Planner is implemented through PLATINUm (https://github.com/pstlab/PLATINUm) implementing a classical plan refinement solving procedure which consists in iterative refining the timelines of a plan by solving *flaws* concerning the completeness and the temporal validity of a plan (*i.e.*, P&S flaws). Also, PLATINUm integrates a pseudo-controllability check analyzing the duration of

[1]The modeled movements for the mannequin are: trunk (2 translations and 3 rotations), shoulders (3 rotations each), elbow (1 rotation each), wrist (3 rotations each).

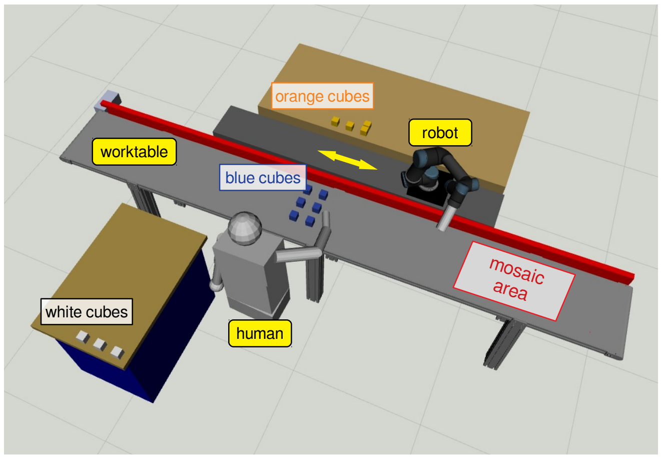


Fig. 4: A 7-degree-of-freedom robot and an operator collaborate to assemble a mosaic on the worktable; cubes of different colors are placed near the agents allowed to pick them.

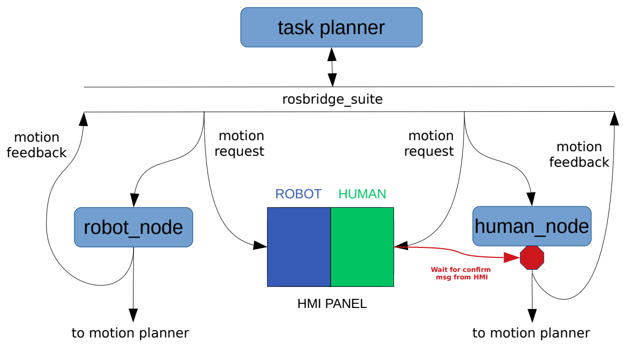


Fig. 5: ROS data-flow scheme between task planner, HMI, and motion planner nodes.

partially-controllable or uncontrollable values of the timelines. The solving procedure is completed when no flaws are found and a complete and valid partially-controllable set of timelines is synthesized. In this experiment, we ran PLATINUm using a hierarchy-based heuristics for flaw selection and a *HR-balancing search strategy* for search space expansion. The hierarchy-based heuristics is domain independent. It *infers* a hierarchy among the detected flaws of a timeline-based plan by using a topological sort algorithm on an acyclic graph-based representation of their dependencies (extracted from the synchronization rules of the model). According to this hierarchy, the algorithm randomly selects one of the flaws belonging to the *equivalent set* at the highest level of the hierarchy (*i.e.*, the “most independent flaws”) [20].

The *HR-balancing search strategy* is specifically designed for HRC planning problems. This search strategy analyzes the partial plans of the search space and selects the ones that better balance the work-load between human and robot.

The communication between the planners is schematized in Fig. 5. One ROS-node for each agent (`human_node` and `robot_node` in Fig. 5) converts the task Planner’s functional goals into motion instances of the motion planner algorithm.

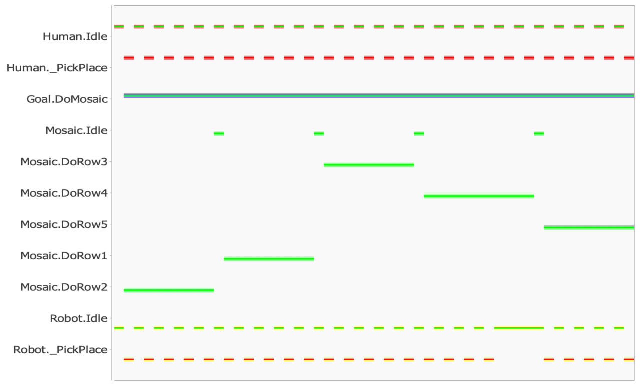


Fig. 6: Plan generated for the collaborative mosaic

One request topic and one feedback topic are defined for each agent node. The Task Planner publishes messages on the request topics to request the execution of a task and receives messages from the feedback topics to get the outcome (success/failure) of tasks. Human tasks are not controllable according to Section III-A: their duration is not ensured to be finite, as well as the time from request to beginning (such time might even be infinite if the operator refuses to perform the given task). The ROS package *rosbridge_suite* was used to set up the communication between PLATINUm and ROS.

The feedback from the user have been simulated through the deployment of a simple random-delay trigger that sends the feedback to the Task Planner after the human task is completed. To simulate the effects of safety functions on the robot motion (*i.e.*, robot slowdown/stop when the human enters the collaborative space), we have implemented a Speed and Separation Monitoring function, according to ISO/TS 15066 [21] using a simplified safety-zone approach that reduces the robot speed to: i) 50% of the nominal speed when the distance from the human to the worktable is less than 1 m; ii) 30% of the nominal speed when the distance is less than 0.5 m; iii) 0% of the nominal speed (*i.e.*, stop of the robot) when the human is working on the worktable.

D. Use Case Simulation

Fig. 6 shows a Gantt representation of a timeline-based plan synthesized of the collaborative mosaic. A timeline-based plan encapsulates an envelope of possible temporal behaviors as valid solutions to a planning problem. The Gantt shows a single particular solution we refer to as the *earliest start time solution*. The task planner takes about 20 seconds for the synthesis of this plan which represents the time spent to generate a task plan of the overall collaborative process to build the mosaic. The collaborative mosaic indeed requires the execution of a total number of 50 $PickPlace$ tasks. If we consider an average duration of 45 seconds for task, the total execution time of the overall process would be 38 minutes in the worst case (all tasks assigned to the human or to the robot) and 20 minutes in the best case (perfect balancing between human and operator, without physical disturbance).

As can be seen in the Gantt, plans synthesized by the task planner uniformly distribute the tasks between the human and the robot. The resulting plans indeed have an average *makespan* of 52 time units with 26 $PickPlace$ tasks assigned to the human and 24 $PickPlace$ tasks assigned to the robot. Such plans show the efficacy of the task planner in synthesizing suitable and effective collaborations with reasoning time that complies with production latency. When the plan is ready, the communication between the task planner and the motion planning framework starts. The task planner sends the robot current task to `robot_node` and the human current task to `human_node`, then it waits for the feedback message of each task. Feedback messages communicate the outcome of the task. If the task was not achieved, is refused by the operator, or exceeds its maximum duration, task planner detects that the plan is not valid anymore and re-plans the remainder of the process; otherwise, the plan execution continues unaltered. Several phases of the case-study simulation are shown in Fig. 7. The progress of the mosaic assembly is shown from 0% (Fig. 7a) to 100% (Fig. 7f). For example, in Fig. 7a, the execution has just started; while the human is placing a blue cube, the robot is still because of the safety functions that reduce the robot speed based on the proximity to the human. In Fig. 7e, the human waits for the robot to finish its task, while the robot is picking the last cube of Row 4. Fig. 7f shows the final mosaic.

IV. Conclusions and Future Works

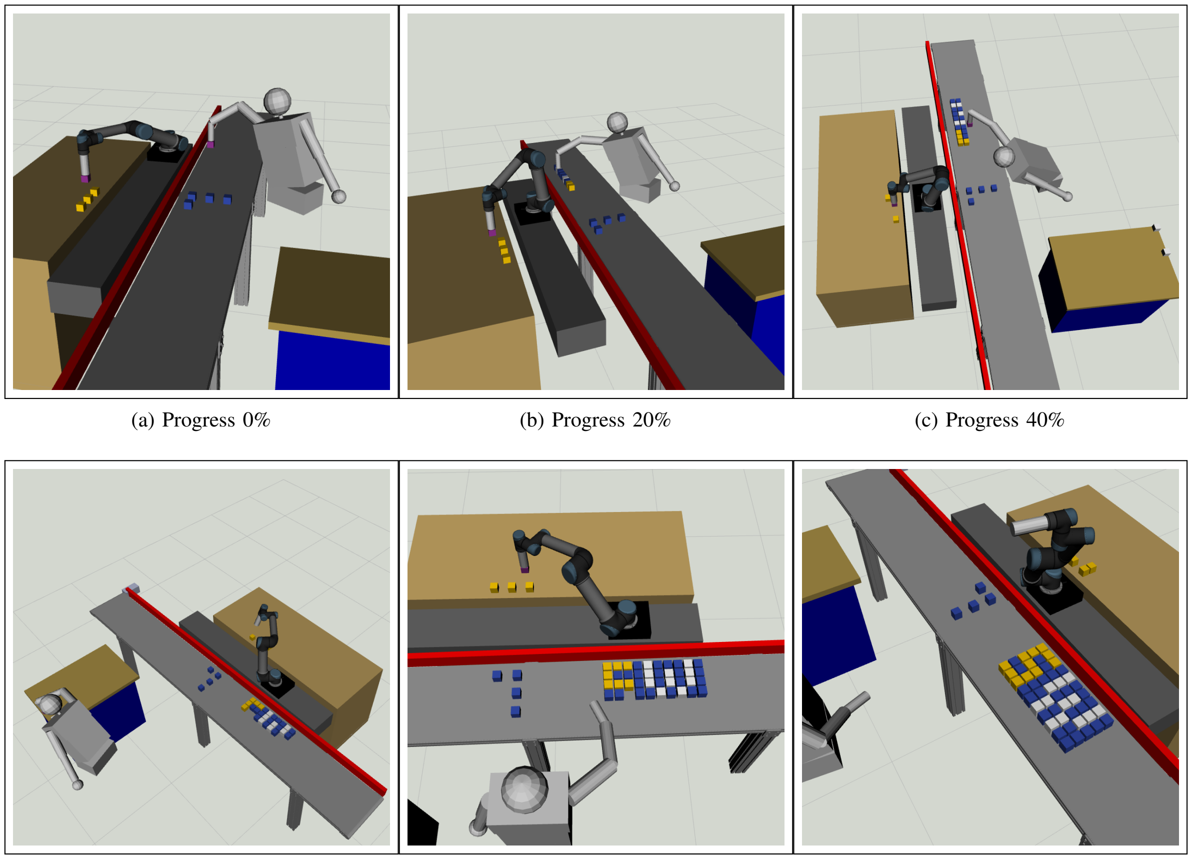
The paper proposes a control-based approach based on two layers (task and action planning) where each layer reasons at a different level of abstraction. This constitutes a hierarchical framework implementing a control loop that leverages feedback to monitor the execution and dynamically (re)optimize the process plan. Moreover, operators can accept/discard commands (via HMI) and give feedback about the outcome of his/her tasks. The method is applied to an industrial case study in which a robot and a human worker cooperate to assemble a mosaic. The main current limitation is related to the need of an *a-priori* estimation of the sub-tasks duration. This estimation is often inaccurate (e.g., it is hard to account for robot safety holds/stops). Despite the timeline-based modelling deals with temporal variability, larger error in the time estimation weakens the robustness of the plan, and the optimality is hard to be guaranteed. Future works will focus on including a learning phase to produce a realistic estimation of the sub-task durations. Moreover, experiments in a real setting are necessary to assess user dependability, investigate scalability and extensibility of the approach and to compare its performance against industrial best practices.

Acknowledgment

This work is partially supported by ShareWork project (H2020, European Commission – G.A. 820807). Alessandro Umbrico is also partially supported by ROXANNE project (H2020, ROSIN – G.A. 732287).

References

[1] L. Monostori, B. Kádár, T. Bauernhansl, S. Kondoh, S. Kumara, G. Reinhart, O. Sauer, G. Schuh, W. Sihn, and K. Ueda, “Cyber-physical systems in manufacturing,” *CIRP Annals*, vol. 65, no. 2, pp. 621 – 641, 2016.



(a) Progress 0% (b) Progress 20% (c) Progress 40%

(d) Progress 60% (e) Progress 80% (f) Progress 100%

Fig. 7: Assembly of the cooperative mosaic during the simulation.

[2] S. Pellegrinelli, A. Orlandini, N. Pedrocchi, A. Umbrico, and T. Tolio, "Motion planning and scheduling for human and industrial-robot collaboration," *CIRP Annals*, vol. 66, no. 1, pp. 1 – 4, 2017.

[3] S. Pellegrinelli, N. Pedrocchi, L. M. Tosatti, A. Fischer, and T. Tolio, "Multi-robot spot-welding cells for car-body assembly: Design and motion planning," *Robotics and Computer-Integrated Manufacturing*, vol. 44, pp. 97 – 116, 2017.

[4] S. Srivastava, E. Fang, L. Riano, R. Chitnis, S. Russell, and P. Abbeel, "Combined task and motion planning through an extensible planner-independent interface layer," in *2014 IEEE International Conference on Robotics and Automation*, 2014, pp. 639–646.

[5] J. Wolfe, B. Marthi, and S. J. Russell, "Combined task and motion planning for mobile manipulation," in *Proceedings of the 20th International Conference on Automated Planning and Scheduling, ICAPS 2010*, 2010, pp. 254–258.

[6] L. de Silva, R. Lallement, and R. Alami, "The HATP hierarchical planner: Formalisation and an initial study of its usability and practicality," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 6465–6472.

[7] T. Lozano-Pérez and L. P. Kaelbling, "A constraint-based method for solving sequential manipulation planning problems," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014, pp. 3684–3691.

[8] M. Toussaint, "Logic-geometric programming: An optimization-based approach to combined task and motion planning," 2015.

[9] C. Zhang and J. A. Shah, "Co-optimizing task and motion planning," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 2016-Novem. IEEE, oct 2016, pp. 4750–4756.

[10] N. Muscettola, "HSTS: Integrating Planning and Scheduling," in *Intelligent Scheduling*, Zweben, M. and Fox, M.S., Ed. Morgan Kauffmann, 1994.

[11] A. Cesta, G. Cortellessa, S. Fratini, and A. Oddi, "MrSPOCK: Steps in Developing an End-to-End Space Application," *Computational Intelligence*, vol. 27, no. 1, 2011.

[12] A. Umbrico, A. Cesta, M. Cialdea Mayer, and A. Orlandini, "PLATINUm: A New Framework for Planning and Acting," *Lecture Notes in Computer Science*, pp. 498–512, 2017.

[13] A. Orlandini, M. Cialdea Mayer, A. Umbrico, and A. Cesta, "Design of timeline-based planning systems for safe human-robot collaboration," in *Knowledge Engineering Tools and Techniques for AI Planning*, M. Vallati and D. Kitchin, Eds. Springer, 2020, pp. 231–248.

[14] J. A. Marvel, J. Falco, and I. Marstio, "Characterizing task-based human-robot collaboration safety in manufacturing," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 45, no. 2, 2015.

[15] M. Cialdea Mayer, A. Orlandini, and A. Umbrico, "Planning and execution with flexible timelines: a formal account," *Acta Informatica*, vol. 53, no. 6-8, pp. 649–680, 2016.

[16] P. H. Morris, N. Muscettola, and T. Vidal, "Dynamic Control of Plans With Temporal Uncertainty," in *International Joint Conference on Artificial Intelligence (IJCAI)*, 2001, pp. 494–502.

[17] A. Umbrico, A. Cesta, M. Cialdea Mayer, and A. Orlandini, "Evaluating robustness of an acting framework over temporally uncertain domains," in *AI*IA 2019 – Advances in Artificial Intelligence*, M. Alviano, G. Greco, and F. Scarcello, Eds. Springer, 2019, pp. 250–263.

[18] I. A. Şucan, M. Moll, and L. E. Kavraki, "The Open Motion Planning Library," *IEEE Robotics & Automation Magazine*, vol. 19, no. 4, pp. 72–82, December 2012, http://ompl.kavrakilab.org.

[19] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "STOMP: Stochastic trajectory optimization for motion planning," in *Proc. IEEE ICRA*, Shanghai (China), 2011, pp. 4569–4574.

[20] A. Umbrico, A. Orlandini, and M. Cialdea Mayer, "Enriching a temporal planner with resources and a hierarchy-based heuristic," in *AI*IA 2015, Advances in Artificial Intelligence*. Springer International Publishing, 2015, pp. 410–423.

[21] "ISO/TS 15066:2016 Robots and robotic devices – Collaborative robots," International Organization for Standardization, Geneva, CH, Standard, 2016.