A Knowledge-Driven Shared Autonomy Human-Robot Interface for Tablet Computers

Peter Birkenkampf, Daniel Leidner, and Christoph Borst

***Abstract*— Autonomous and teleoperated robots have been proven to be capable of solving complex manipulation tasks. However, autonomous robots can only be deployed in predefined domains and teleoperation requires full attention of a human operator. Therefore, combining autonomous capabilities of the robot with teleoperation is desirable, yet balancing the work load between robot and operator for intuitive human-robot interfaces is still an open issue. In this paper, we present a knowledge-driven tablet computer application for commanding a robot on a high level of abstraction. The application guides an operators decisions based on the actual world state of the robot and enables the operator to command object-centered actions, which are autonomously interpreted symbolically and geometrically by the robot. We evaluate our approach using the humanoid robot Rollin' Justin for an elaborate manipulation experiment and an user study. We show that our approach efficiently balances the work load between robot and operator and provides an intuitive interface for human-robot interaction.**

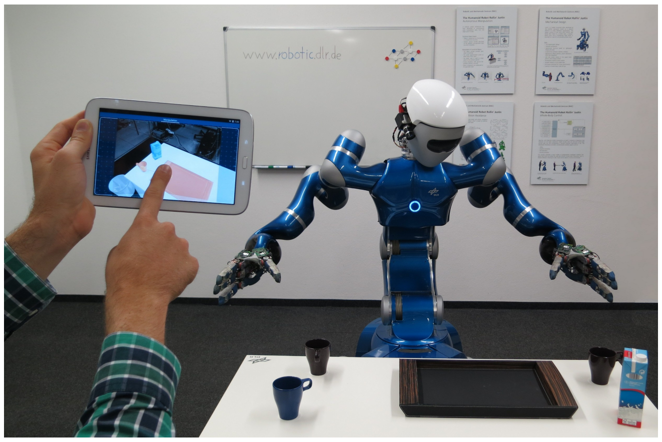


Fig. 1. The humanoid robot Rollin' Justin of DLR operated via a shared autonomy human-robot interface implemented as a tablet computer application. Objects with potential actions are highlighted.

I. Introduction

In the near future, service robots will be introduced to varying scenarios including elderly care, disaster response, and space applications. With multiple manipulators and sophisticated sensor systems, the capabilities of such robots will exceed the capabilities of today's service robots, such as vacuum robots, by far. As these new capabilities introduce higher complexity, commanding robots gets more difficult and therefore raises the need for autonomously acting robots. However, solving all kind of service tasks fully autonomous in unstructured and changing environments without human intervention is a challenging task. A possible solution to circumvent this issue is *shared autonomy*, where an operator remote controls the robot on a high level of abstraction.

In order to command complex manipulation tasks to a service robot, it is crucial to provide an intuitive high level *human-robot interface (HRI)*. This implies not just the way of commanding a certain task *to the robot*, but also the opposite direction of presenting task relevant information *to the operator*. The main challenge hereby lies in guiding the operators attention intuitively by visualizing the required information, while simultaneously hiding irrelevant information to prevent the operator from distractions. Furthermore, it is important to narrow the operators possible decisions w. r. t. the current state of the environment and task since the number of possible manipulation actions increases drastically with the number of available objects. An appropriate knowledge representation along with extensive reasoning mechanisms is therefore mandatory.

The goal of this research is to provide an intuitive HRI enabling an untrained operator to effectively command a service robot with advanced capabilities. We believe it is helpful for the untrained operator to use a familiar device for commanding the robot. We exclude special designed HRI hardware as presented in e. g. [1], [2] or [3] and focus on affordable mobile devices of everydays life. Mast *et al.* [4] present a concept providing different sized devices to control a robot on different levels of abstraction, including *smartphones*, *tablet computers* and *desktop computers*. Rouanet *et al.* [5] confirm the advantages of using a touchscreen device for commanding robots as it provides intuitive feedback by showing video streamed from the robot on the screen. Because of these investigations, the high mobility, and good availability, we utilize a tablet computer for our work.

The touchscreen of the tablet computer is used to display feedback from the robot to the operator providing the information used for gaining common ground with the robot and situational awareness in the remote environment [6]. An often utilized approach uses the major part of the screen for showing the video stream of a camera mounted on the robot [7][8] optionally augmented with additional sensor [9] or status information [10][11][12]. In some HRI the perspective is changed to show the robot in its environment [13][14] or from a top-down angle [15][16]. Investigations of the influence and user preferences of these different perspectives showed that an egocentric perspectives supports identifica-

All authors are affiliated with the Institute of Robotics and Mechatronics, German Aerospace Center (DLR), Wessling, Germany. Contact: peter.birkenkampf@dlr.de

tion tasks and manipulation tasks whereas an exocentric perspective has advantages in navigation tasks [14] [9].

In recent years a lot of touchscreen based - mobile and stationary - HRIs were proposed. Some of those control basic movement functions of the commanded robot by the use of virtual buttons [7], joysticks [17], sketched gestures [8], or use virtual handles that are shown after the user touches the robot on the video stream of an external camera [13].

Other approaches command high level capabilities of the robot. Sakamoto *et al.* [16] do this by directly mapping gestures to robot actions. Correa *et al.* [12] reduce the gesture set by assigning different robot actions to the same gesture depending on its location in the video streamed from the robot. Paravati *et al.* [18] allows the operator to self-assign gestures to actions. However, even if the set of gestures is small, the operator has to memorize the gestures and their mapping to actions. Furthermore, gestures can be easily misused due to their ambiguous nature. An approach around this issue uses a virtual toolbox providing parameterizable robot capabilities as presented by Liu *et al.* [15].

The HRIs described so far focus either on teleoperation or autonomous applications. However, real world environments often demand a combination of both [6]. Muszynski *et al.* [11] proposed a tablet computer based HRI allowing the operator to adjust the autonomy of the robot by switching between commanding of basic movements, semiautonomous capabilities or high level tasks. Ciocarlie *et al.* [19] and Chen *et al.* [10] use a similar approach with robot-capability oriented components such as gazing, moving and manipulating for semi-autonomous pick and place tasks.

The described HRIs follow an action-centered approach for commanding the robot. Expert knowledge about the capabilities of the robot is required in order to execute a specific task increasing the cognitive load of the operator. An alternative approach avoiding this problem is the use of an object-centered focus as applied by Suzuki *et al.* [20]. Their HRI restricts the movement of a surveillance robot based on the relationship between the robot and a chosen object allowing simplified inspection. Fung *et al.* [21] teach sequential tasks to a robot by augmenting photographies of task related objects with icons for possible actions. The selected actions are manually parametrized by the operator and afterwards checked for semantic validity. Although the proposed system is utilized for teaching tasks offline, using a similar approach for an online HRI could efficiently lower the cognitive load of the operator because after selecting the objects to be manipulated, a choice from only a small set of action options has to be made. This can be supported by performing semantic checks in advance to provide only valid action possibilities to the operator.

Our contribution presented in this paper is a novel knowledge-driven approach for a shared autonomy HRI. We are able to reduce the cognitive load for the operator to a minimum by choosing an object-centered high level of abstraction to command a robot. The symbolic and geometric reasoning is thereby shifted from the operator to the robot allowing the operator to focus on the task rather then on robot-specific capabilities. Evaluating the possibility of actions in advance allows us to simplify the operator frontend by - in contrast to known approaches - providing only possible actions for commanding the robot. We apply minimalistic *graphical user interface (GUI)* design principles to guide the operators decisions w. r. t. the actual world state as it is perceived by the robot. By directly displaying video from the robot, we obtain an egocentric perspective which we augment with information about the internal state of the robot. We provide a mechanism for the operator to directly interact with the displayed information by applying a point-and-click paradigm. This way, we demonstrate a novel approach for commanding manipulation tasks based on intuitive selection of the objects to be manipulated. We present a prototypical tablet computer application based on the proposed approach to operate our humanoid robot Rollin' Justin in an everyday manipulation scenario. The usability of the application is evaluated in an user study based on three experiments with multiple users.

The paper is organized as follows: We begin with a presentation of the methods we used to realize our approach in Sec. II. A prototypical tablet computer application implementing these methods is presented in Sec. III. Afterwards the capabilities of the implementation of the proposed system are shown in Sec. IV. We conclude the paper with a brief summary of our results in Sec. V.

II. Methods

While remote commanding a robot, the focus for the operator should be on the task to be solved, rather than on the robot to be operated. We believe this is supported by a HRI replicating the real world experience, as if someone would solve the given task on its own. In order to achieve this feeling, the robot has to have the same information about the environment and the contained objects as the operator would have in its place. We address this issue by facilitating an object-centered knowledge base which provides symbolic and geometric object information to the robot. With this information it is possible to generate actual feasible action possibilities from the current world state as described in Sec. II-A. To avoid distractions from too much information while still guiding the operators decisions, an intuitive *user interface (UI)* design along with a straightforward command concept is developed according to Sec. II-B.

A. Providing Information

In our previous work we have shown that it is vital to arrange knowledge within an object-centered context to solve everyday manipulation tasks [22]. Following this concept, objects are organized in a hierarchical structure according to their functionality. The preliminary software architecture of the knowledge system is summarized as follows:

An *object storage* provides prior knowledge for all known objects by the robot. The objects are hierarchically arranged in the object-oriented paradigm and categorized by functionality. Objects of the same class share the same process models for handling and can therefore be manipulated in the same way while considering their specific properties such as size and shape. The *world representation* holds the current state of the environment of the robot. Objects as described in the object storage are instantiated here with specific symbolic and geometric properties.

The task reasoning as it is utilized in this paper is performed in the context of the objects related to the task. The object functionality is therefore stored in so-called *action templates* which define distinct manipulation instructions. Those action templates consist of two segments which are evaluated in a two step *hybrid reasoning* approach. First, the symbolic headers defined in the *Planning Domain Definition Language (PDDL)* [23] are parsed in order to construct the symbolic domain and to solve a given task symbolically. The resulting symbolic transition is afterwards evaluated in the second step. Therefore the geometric body of the action templates is evaluated to ground the symbolic actions into robot-specific actions by the use of modular geometric simulations such as navigation, motion planning or dynamics simulations. Geometric backtracking in case of unsuccessful simulation is inherent in this step. Please refer to [22] for a more detailed description. This approach has been successfully evaluated to solve everyday manipulation tasks [22], mobile manipulation tasks [24], and force-sensitive whole-body manipulation [25].

However, to command versatile manipulation tasks to a service robot in an effective manner, an intuitive UI is required. We approach this issue by translating the internal world state of the robot into an object-centered UI. The entry point for the operator is thereby not task-specific but the selection of a single object or a set of multiple objects allowing our approach to scale for arbitrary tasks. A list of all selected objects is passed to the the *ListActions* function described in algorithm II.1 that identifies all available action permutations for the selected objects.

Algorithm II.1: ListActions(*objectList*)

```
$allActionsList \leftarrow$ List()
$combiningActionsList \leftarrow$ List()

for each $object \in objectList$
  do $allActionsList$.Append(GetAllActions($object$))

for each $action \in allActionsList$
  do
    $valid \leftarrow True$
    for each $object \in objectList$
      do
        $res \leftarrow$ SubstituteParameter($action, object$)
        if $res = False$
          then
            $valid \leftarrow False$
            break
    if $valid = True$
      then $combiningActionsList$.Append($action$)
return ($combiningActionsList$)
```

First all available action templates are collected, by iteratively calling the *GetAllActions* function for all objects of interest. This includes all actions for the particular object class, and all parent classes. In a second loop all object classes respectively object parent classes are substituted at their first appearance in the *action* (see *SubstituteParameter*). As a result, a *combiningActionsList* containing common PDDL parameter sets for all valid combinations is obtained. The remaining parameters in the actions are afterwards resolved according to algorithm II.2

Algorithm II.2: ResolveAction(*action*)

```
$resolved \leftarrow$ List()
for each $resolvedAction \in$ RecurseParameters($action$)
  do
    if $resolvedAction \in resolved$ :
      then continue
    $precons \leftarrow$ GetPrecons($resolvedAction$)
    $effects \leftarrow$ GetEffects($resolvedAction$)
    if $effects \subset worldState$ :
      then $tier \leftarrow 0$
    elif $precons \subset worldState$
      then $tier \leftarrow 1$
    else
      $plan \leftarrow$ SymbolicPlan($effects$)
      if Length($plan$) $\leq 1$
        then $tier \leftarrow 0$
      else $tier \leftarrow 2$
    $resolved$.Append([$resolvedAction, tier$])
return ($resolved$)
```

The *ResolveAction* function is executed for each action given by algorithm II.1. To cover all action possibilities, the world state is recursively parsed for all available parameter combinations by the *RecurseParameters* function. All not yet substituted parameters are therefore iterated and listed according to the current world state. The resulting list of action possibilities is divided in three tiers w. r. t. the preconditions and the effects of the action:

Tier 0) Idle Actions: Idle actions have no influence to the environment. They are of little relevance since they create effects equal to a subset of the current world state.

Tier 1) Single-Step Actions: To solve tier 1 actions, the robot has to execute one single action. The required symbolic preconditions are subset of the current world state. Since they are of limited complexity they are less error-prone and thus of high interest for the operator.

Tier 2) Multi-Step Actions: The preconditions of tier 2 actions are not available in the current world state. In order to perform the action, the preconditions have to be reached by executing at least one additional action. However, since tier 2 actions are not guaranteed to be symbolically achievable or possibly replaceable by a single tier 1 action, it is mandatory to verify their symbolic feasability (see *SymbolicPlan*). Tier 2 actions are valuable for the operator since they provide an overview of possible long-term tasks.

The described method generates a set of currently possible actions which are divided in three tiers. For clarity, we are going to explain our approach with an example. The domain for this example is a table-top scenario. It is constructed of three mugs (*mug1*, *mug2*, and *mug3*) of object class *_mug*, a *tray* which inherits from class *_surface*, and a table which also inherits from the *_surface* class. The robot provides a *left_arm* and a *right_arm* manipulator of class *_manip*. The geometric topology is outlined in Fig. 2. The listing below illustrates the outcome of algorithm II.1 for this scenario. The list of *actions* for the demanded *mug1* object consists

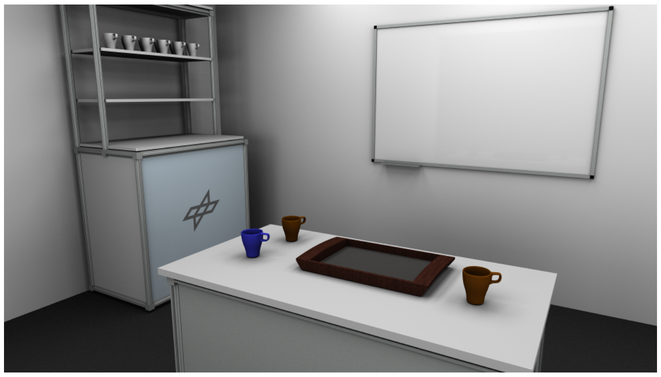


Fig. 2. A table-top scenario including three mugs and a tray.

of all available actions for the single object instance. The parameters *?s - _surface*, *?c2 - _mug*, and *?m - _manip* are not yet resolved.

```
>>> actions = ListActions('mug1')

actions = [
  ['pick',    'mug1', '?s - _surface',   '?m - _manip'],
  ['place',   'mug1', '?s - _surface',   '?m - _manip'],
  ['stack',   'mug1', '?c2 - _mug',      '?m - _manip'],
  ['unstack', 'mug1', '?c2 - _mug',      '?m - _manip'],
  ['dispose', 'mug1', '?t - _trash_can', '?m - _manip']]
```

The remaining open parameters have yet to be resolved according to all other objects in the world state as illustrated in algorithm II.2. Given the table-top scenario, the action possibilities for *mug1* correspond to the listing below. The *pick from table* action is of tier 1 since the preconditions match the current world state. The *place* and *stack* actions are of tier 2. In order to execute these actions, *mug1* has to be picked first by the robot. The *unstack* actions are superfluous since they replicate the symbolic effects of the pick action. The *dispose* action cannot be resolved because no object of the class *_trash_can* exists in the scenario.

```
>>> resolved = []
>>> for a in actions:
>>>   resolved.extend( ResolveAction(a) )

resolved = [
  (['pick',    'mug1', 'table', 'right_arm'], 1),
  (['pick',    'mug1', 'table', 'left_arm' ], 1),
  (['pick',    'mug1', 'tray',  'right_arm'], 0),
  (['pick',    'mug1', 'tray',  'left_arm' ], 0),
  (['place',   'mug1', 'tray',  'right_arm'], 2),
  (['place',   'mug1', 'tray',  'left_arm' ], 2),
  (['stack',   'mug1', 'mug2',  'right_arm'], 2),
  (['stack',   'mug1', 'mug2',  'left_arm' ], 2),
  (['stack',   'mug1', 'mug3',  'right_arm'], 2),
  (['stack',   'mug1', 'mug3',  'left_arm' ], 2),
  (['unstack', 'mug1', 'mug2',  'right_arm'], 0),
  (['unstack', 'mug1', 'mug2',  'left_arm' ], 0),
  (['unstack', 'mug1', 'mug3',  'right_arm'], 0),
  (['unstack', 'mug1', 'mug3',  'left_arm' ], 0)]
```

Even though not all of the actions are executable by the robot within only one single action, they are all valuable to the operator to get an overview of the actions the robot can possibly execute. It is not important to the operator that neither symbolic connections or geometric parameters are listed within the action list, since the hybrid reasoning is autonomously executed by the robot as described earlier. However, the operator might still get overwhelmed by too many available actions since everyday environments offer a large number of objects that are subject to manipulation tasks. In the following Sec. II-B we are going to emphasize how this can be avoided by integrating the presented approach into an intuitive UI design.

B. Visualizing Information

The method proposed above generates large action sets for complex world states. Presenting all of these actions to the operator is inapplicable as it results in high cognitive load. Therefore we develop an UI concept based on a point-and-click paradigm and object-centered information reduction.

In a nutshell, the point-and-click paradigm defines that pointing to a specific location indicates an users interest and clicking executes some kind of action. This approach is commonly used in window-based operating systems and adventure video games. The latter often facilitates control over a virtual agent whose movements are directed, but not completely specified by the user through clicking on objects in the virtual world. By this, high level targets are specified that are reasonable for the clicked object in the current context. The corresponding actions to achieve these targets are autonomously scheduled and executed by the agent.

We adapt the point-and-click paradigm for our needs to command service robots through a minimalistic and intuitive UI. Therefore the operator selects the objects to be manipulated in the GUI of the HRI. The robot then autonomously determines possible actions combining all selected objects as described in Sec. II-A and provides the resulting action set to the operator. After choosing an action of the set, the execution is done autonomously by the robot.

As a basis for command decisions, we establish common ground between robot and operator by streaming real time video of the head mounted camera of the robot to the operator. This provides a robot-centered perspective on the (remote) world supporting situational awareness of the operator. We believe this perspective supports a focus on the task to be solved as it replicates the real world experience the

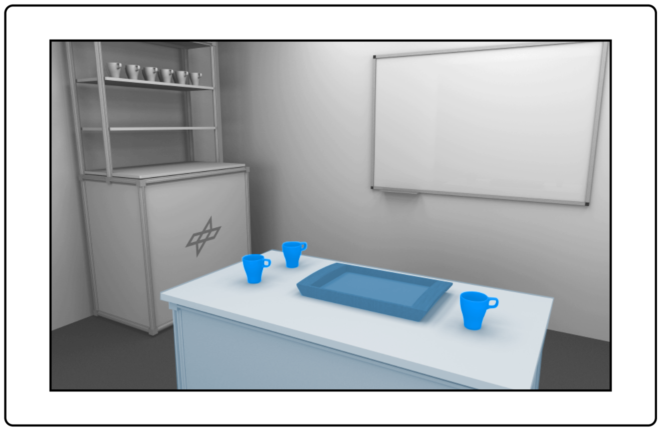


Fig. 3. Conceptual HRI GUI showing the video stream of the robot augmented with CAD models of the world state objects: table, tray, and three mugs. A high saturation of the highlight color corresponds with a high number of possible actions for the respective object.

operator would perceive in place of the robot. Additionally we augment the video stream with CAD models of all objects that are currently part of the internal world state of the robot. Fig. 3 shows the concept of the resulting GUI.

The CAD data of the world state objects of the robot is used to guide the focus of the operator by emphasizing regions in the video stream. Therefore each object is overlayed with its semitransparent 3D model. The saturation of the overlay increases with the number of possible actions of the object. Fig. 3 illustrates this approach: The world state objects (table, tray, and mugs) are emphasized by blue color. The mugs provide the most possible actions (*pick*, *place*, *stack*) and are therefore shown with a high saturation. The table provides no possible action thus it is draw with a low saturation. Areas of the video not showing detected objects are deemphasized using greyscale colors but are nevertheless displayed to help the operator gaining situation awareness in the remote environment. This adaptation of visual feedback efficiently reduces unimportant information resulting in lower cognitive load and better guidance towards task achievement for the operator. For better overview in cluttered scenes, the highlighted objects may be outlined.

The described highlighting of objects helps the operator to identify possible objects for manipulation. Selecting objects-of-interest is the entry point for commanding the robot. A set of possible actions w. r. t. the selected objects is determined according to Alg. II.1 and resolved into single-step actions and multi-step actions according to Alg. II.2. The selection of an action from the set is simplified by splitting the action descriptions into layers as illustrated in Fig. 4.

As a result a step-by-step parametrization can be applied which results in fewer choices per layer and thus reduces the cognitive load of the operator when selecting an action.

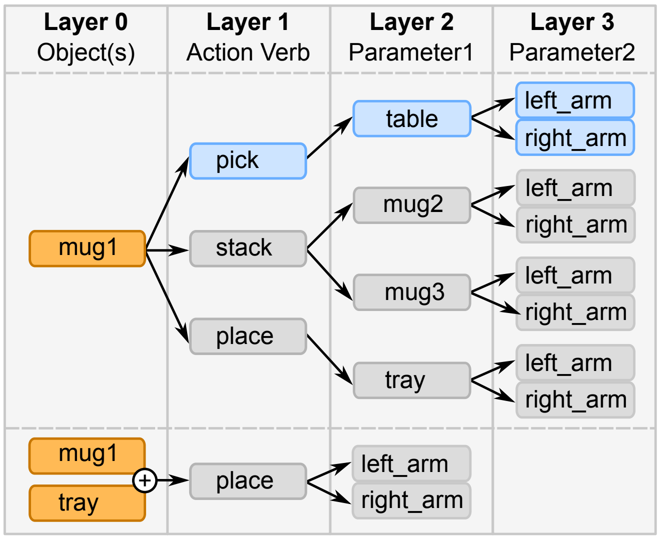


Fig. 4. Decomposition of actions into layers. The upper part illustrates a single selection, while two objects are selected in the lower part. The orange box indicates the selected objects. Blue boxes are available *tier 1* actions for the selected objects and are executable within one step. Gray boxes indicate *tier 2* actions which require multiple steps.

The operator can furthermore actively influence the size of the generated action set by the number of selected objects. The more objects are chosen by the operator the less actions combine all of these objects as shown at the bottom of Fig. 4.

The objects used to generate the action set are collected in layer 0 and stripped from the PDDL action descriptions to minimize the depth of the hierarchy. At the first layer, the operator has to choose between possible action verbs - e. g. *pick*. These are refined by their respective resolved PDDL parameters at the following layers - e. g. *table, left_arm*. This layering allows the stepwise specification of an action giving the operator better focus on the aimed target as the range of possible selections is bounded to a single layer at a time. Using colors, we differentiate between single-step actions (blue) and multi-step actions (grey) in the hierarchy to help the operator choose appropriate actions.

A simple selection could be performed by classical input methods, such as one-dimensional list boxes and drop down menus as used in many HRIs, e.g. [11][21]. However, it would get more difficult for an operator to pick a desired action from the list represented by such an input element if the set of possible actions gets large due to enhanced capabilities of the robot and complex world states. Therefore we utilize a hierarchical ring menu [26] as shown in Fig. 5 where every ring represents possible selections from the respective layer of the hierarchy.

By dynamically showing children rings as their respective predecessor is selected in the parent ring, the displayed information is reduced to choices the operator is currently interested in. For a better understanding of the selection, the fully specified PDDL definition of the currently selected action is shown at the top of the screen. Emphasizing single-step actions in the ring menu makes it easy for the operator to identify simple - possibly more often needed - actions. This is particularly useful when the operator plans to command a simple action, e. g. *pick*, where only one object is selected so a large number of possible actions is generated due to that non-restrictive operator input as shown in Fig. 4.

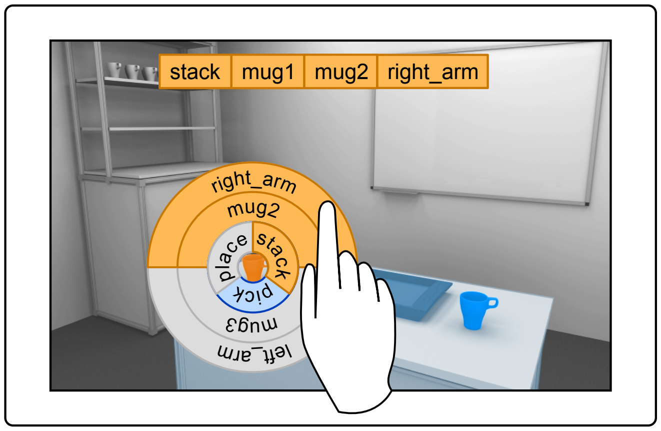


Fig. 5. Use of a ring menu to reduce the complexity of action selection by exploiting the hierarchical structure of actions. The colors in the ring menu correspond to Fig. 4. Currently selected regions of each ring as well as the selected object are highlighted with orange color.

III. Prototype Implementation

A prototype of the proposed HRI is implemented for tablet computers. The complexity of the system is distributed between the mobile device and the robot as shown in Fig. 6. Modules for reasoning, as described in Sec. II-A and hardware access are provided by the robot. The mobile device implements the visualization and the handling of operator input, as described in Sec. II-B.

The UI provided by the mobile device requires only few continuous input from the robot: compressed video stream, selected telemetry data, and current world states. CAD data for the visualization of world state objects is downloaded from the robot and cached at the mobile device on demand. The commanding of the robot is done by transmitting PDDL action descriptions from the mobile device to the robot. The application requires little bandwidth due to the transmission of few selected and compressed data. Furthermore, higher latency of the overall system is tolerated by shared autonomy compared to a teleoperation approach.

Using the mobile device, the operator interacts with the objects in the world state of the robot. The respective CAD models are rendered on top of the live video stream. A set of possible actions w. r. t. the selected objects is generated from the symbolic world state and presented to the operator using a ring menu. Fig. 7 shows a screenshot of the implemented GUI. After the operator selects an action from the set, the respective action description is sent to the hybrid planner of the robot. As described in Sec. II-A the task is first resolved symbolically and afterwards geometrically. The robot relies thereby on the same information from the action templates that is used to visualize the manipulation possibilities to the operator. Finally an execution confirmation is requested from the operator. When the execution is confirmed, the planning result is executed autonomously. For additional safety, a command immediately stopping the action execution can be sent by the operator at every time.

The described HRI strongly depends on the operator seeing the objects, he wants to manipulate on the screen of the mobile device. In cases when the camera is not facing the desired objects, different methods for changing the visual gaze of the robot are implemented for the prototype: Gaze following drag gestures on the touchscreen of the mobile device, gaze mimicking movements captured by inertial sensors of the mobile device and gaze autonomously facing objects by using a *look_at* capability of the robot.

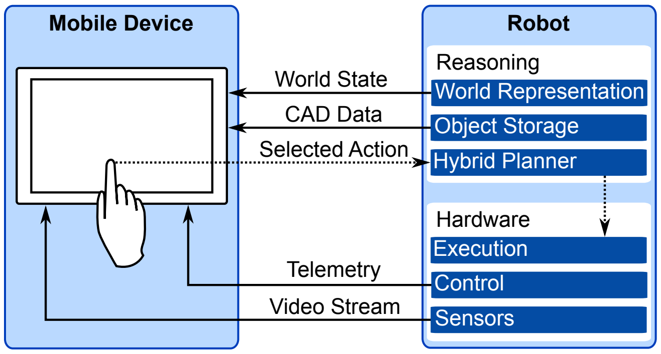
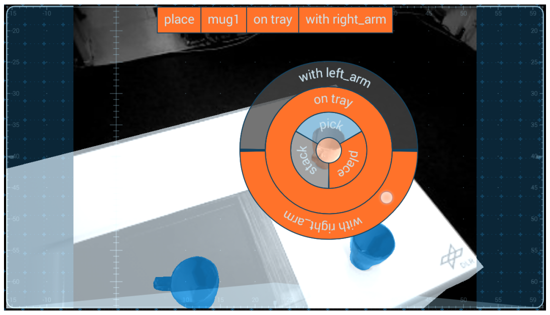
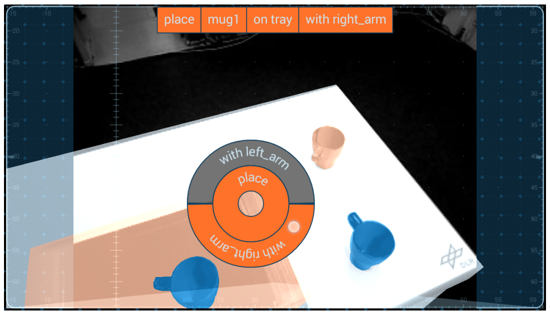


Fig. 6. Overview of software components of the prototypical HRI implementing the proposed system. Continuous arrows show constant data updates. Dotted arrows show the basic data flow of an operator-initiated action command.



(a)



(b)

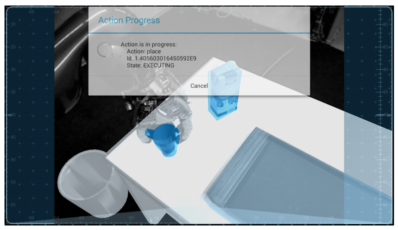
Fig. 7. Screenshots of the prototypical implementation of the HRI tablet computer application illustrating the action selection procedure: single object selection (a), multi object selection (b)

IV. Evaluation

This work describes the development of a shared autonomy HRI for tablet computers implementing an object-centered and knowledge-driven approach. The capabilities of the overall system are evaluated in an use-case scenario with practical relevance for modern service robots according to Sec. IV-A. An user study has been conducted in Sec. IV-B in order to measure the usability of the HRI. All experiments are executed with the humanoid robot Rollin' Justin [27] and a Samsung Galaxy Note 8 tablet computer.

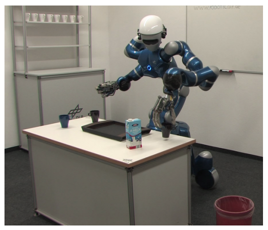
A. Experiment

Cleaning up an untidy environment is one of the most frequent tasks in human households. In order to solve this task objects have to be relocated which results in pick and place sequences. For this the robot has to figure out which objects have to be relocated and where to place them. As the expected target location of an object changes w. r. t. the environmental context, hardcoding a specific target location is insufficient. E.g. a dirty mug should be placed in the dishwasher but a book that is currently read should stay in its place near the couch. The contextual awareness needed to satisfactorily solve the task is a hard problem for service robots. Furthermore, relocating an object may require to open and close designated containers such as cupboards or

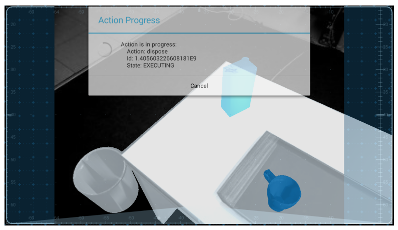


Action is in progress:
Action: place
Id: 1.405603016450592E9
State: EXECUTING

(a)

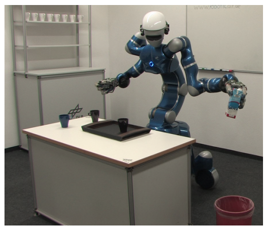


(b)

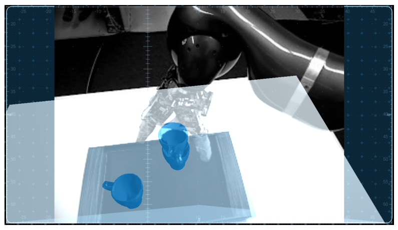


Action is in progress:
Action: dispose
Id: 1.405603226608181E9
State: EXECUTING

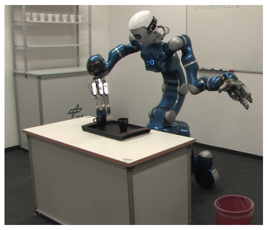
(c)



(d)



(e)



(f)

Fig. 8. Procedure of the experimental clean up task: Picking *mug3* from the table, operator view (a), external view (b). Disposing *milk1*, operator view (c), external view (d). Stacked *mug1* on *mug3*, operator view (e), external view (f).

drawers. Due to these problems and the ambiguity of the problem statement in general, a generic approach to solve clean up tasks can hardly be defined. By applying shared autonomy, some of these problems can be avoided by using the contextual and cognitive capabilities of the operator.

For our experiment, operator and robot are spatially separated, so the operator can only gain information about the state of the robot and its environment via the tablet computer HRI. The robot is prelocalized in its environment consisting of priorly known objects. An artificial clean up scenario was constructed based on the example introduced in Sec. II-A Fig. 2 which was extended by a milk carton located on the table and a trash can placed next to the table on the floor. The robot has to move the three mugs from the table to the tray and has to put the empty milk carton into the trash can. Fig. 8 shows the experimental procedure.

Our system allows the operator to gain situational awareness in the remote environment by turning the head of the robot while observing the live video stream from the head mounted camera of the robot.

The action scheduling is done by the operator via our tablet HRI and can be split in three steps: (1) placing the mug which blocks the milk carton on the tray, (2) putting the milk carton in the trash can, and (3) placing the remaining mugs on the tray.

1) The robot can reliably place mugs on a tray with sufficient free space. Therefore *mug3* is directly commanded to be *placed* on the tray.
2) *Picking* the *milk1* object and *disposing* it in the *trash_can* is commanded using single-step actions to demonstrate the stepwise execution of an otherwise multi-step action. This approach allows the operator to monitor the state of the robot after action execution and to be responsive to execution problems.
3) *Mug2* is directly *placed* on the tray. *Mug1* is commanded to be *stacked* on the already placed *mug2* due to little remaining free space on the *tray*.

The experiment showed the capability to solve the ambiguous clean up task using our HRI by commanding robot actions on a high level of abstraction. The HRI allowed the operator to observe the task and respond to situations that otherwise would have been perceptive and cognitive challenging for our robot. The accompanied video illustrates the experiment in detail.

B. User Study

For a more general evaluation of the usability of our HRI, we conducted a user study with 6 female and 14 male participants aged between 20 and 52 (average 24.5). The participants were familiar with modern smartphones or tablet computers but had no prior experience with our HRI or the Rollin' Justin robot. During the study, each participant was spatially separated from the robot and introduced to the HRI (max. 10 minutes). Afterwards three tasks had to be solved: (T1) pick a specified object, (T2) find and pick a specified object, and (T3) find, pick and place a specified object in cluttered scene on a specified target. The duration of the execution and the number of practised clicks were captured and set into relation with the results of an expert-user to gain simple objective measures. The *System Usability Scale (SUS)* [28] was used in addition to measure the usability of the HRI on a scale from 0 (bad) to 100 (good).

All participants succeed in all tasks and reached averaged execution durations between 2.0 (σ 0.5) and 4.1 (σ 2.4) times the expert-user result and averaged numbers of practised clicks between 1.5 (σ 0.4) and 2.3 (σ 0.8) times the expert-user result. These results show, that the developed HRI can be used to achieve simple tasks after only minimal training. In addition, a good usability of the HRI is expected as the number of practised clicks is near the expert-user results. This is confirmed by a high SUS of 87.5 (σ 6.25).

As there existed no prior HRI allowing non-expert users to command manipulation capabilities of Rollin' Justin, the positive results encourage us to further develop the system. Comparing different UI approaches is subject of future work.

V. Summary

In this paper we presented a shared autonomy HRI that enables an operator to command high level actions rather than robot capabilities. We argued that the cognitive and perceptive capabilities of the operator together with hybrid reasoning mechanisms of the robot allow for a semi-autonomous execution of otherwise challenging robot manipulation tasks. We efficiently lowered the cognitive load of the operator by using the knowledge of the robot about its environment w. r. t. current manipulation possibilities. A point-and-click paradigm based on this knowledge was used to intuitively command robot actions by selecting the objects to be manipulated on a touchscreen.

This approach simplified the UI of the HRI and provided access to priorly known actions for manipulation of priorly known objects that are in the field of view of the robot. Objects the robot cannot see can't be selected by the operator, thus e.g. fetching tasks for objects located in other rooms or inside cupboards cannot be commanded. We plan to solve this issue by adding other levels of shared autonomy such as navigation in semantic maps in future work.

A prototypical tablet computer application was used to successfully evaluate the proposed concepts in a clean up scenario with the robot Rollin' Justin. In a user study, we showed that our system is highly intuitive and enables novice users to easily command a humanoid service robot.

VI. Acknowledgments

The authors would like to thank Florian Schmidt for his support with the multi-platform communication.

References

[1] T. Hulin, K. Hertkorn, P. Kremer, S. Schätzle, J. Artigas, M. Sagardia, F. Zacharias, and C. Preusche, "The dlr bimanual haptic device with optimized workspace," in *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2011, pp. 3441–3442.

[2] C. Stanton, A. Bogdanovych, and E. Ratanasena, "Teleoperation of a humanoid robot using full-body motion capture, example movements, and machine learning," in *Proc. of the Australasian Conference on Robotics and Automation (ACRA)*, 2012.

[3] S. Amiri, R. Fazel-Rezai, and V. Asadpour, "A review of hybrid brain-computer interface systems," *Advances in Human-Computer Interaction*, 2013.

[4] M. Mast, M. Burmester, K. Krüger, S. Fatikow, G. Arbeiter, B. Graf, G. Kronreif, L. Pigini, D. Facal, and R. Qiu, "User-centered design of a dynamic-autonomy remote interaction concept for manipulation-capable robots to assist elderly people in the home," *Journal of Human-Robot Interaction*, vol. 1, no. 1, 2012.

[5] P. Rouanet, P.-Y. Oudeyer, F. Danieau, and D. Filliat, "The impact of human–robot interfaces on the learning of visual objects," *IEEE Transactions on Robotics*, vol. 29, no. 2, pp. 525–541, 2013.

[6] M. A. Goodrich, J. W. Crandall, and E. Barakova, "Teleoperation and beyond for assistive humanoid robots," *Reviews of Human Factors and Ergonomics*, vol. 9, no. 1, pp. 175–226, 2013.

[7] D. A. Lazewatsky and W. D. Smart, "An inexpensive robot platform for teleoperation and experimentation," in *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2011, pp. 1211–1216.

[8] S. M. Jung, D. Choi, K. Kwon, and J. W. Jeon, "A sketch-based interface for remote robot control on an online video screen," in *Proc. of the IEEE International Conference on Consumer Electronics (ICCE)*, 2013, pp. 428–429.

[9] D. Labonte, P. Boissy, and F. Michaud, "Comparative analysis of 3-d robot teleoperation interfaces with novice users," *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics*, vol. 40, no. 5, pp. 1331–1342, 2010.

[10] T. L. Chen, M. Ciocarlie, S. Cousins, P. Grice, K. Hawkins, K. Hsiao, C. C. Kemp, C.-H. King, D. A. Lazewatsky, A. Leeper, H. Nguyen, A. Paepcke, C. Pantofaru, W. D. Smart, and L. Takayama, "Robots for humanity: A case study in assistive mobile manipulation," *IEEE Robotics & Automation Magazine, Special issue on Assistive Robotics*, vol. 20, 2013.

[11] S. Muszynski, J. Stuckler, and S. Behnke, "Adjustable autonomy for mobile teleoperation of personal service robots," in *Proc. of the IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, 2012, pp. 933–940.

[12] A. Correa, M. R. Walter, L. Fletcher, J. Glass, S. Teller, and R. Davis, "Multimodal interaction with an autonomous forklift," in *Proc. of the ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, 2010, pp. 243–250.

[13] S. Hashimoto, A. Ishida, M. Inami, and T. Igarashi, "Touchme: An augmented reality based remote robot manipulation," in *Proc. of the International Conference on Artificial Reality and Telexistence*, 2011.

[14] F. Ferland, F. Pomerleau, C. T. Le Dinh, and F. Michaud, "Egocentric and exocentric teleoperation interface using real-time, 3d video projection," in *Proc. of the ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, 2009, pp. 37–44.

[15] K. Liu, D. Sakamoto, M. Inami, and T. Igarashi, "Roboshop: Multi-layered sketching interface for robot housework assignment and management," in *Proc. of the SIGCHI Conference on Human Factors in Computing Systems (CHI)*, 2011, pp. 647–656.

[16] D. Sakamoto, K. Honda, M. Inami, and T. Igarashi, "Sketch and run: A stroke-based interface for home robots," in *Proc. of the SIGCHI Conference on Human Factors in Computing Systems (CHI)*, 2009, pp. 197–200.

[17] H. M. Do, C. J. Mouser, Y. Gu, W. Sheng, S. Honarvar, and T. Chen, "An open platform telepresence robot with natural human interface," in *Proc. of the IEEE Annual International Conference on Cyber Technology in Automation, Control and Intelligent Systems (CYBER)*, 2013, pp. 81–86.

[18] G. Paravati, A. Sanna, F. Lamberti, and C. Celozzi, "A reconfigurable multi-touch framework for teleoperation tasks," in *Proc. of the Conference on Emerging Technologies & Factory Automation (ETFA)*, 2011.

[19] M. Ciocarlie, K. Hsiao, A. Leeper, and D. Gossow, "Mobile manipulation through an assistive home robot," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012, pp. 5313–5320.

[20] Y. Suzuki, M. Terashima, K. Takeuchi, and H. Kuzuoka, "An object-defined remote robot control interface," *Journal of Information Processing*, vol. 21, no. 2, pp. 304–314, 2013.

[21] R. Fung, S. Hashimoto, M. Inami, and T. Igarashi, "An augmented reality system for teaching sequential tasks to a household robot," in *Proc. of the IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, 2011, pp. 282–287.

[22] D. Leidner, C. Borst, and G. Hirzinger, "Things are made for what they are: Solving manipulation tasks by using functional object classes," in *Proc. of the IEEE/RAS International Conference on Humanoid Robots (ICHR)*, 2012, pp. 429–435.

[23] M. Ghallab, A. Howe, D. Christianson, D. McDermott, A. Ram, M. Veloso, D. Weld, and D. Wilkins, "Pddl—the planning domain definition language," *AIPS98 planning committee*, vol. 78, no. 4, pp. 1–27, 1998.

[24] D. Leidner and C. Borst, "Hybrid reasoning for mobile manipulation based on object knowledge," in *Workshop on AI-based Robotics at IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2013.

[25] D. Leidner, A. Dietrich, F. Schmidt, C. Borst, and A. Albu-Schaffer, "Object-centered hybrid reasoning for whole-body mobile manipulation," in *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 1828–1835.

[26] S. Gebhardt, S. Pick, F. Leithold, B. Hentschel, and T. Kuhlen, "Extended pie menus for immersive virtual environments," *IEEE Transactions on Visualization and Computer Graphics*, vol. 19, no. 4, pp. 644–651, 2013.

[27] C. Borst, T. Wimbock, F. Schmidt, M. Fuchs, B. Brunner, F. Zacharias, P. R. Giordano, R. Konietschke, W. Sepp, S. Fuchs, *et al.*, "Rollin' justin - mobile platform with variable base," in *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2009, pp. 1597–1598.

[28] J. Brooke, "Sus: a 'quick and dirty' usability scale," *Usability Evaluation In Industry*, vol. 189, pp. 189–194, 1996.