Incremental Language Understanding for Online Motion Planning of Robot Manipulators

Mitchell Abrams*,1, Thies Oelerich*,2, Christian Hartl-Nesic2, Andreas Kugi2,3, Matthias Scheutz1

***Abstract*— Human-robot interaction requires robots to process language incrementally, adapting their actions in real-time based on evolving speech input. Existing approaches to language-guided robot motion planning typically assume fully specified instructions, resulting in inefficient stop-and-replan behavior when corrections or clarifications occur. In this paper, we introduce a novel reasoning-based incremental parser which integrates an online motion planning algorithm within the cognitive architecture. Our approach enables continuous adaptation to dynamic linguistic input, allowing robots to update motion plans without restarting execution. The incremental parser maintains multiple candidate parses, leveraging reasoning mechanisms to resolve ambiguities and revise interpretations when needed. By combining symbolic reasoning with online motion planning, our system achieves greater flexibility in handling speech corrections and dynamically changing constraints. We evaluate our framework in real-world human-robot interaction scenarios, demonstrating online adaptions of goal poses, constraints, or task objectives. Our results highlight the advantages of integrating incremental language understanding with real-time motion planning for natural and fluid human-robot collaboration. The experiments are demonstrated in the accompanying video at `www.acin.tuwien.ac.at/42d5`.**

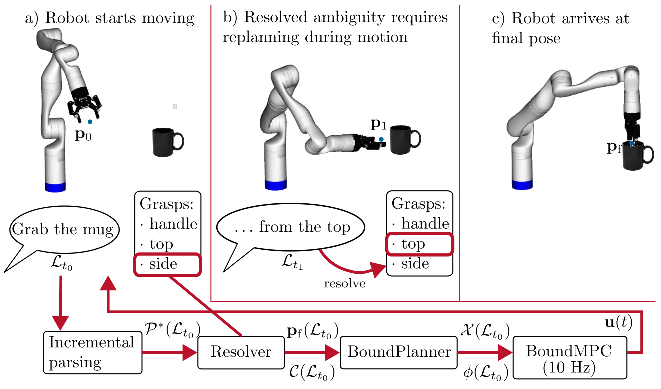
a) Robot starts moving
b) Resolved ambiguity requires replanning during motion
c) Robot arrives at final pose
$\mathbf{p}_0$
$\mathbf{p}_1$
$\mathbf{p}_f$
Grab the mug
$\mathcal{L}_{t_0}$
Grasps: · handle · top · side
. . . from the top
$\mathcal{L}_{t_1}$
resolve
Grasps: · handle · top · side
Incremental parsing
$\mathcal{P}^*(\mathcal{L}_{t_0})$
Resolver
$\mathbf{p}_f(\mathcal{L}_{t_0})$
$\mathcal{C}(\mathcal{L}_{t_0})$
BoundPlanner
$\mathcal{X}(\mathcal{L}_{t_0})$
$\phi(\mathcal{L}_{t_0})$
BoundMPC (10 Hz)
$\mathbf{u}(t)$

Fig. 1. Schematic of the incremental motion planning framework. The natural language instruction $\mathcal{L}_{t_0}$ is parsed to $\mathcal{P}^*(\mathcal{L}_{t_0})$. The resolver translates this to a goal pose $\mathbf{p}_f(\mathcal{L}_{t_0})$ and a set of constraints $\mathcal{C}(\mathcal{L}_{t_0})$. The ambiguity in of the grasping pose is resolved randomly to a side grasp. Based on this, BoundPlanner [6] computes a set of admissible states $\mathcal{X}(\mathcal{L}_{t_0})$ and cost function parameters $\phi(\mathcal{L}_{t_0})$ and passes it to BoundMPC [7] which computes the joint input $\mathbf{u}(t)$ at 10 Hz. As the grasp assumption is false, the human instructor utters the language instruction $\mathcal{L}_{t_1}$ at time $t_1 > t_0$ to correct the robot. This correction requires a replanning at pose $\mathbf{p}_1$ which uses the same steps as before. The robot then reaches the final pose $\mathbf{p}_f$.

I. INTRODUCTION

As robots become more involved in everyday life, they are expected to interact seamlessly with humans. The interactions involve commanding the robot to perform a certain task or asking for help. Language instructions play a crucial role in these situations and robots need to understand human intentions. However, natural language is often ambiguous, and humans may need to correct or specify their instructions on the fly. An example is given in Fig. 1. The instruction "*Grab the mug*" does not specify a particular way to grasp the mug, whether by the handle or from the top. As the robot starts reaching for the handle, the human can add "*from the top*" to the instruction, to which the robot needs to react in real time. Such a behavior requires a continuous interaction between language understanding and motion planning. This is lacking in current works, which often assume natural language instructions to be given in text form [1], [2], [3]. Real speech input is used in [4], but the setting is limited, and the instruction is still parsed in its entirety but not incrementally. When the robot waits for the full instruction in order to begin planning a suitable motion and then stops and replans once the motion needs to be adapted, then this is time-consuming and frustrating for the human instructor. Existing work in [5] uses language corrections but requires manual control interventions and only wrong actions are corrected, but the task is not changed. Our work specifically focuses on real-time speech parsing with online motion planning, mandatory for real human-robot interaction.

Incremental language processing offers flexibility, allowing robots to process linguistic input as it arrives rather than waiting for full utterance completion. Humans naturally employ incremental processing, continuously updating their understanding and refining interpretations based on evolving context [8], [9]. Similarly, a robot that processes language incrementally can begin formulating an action plan based on an initial command, and dynamically revise its plan as new linguistic constraints emerge.

There are various incremental language processing systems; some focus on syntactic parsing [10] or semantic parsing [11], [12], while others are designed specifically for robotic agents [13] or built into incremental dialogue processing software such as InproTK [14] and Retico [15]. We argue that incremental language processing can benefit online replanning in robotic agents, in contrast to their non-

This work was not supported by any organization.

* Authors contributed equally
1 Authors are with the Human-Robot Interaction Lab, Tufts University, Boston, USA, {`mitchell.abrams, matthias.scheutz`}`@tufts.edu`
2 Authors are with the Automation and Control Institute (ACIN), TU Wien, Vienna, Austria, {`oelerich, hartl, kugi`}`@acin.tuwien.ac.at`
3Andreas Kugi is with the center for Vision, Automation & Control, AIT Austrian Institute of Technology GmbH, Vienna, Austria `andreas.kugi@ait.ac.at`

incremental counterparts. Non-incremental parsers—such as Abstract Meaning Representation (AMR) parsers [16], dependency parsers [17], or Large Language Models (LLMs) used for text-based robot instructions [18]—require utterances to be complete before parsing, delaying action planning and making real-time interaction infeasible.

As mentioned, a motion planner that can react to changes in the instructions online is needed in order to couple it with an incremental parsing system. This includes motion constraints that may be added incrementally. Imagine the robot manipulating a cup filled with a liquid. The robot might not know that it needs to keep it upright to avoid spilling, which the human can communicate with the corrective statement "*keep the cup upright*". This new motion constraint needs to be taken into account during the motion of the robot. Considering Cartesian constraints in an online motion planner is challenging. Resolving ambiguity of language instructions needs to work online and promptly because otherwise, the actions taken by the robot might not be reversible anymore. When the robot transports a filled cup but is not aware of the upright constraint, an online correction is able to limit the damage. When this is not the case, the robot might spill the liquid during its motion, which is not reversible.

Recently, [7] proposed an optimization-based approach using a Cartesian bounded reference path which is used in this work in combination with the bounded reference path planner [6]. This combination is used for online motion planning in this work. It is able to rigorously communicate constraints from the language input to the motion planner using a model-based approach with complete consideration of the environment. The motion instructions are created by the novel incremental language parser that utilizes explicit reasoning to parse utterances incrementally and understand at which point in time the utterance is meaningful to send to the underlying motion planner. This way it is tailored specifically to robot motion planning. The proposed approach is demonstrated on a 7-DoF robot manipulator in four scenarios. These scenarios demonstrate how the proposed framework handles changes in goal pose, adding motion constraints, i.e., a region avoidance and upright orientation constraint, and changing the speed of the motion while the robot is moving. We further demonstrate that using an offline motion planner degrades performance.

The main contributions of this paper are

- a novel incremental language parser specifically tailored to interact with robot motion planning.
- the tight integration of this incremental language parser with online motion planning based on BoundMPC [7] and BoundPlanner [6]. This novel combination allows fast acting in human-robot collaborations and quick corrections of undesired motions.

II. RELATED WORK

A. Incremental Processing

Humans process language incrementally, interpreting linguistic input as it is received rather than waiting for full utterance completion [8], [9]. This ability enables rapid adaptation in interactive settings, allowing speakers and listeners to dynamically adjust their actions based on evolving context [19], [20]. Psycholinguistic research has shown that humans make predictions about meaning in real time, guided by syntactic, semantic, and pragmatic constraints [21].

Motivated by these human capabilities, computational systems for incremental natural language understanding have been developed to support real-time processing and early action execution [22]. These systems enable dialogue components to process partial utterances, reducing response latency and improving interactive fluency.

Several approaches to incremental language processing have emerged, each focusing on different aspects of the problem. Some systems prioritize incremental syntactic parsing [23], [10], allowing for continuous refinement of the sentence structure as new words arrive. Others emphasize incremental semantic interpretation [11], [12], where meaning is continuously updated based on available linguistic and contextual information. One important subfield is incremental reference resolution, in which systems dynamically resolve a referent without waiting for a fully disambiguated description [24].

While these systems have demonstrated the advantages of incremental processing, most approaches are designed for interpretation rather than planning and action execution. In situated environments, incremental processing must go beyond language and integrate with an agent's perception, reasoning, and action planning. Some systems have begun to explore this intersection by incorporating multimodal cues such as gaze and gesture to improve reference resolution [25]. Others have focused on real-time robotic interaction, such as the RISE system, which allows robots to maintain multiple possible interpretations of an unfolding utterance [26]. However, a critical gap remains: most incremental systems do not support real-time action planning and execution based on partially complete utterances.

B. Motion Planning

Motion planning for robot manipulators is a challenging and active research field. It is broadly divided into offline [27], [28] and online motion planning [29], [7]. Offline motion planning is often employed in industrial settings where minimum-time trajectories can improve the production output or for very precise tasks which are difficult to compute. However, when the robot is acting in an uncertain or dynamic environment, e.g., in an environment shared with humans, the robot must be able to adapt its behavior online [30]. The needed real-time capability can be achieved by planning only over a finite time horizon. Popular methods include sampling- and optimization-based model predictive control (MPC) [29], [7], [31] and reinforcement learning [32]. However, finite horizons for planning often yield local minima due to obstacles and joint limits. Thus, the works [1] and [6] split the planning problem into reference path planning and trajectory planning. The reference path is planned from the initial to the goal pose in Cartesian space and guides the finite-horizon joint-trajectory planning. This separation enables real-time planning. Popular path planning methods include RRT [33] and convex sets [34], [1], [6]. In this work we use BoundPlanner [6] in combination with BoundMPC [7], which was shown to be computationally efficient and is able to adhere to Cartesian bounds.

C. End-to-End Learning

Another interesting development is the emergence of end-to-end learning as used in [5], [4]. However, this lacks any guarantees of constraint satisfaction and structured incremental understanding. Therefore, this approach is not further considered in this work.

III. INCREMENTAL LANGUAGE PROCESSING

[35] present a high-level framework for designing incremental language processing systems and outline key architectural considerations: *modularity*, *granularity*, and *revisability*. In this section, we introduce an incremental parsing approach that enables a robotic agent to process language in real time, submit plans on the fly, and dynamically adjust its actions based on evolving linguistic input. Our system bridges this gap by supporting: 1) Partial parsing (*modularity*): the system generates usable interpretations before an utterance is complete; 2) Integration with motion planning (*granularity*): words and partial parses are used to submit and update robot action plans; and 3) Real-time plan revision (*revisability*): the system dynamically updates actions and utterance interpretations when new constraints are introduced. By combining incremental parsing with the online motion planning framework described in Section IV, our system efficiently handles dynamic human-robot interactions, adapting to spoken corrections and clarifications utilizing replanning during the robot's motion.

Algorithm 1: Incremental Chart Parsing

```
   Input: Language Instruction $\mathcal{L}_t = (w_0, \dots, w_{n-1})$,
   Dictionary $D$
   Output: Parsed utterance $\mathcal{P}(\mathcal{L}_t)$
1  if chart uninitialized then
2      InitializeChart($n$)
3  else
4      ExpandChart($n$)
5  foreach $w_i \in \mathcal{L}_t$ do
       // Add initial word node to chart
6      $\mathcal{E}_i \leftarrow D$.lookup($w_i$)
7      if $\mathcal{E}_i \neq \emptyset$ then
8          AddNode($\mathcal{E}_i, i, i$)
9  for $s = 2$ to $n$ do
10     for $j = 0$ to $n - s$ do
11         $k \leftarrow j + s - 1$
12         for $m = j$ to $k - 1$ do
13             foreach $(L, R) \in chart[j][m], chart[m+1][k]$ do
                   // Try combining constituents
14                 $\mathcal{N} \leftarrow$ Combine($L, R$)
15                 if $\mathcal{N} \neq \emptyset$ then
                       // Add new node to chart
16                     chart[$j$][$k$] $\leftarrow$ chart[$j$][$k$] $\cup$ {Node($L, R, \mathcal{N}$)}
17 return best parse $\mathcal{P} \in$ chart[0][$n-1$]
```

Our parser constructs a hierarchical structure in several stages. First, during lexical processing, the words are assigned their respective syntactic categories and semantic representations (using CCG-style (combinatory categorial grammar) parsing) combining syntax and semantics based on predefined grammar rules. Words are dynamically inserted into the chart and stored in separate cells as they arrive. The parser then begins phrase-level combination. The final parse is selected from the final cell of the chart. If a user stops mid-utterance, such as saying *"grab the mug..."* without completing or adding more to the utterance (*"grab the mug by the top"*), the parser maintains a partial parse and uses it as a basis for future input. This ensures that parsing remains fluid and can accommodate dynamically evolving utterances.

The incremental chart parsing algorithm (**Algorithm 1**) presents a more formal description. The algorithm takes as input an utterance $\mathcal{L}_t = (w_0, w_1, \dots, w_i)$ at time t and a dictionary D containing stored grammar rules. It outputs a parsed representation $\mathcal{P}(\mathcal{L}_t)$. The chart data structure is initialized (**lines 1–4**) if not already allocated; otherwise, it is expanded to accommodate newly received words.

Each word w_i in the utterance undergoes lexical processing (**lines 5–8**). The function D.lookup(w_i) retrieves a set of grammar rules $\mathcal{E}_i$ associated with the word. If the word is unknown ($\mathcal{E}_i = \emptyset$), it is skipped. Otherwise, a node[1] is created from these rules and inserted into the chart at position (i, i), see Table I. Once all words are inserted, the parser performs phrase combination using a bottom-up dynamic programming approach (**lines 9–16**). It iterates over spans of length s, progressively merging smaller phrases into larger constituents. At each span, split points m are considered, allowing for binary branching combinations of subparses. The function `Combine`(L, R) checks whether two nodes can be merged using grammatical constraints.

If a valid rule $\mathcal{N}$ permits the combination of a left parse L and a right parse R, a new node is created and inserted into the chart (**line 14**). The process continues until the entire utterance is analyzed. The final parse is retrieved from chart cell $[0, n-1]$ (**line 17**), which stores the final interpretations. Multiple parses of the same words can be stored in the chart, allowing a reasoning mechanism to return the best interpretation[2]. The structured storage of alternative parses also allows the system to defer disambiguation until further context is available. Our parser operates with a worst-case time complexity of $O(n^3)$, similar to CYK parsing [36]. The proposed parser is designed to work with the English language. Extending this to other languages requires an additional translation step or language-specific adaptions.

A. Parsing Walkthrough

Table I illustrates the incremental chart parse for *"grab the mug by the top."* The parsing process begins at the lexical

[1]Nodes are created using the constructor `new Node(List<Entry> entries)`, which generates lexical nodes from dictionary entries. As parsing progresses, the method `combine(Node left, Node right)` merges compatible nodes into higher-level phrases.

[2]Multiple parses of are stored in the last cell and a single parse can be returned depending on the context (e.g. available referents or objects) of the utterance that reflects the best interpretation.

Idx	**0**	**1**	**2**	**3**	**4**	**5**
0	grab	VP $\rightarrow$ grab	VP $\rightarrow$ grab the mug			**S $\rightarrow$ grab (the mug) (by the top)** S $\rightarrow$ grab (the mug by the top)
1		the	NP $\rightarrow$ the mug			
2			mug			
3				by		**PP $\rightarrow$ by the top** PP $\rightarrow$ by the top (modifies NP)
4					the	NP $\rightarrow$ the top
5						top

TABLE I

PARSING CHART FOR "GRAB THE MUG BY THE TOP"

level, where each word is inserted into its corresponding diagonal cell. The first word, *"grab"*, is identified as a verb and stored at position $(0,0)$. Next, *"the"* and *"mug"* (at positions $(1,1)$ and $(2,2)$, respectively) are recognized as part of a noun phrase (NP $\rightarrow$ *"the mug"*). At the phrase combination stage, the parser merges *"the"* and *"mug"* into NP $\rightarrow$ *"the mug"*, stored at $(1,2)$. This NP is then combined with the verb at $(0,0)$, forming a VP $\rightarrow$ *"grab the mug"*, a candidate parse for a complete verb phrase at $(0,2)$. This is the first completed parse that can be submitted as a goal for the agent ($\mathcal{P}^*(\mathcal{L}_{t_0})$ at time t_0 rather than *"grab the"* ($\mathcal{P}(\mathcal{L})$). We use a first-order logic (FOL) style semantic representation[3]:

```
INSTRUCT(speaker,listener,graspObject(listener,mug))
```

While the agent starts to achieve this goal (grabbing the relevant mug), the parse continues until it gets to another completed parse $\mathcal{P}^*(\mathcal{L}_{t_1})$ at time t_1. The next word, *"by"*, is inserted at $(3,3)$ as a preposition (PP $\rightarrow$ *"by"*). Similarly, *"the top"* is recognized as a noun phrase (NP $\rightarrow$ *"the top"*) and stored at $(4,5)$. The parser then combines these elements into PP $\rightarrow$ *"by the top"*, stored at $(3,5)$.

The final parse selection occurs at cell $(0,5)$, where the parser considers two interpretations: (1) **Best Parse (Bold):** S $\rightarrow$ *"grab (the mug) (by the top)"*, where *"by the top"* modifies the verb, indicating the manner of grabbing. (2) **Alternative Parse (Gray):** S $\rightarrow$ *"grab (the mug by the top)"*, where *"by the top"* modifies the noun, implying a location.

This approach ensures that multiple interpretations are retained within the chart, enabling flexible reasoning about attachment ambiguities in real time. Each parse stored in `chart[0][n-1]` can be evaluated and resolved using additional context or external constraints. The final parse places additional constraints on the original plan ($\mathcal{P}^*(\mathcal{L}_{t_1})$):

```
INSTRUCT(speaker,listener,graspObject
        (listener,mug), by(mug, top))
```

IV. ONLINE MOTION PLANNING

The incremental language parsing system from Section III is used to parse a motion instruction $\mathcal{L}_{t_k}$ at time t_k for a robot manipulator. Once a meaningful phrase $\mathcal{P}^*(\mathcal{L}_{t_k})$ is obtained, the motion planner needs to react online to the adaptions. The resolver in Fig. 1 resolves $\mathcal{P}^*(\mathcal{L}_{t_k})$ into a desired final pose $\mathbf{p}_\mathrm{f}(\mathcal{L}_{t_k})$ and a set of constraints $\mathcal{C}(\mathcal{L}_{t_k})$ as inputs to the motion planner. For the purpose of this work, the goal poses and obstacles are assumed to be known such that the resolver simply maps the parses, e.g., from Section III-A, to poses and constraints. In future work, this will be extended to include vision information. For example, the specification *"keep the cup upright"* will add a constraint on the orientation to $\mathcal{C}(\mathcal{L}_{t_k})$.

[3]This representation includes the speaker intent (`INSTRUCT`) and the core action (`graspObject`) on an object (`mug`). The object reference (`mug`) can resolve to a specific object in the environment known by the agent.

A. From Parsed Instruction to Safe Robot Motion

In this work, a combination of global reference path planning in the Cartesian space with subsequent trajectory planning in the joint space over a finite time horizon is used. A reference path π is planned by BoundPlanner [6] and bounded by convex sets such that the robot's end effector is collision free if it remains within the bounds along π that ensure the satisfaction of constraints $\mathcal{C}(\mathcal{L}_{t_k})$. These bounds define the set of permissible states

$$\mathcal{X}(\mathcal{L}_{t_k}) = \mathcal{X}_\mathrm{safe}(\mathcal{L}_{t_k}) \cup \mathcal{X}_\mathrm{task}(\mathcal{L}_{t_k}) \cup \mathcal{X}_\mathrm{robot} \tag{1}$$

for the robot, which constitutes safe states $\mathcal{X}_\mathrm{safe}(\mathcal{L}_{t_k})$, task-specific states $\mathcal{X}_\mathrm{task}(\mathcal{L}_{t_k})$, and kinematic and dynamic states of the robot $\mathcal{X}_\mathrm{robot}$. The set of safe states $\mathcal{X}_\mathrm{safe}(\mathcal{L}_{t_k})$ will always ensure obstacle avoidance. Convex sets define the task set $\mathcal{X}_\mathrm{task}(\mathcal{L}_{t_k})$ and the safety set $\mathcal{X}_\mathrm{safe}(\mathcal{L}_{t_k})$ for the position and orientation of the robot's end effector and the collision avoidance of the robot's kinematic chain. The set $\mathcal{X}_\mathrm{robot}$ is independent of the instruction $\mathcal{L}_{t_k}$ as the robot's limits do not change. The original formulation of BoundPlanner [6] does not handle orientation constraints. This work lifts this limitation by using box constraints for the orientation formulation introduced in [7]. The parsed language instruction $\mathcal{P}^*(\mathcal{L}_{t_k})$ defines $\mathbf{p}_\mathrm{f}(\mathcal{L}_{t_k})$ and $\mathcal{C}(\mathcal{L}_{t_k})$, which need to be translated to the set $\mathcal{X}(\mathcal{L}_{t_k})$, and the parameters $\phi(\mathcal{L}_{t_k})$ using BoundPlanner. The parameters $\phi(\mathcal{L}_{t_k})$ parametrize the cost function for the subsequent trajectory optimization influencing the shape and speed of the trajectory. Specifically, this work presents the following adaptions in several examples:

- $\mathbf{p}_\mathrm{f}(\mathcal{L}_{t_k}) \rightarrow \mathcal{X}_\mathrm{task}(\mathcal{L}_{t_k})$: Change of the goal pose by replanning the reference path,
- $\mathcal{C}(\mathcal{L}_{t_k}) \rightarrow \mathcal{X}_\mathrm{safe}(\mathcal{L}_{t_k})$: Change of the avoidable obstacles or allowed orientations,
- $\mathcal{C}(\mathcal{L}_{t_k}) \rightarrow \phi(\mathcal{L}_{t_k})$: Change of the motion speed as a soft constraint.

The adaptions will be discussed in more detail in Section V.

The sets and parameters introduced above are further provided to BoundMPC [7], which uses online optimization to compute the joint input $\mathbf{u}(t)$ over a finite time horizon from time $t_0 \geq t_k$ to the end time t_e. The finite horizon planning problem is formulated as

$$\begin{aligned}
\min_{\mathbf{u}(t)} \quad & c_{\phi(\mathcal{L}_{t_k})}(\mathbf{x}(t_\mathrm{e}), \mathbf{u}(t_\mathrm{e})) + \int_{t_0}^{t_\mathrm{e}} J_{\phi(\mathcal{L}_{t_k})}(\mathbf{x}(t), \mathbf{u}(t)) \mathrm{d}t \\
\text{s.t.} \quad & \mathbf{x}(t_0) = \mathbf{x}_0 \\
& \mathbf{u}(t_0) = \mathbf{u}_0 \\
& \mathbf{x}(t) \in \mathcal{X}(\mathcal{L}_{t_k}) \quad \forall t : t_0 \leq t \leq t_\mathrm{e} \\
& \mathbf{u}(t) \in \mathcal{U} \quad \forall t : t_0 \leq t \leq t_\mathrm{e} \ ,
\end{aligned} \tag{2}$$

with the state $\mathbf{x}(t)$, the input $\mathbf{u}(t)$, and their initial values $\mathbf{x}_0$ and $\mathbf{u}_0$ at t_0. The objective function is composed of the terminal cost $c_{\phi(\mathcal{L}_{t_k})}(\mathbf{x}(t_\mathrm{e}), \mathbf{u}(t_\mathrm{e}))$ and the stage cost $J_{\phi(\mathcal{L}_{t_k})}(\mathbf{x}(t), \mathbf{u}(t))$, which are parametrized by $\phi(\mathcal{L}_{t_k})$. The permissible inputs are given by the set $\mathcal{U}$. For more information on BoundPlanner and BoundMPC, the reader is referred to [7] and [6], respectively. The instruction $\mathcal{L}_{t_k}$ is constant in (2) because it is not predicted over the time horizon.

Remark. *This work focuses on adding constraints that need to be enforced immediately at time t_k, e.g., reacting to a new obstacle, which may render the current state infeasible. To overcome this, the respective constraints are enforced using slack variables. This is a common practice in optimization [37].*

B. Taxonomy of Constraints in Incremental Language Processing

Incremental language processing enables dynamic adaptation of robot motion plans based on evolving linguistic constraints $\mathcal{C}(\mathcal{L}_t)$. These constraints may be explicitly stated in speech or inferred from context and can emerge at different stages of execution. We categorize them into six types: manner, target, object, action, safety, and sequential constraints, as summarized in Table II.

Manner constraints adjust how an action is performed without altering its goal. These constraints are often introduced via adverbs or prepositional phrases, such as "Pass the screwdriver *but go faster*," which modifies the execution speed. Target constraints refine the intended destination. For instance, "Put the apple in the box... *no, the one on the right*" forces the robot to update its spatial goal in real time. Object constraints resolve referential ambiguity when multiple items fit a description. If a robot initially reaches for the wrong object, additional clarification—such as "Grab the mug... *no, the blue one*"—helps disambiguate the target. Action constraints redefine or modify the task itself. A command like "Grab the apple" might initially be interpreted as a grasping action, but a correction such as "*no, push it*" redirects the execution to a different motion. Safety constraints introduce requirements to prevent damage or failure. Commands like "Move the cup *upright*" ensure that the robot adjusts its trajectory to maintain a stable orientation. Sequential constraints establish dependencies between actions. A directive like "Pick up the apple *after you put down the mug*" enforces a sequence that the robot must integrate dynamically into its plan.

This taxonomy highlights the need for an incremental parsing system capable of maintaining multiple candidate interpretations and refining them in real time as new constraints emerge.

V. EXPERIMENTS

In this section, the proposed framework is evaluated in three experiments on the real robot:

1) Grabbing a mug and changing the desired grasp direction during the motion,
2) transferring a mug filled with liquid where additional constraints (keeping the mug upright and avoiding going over a laptop) are added during the motion, and
3) giving a screwdriver to a human where the human controls the speed of the robot's motion through speech.

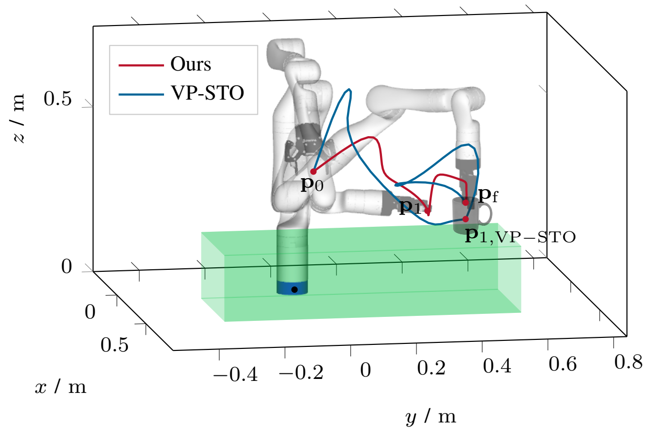

Fig. 2. Experiment 1: The robot is visualized during the grabbing of the mug at the initial pose $\mathbf{p}_0$, the replanning pose $\mathbf{p}_1$, and the final pose $\mathbf{p}_\mathrm{f}$. The black mug rests on a green box in front of the robot.

These and more experiments are demonstrated in the accompanying video at `www.acin.tuwien.ac.at/42d5`.

Natural language from the human operator is the only input for the motion generation in all experiments. The position of the objects, as well as the grasp poses, are assumed to be known to focus on the language processing and motion generation of the framework. The proposed method is compared to the offline trajectory planner VP-STO [28], a state-of-the-art global trajectory planner. The maximum iterations are set to be 1000 to balance optimality and planning time. The reader is referred to [7] and [6] for more comparisons of the used motion planner with state-of-the-art planners. The computational effort for the language parsing is below 10 ms for each parse which makes it real-time capable and more performant than querying an LLM.

A. Experiment 1: Grab a mug with replanning

In the first experiment, the robot is instructed to grab a mug using $\mathcal{L}_0 =$ *"Grab the mug"* at time t_0. This instruction does not contain information about how to grab the mug. There are different grasp possibilities, i.e., grab the handle or the body from the side or from the top. The robot starts to grab the mug from the side, but then the human operator intervenes and adds the instruction $\mathcal{L}_1 =$ *"from the top"* at time t_1. This requires the robot to change its plan during its motion. Specifically, it changes the set $\mathcal{X}_\mathrm{task}$ at t_1. The trajectory of the robot is visualized in Fig. 2. The robot's initial end effector pose is $\mathbf{p}_0$. When hearing the instruction $\mathcal{L}_{t_0}$, it starts moving to grab the mug from the side until the instruction $\mathcal{L}_{t_1}$ triggers a replanning at time t_1, where the end effector is at $\mathbf{p}_1$. The robot then proceeds to the final pose $\mathbf{p}_\mathrm{f}$. The end-effector trajectory of planning with VP-STO is also visualized. As VP-STO is unable to plan

TABLE II

TAXONOMY OF INCREMENTAL CONSTRAINTS. EACH CONSTRAINT CORRESPONDS TO A SPECIFIC ADAPTION IN THE MOTION PLANNING (2).

Type of Constraint $\mathcal{C}(\mathcal{L}_t)$	**Adaptions**	**Example**
Manner Constraints	$\phi(\mathcal{L}_t)$	"Pass the screwdriver... *but go faster.*"
Target Constraints	$\mathcal{X}_{\text{task}}(\mathcal{L}_t)$	"Put the apple in the box... *no, the one on the right.*"
Object Constraints	$\mathcal{X}_{\text{task}}(\mathcal{L}_t)$	"Grab the mug... *no, the blue one.*"
Action Constraints	$\mathcal{X}_{\text{task}}(\mathcal{L}_t)$	"Grab the apple... *no, push it.*"
Safety Constraints	$\mathcal{X}_{\text{safe}}(\mathcal{L}_t)$	"Move the cup... *and keep it upright.*"
Sequential Constraints	$\mathcal{X}_{\text{task}}(\mathcal{L}_t)$	"Pick up the apple... *after you put down the mug.*"

TABLE III

EXPERIMENT 1: COMPARISON OF OUR METHOD WITH VP-STO [28] FOR THE TWO PLANNING INSTANCES DURING THE GRABBING OF THE MUG.

	Ours	**VP-STO**
Planning time t_{plan} / s	0.02	5.60
	0.02	5.62
Trajectory duration t_{traj} / s	3.10	2.89
	4.70	2.97
Total task duration t_{task} / s	7.84	17.08

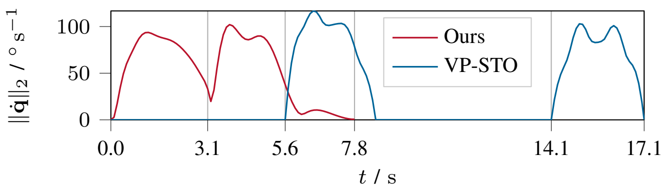

Fig. 3. Experiment 1: Norm of joint velocity $\dot{\mathbf{q}}$ while grabbing the mug.

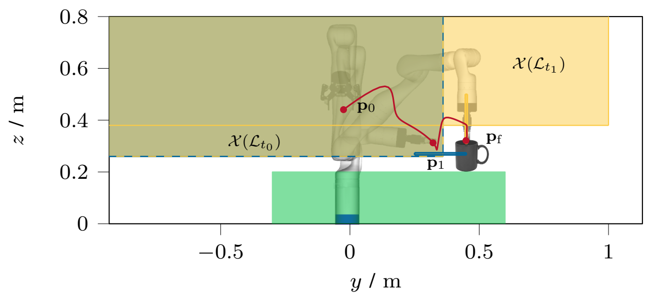

Fig. 4. Experiment 1: Visualization of the sets of permissible states $\mathcal{X}(\mathcal{L}_{t_0})$ in blue and $\mathcal{X}(\mathcal{L}_{t_1})$ in yellow for the mug grabbing in the y-z-plane. The blue sets overlaps with the yellow set in the striped area.

online, see Table III, the robot proceeds to the first grasping point at $\mathbf{p}_{1,\text{VP-STO}}$ and stops. Then, the trajectory from $\mathbf{p}_{1,\text{VP-STO}}$ to $\mathbf{p}_\text{f}$ is planned, and the robot moves to $\mathbf{p}_\text{f}$. With our proposed framework, the robot reacts promptly to the replanning and repositions its end effector according to the new grasping position. The norm of the joint velocity $\dot{\mathbf{q}}$ in Fig. 3 is only zero at the start and end of the motion, but there is no need to stop for the replanning at $t_1 = 3.1\,\text{s}$. In comparison, VP-STO needs to plan for $t_{\text{plan}} = 5.6\,\text{s}$ after receiving the first instruction $\mathcal{L}_{t_0}$ before starting to move and has to remain at $\mathbf{p}_{1,\text{VP-STO}}$ to plan a motion according to $\mathcal{L}_{t_1}$. Due to the longer planning times t_{plan}, VP-STO takes longer to finish the task. The planning times t_{plan} and the trajectory times t_{traj} are summarized in Table III. Our method has faster planning times t_{plan} due to the fast planning times of BoundPlanner [6]. Resolving ambiguity in instructions needs to work online and promptly. The trajectory times t_{traj} are faster for VP-STO as it plans the full trajectory instead of only a finite-horizon approach. However, the overall task time t_{task} is considerably lower with the proposed framework, which justifies using an online trajectory planner. The sets of permissible states $\mathcal{X}(\mathcal{L}_{t_0})$ and $\mathcal{X}(\mathcal{L}_{t_1})$ for the robot's end effector before and after the replanning are shown in Fig. 4. These sets are defined by BoundPlanner and initially constrain the robot to stay to the left of the mug and then grab it. After the replanning, the space above the mug is added to the permissible states $\mathcal{X}(\mathcal{L}_{t_1})$ and the final convex set guides the robot in performing a safe grasping motion from the top.

B. Experiment 2: Transfer a mug with task constraints

In the second experiment, visualized in Fig. 5, the robot is instructed to $\mathcal{L}_{t_0,\text{upright}}$ = *"Pass the mug"* at time t_0. For this, the robot has to pick up a mug and transfer it to a predefined position where the human can receive it. The mug is initially placed on a table next to a laptop. As the mug contains a liquid, spilling the liquid onto the laptop must be avoided, but the robot is not aware of this constraint. As the robot starts moving, the human realizes that the robot will move over the laptop and specifies the instruction at time $t_1 = 6.4\,\text{s}$ by saying $\mathcal{L}_{t_1,\text{upright}}$ = *"but don't spill it"*. In the next run the human specifies $\mathcal{L}_{t_0,\text{avoid}}$ = *"Pass the mug but keep it upright"* at time t_0, the human further specifies $\mathcal{L}_{t_1,\text{avoid}}$ = *"and avoid going over the laptop"* at t_1 to increase safety. Both applications create a task constraint online. This does not change the desired final pose $\mathbf{p}_\text{f}$ but the motion to reach it. Spillage avoidance creates a constraint on the orientation of the robot's end effector, whereas avoiding the laptop adds an obstacle to the path planning.

The utterance $\mathcal{L}_{t_1,\text{upright}}$ avoids spillage on the laptop by keeping the mug upright. The resulting deviations from the upright orientation are described by the two error angles in Fig. 6 for the time after the pickup, i.e., $t \geq 4.5\,\text{s}$. The gray areas indicate the set of permissible deviations from the upright. Initially, the trajectory deviates largely from the upright as no constraint is specified. When the language specification $\mathcal{L}_{t_1,\text{upright}}$ is taken into account at $t_1 = 6.4\,\text{s}$, the upright constraint is added, changing $\mathcal{X}_{\text{task}}$. At this point, the end effector has already deviated from the upright, which makes strictly enforcing $\mathbf{x}(t) \in \mathcal{X}(\mathcal{L}_{t_1,\text{upright}})$ in (2) impossible. Therefore, the upright constraint is enforced with slack variables, effectively bringing the orientation into the desired bounds. Furthermore, the maximum deviation from the upright is at the time of replanning, meaning no further spilling happens afterward. This behavior is safe as the constraint is added before reaching the space over the laptop. Without upright constraints, the deviation from the upright increases further, leading to spillage onto the laptop.

As the upright constraint without consideration of the dynamics is only an approximation for avoiding spillage, the human decides in the second trial to specify $\mathcal{L}_{t_1,\text{avoid}}$ for

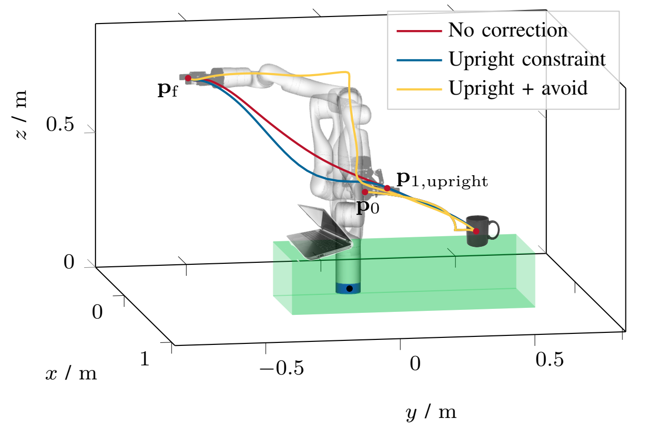

Fig. 5. Experiment 2: The robot is visualized during the transfer of the mug at the initial pose $\mathbf{p}_0$ and the final pose $\mathbf{p}_\mathrm{f}$. The replanning for the upright constraint happens at pose $\mathbf{p}_{1,\mathrm{upright}}$, which is approximately at the same position as for the avoidance $\mathbf{p}_{1,\mathrm{avoid}}$. The black mug and the laptop rest on a green box.

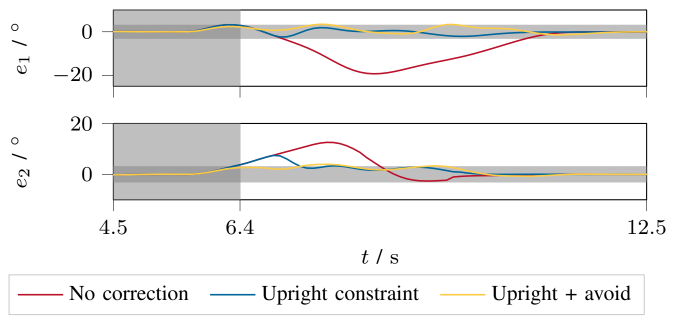

Fig. 6. Experiment 2: Deviations from the upright orientation during the mug transfer after picking up the mug at $t = 4.5$ s. The two error angles e_1 and e_2 correspond to the two RPY-angles that constitute the deviation from the upright, see [7] for the orientation error formulation. The gray areas are the constraints that change during the motion and define the set of permissible states $\mathcal{X}(\mathcal{L}_{t_0})$ that change at time t_1 to $\mathcal{X}(\mathcal{L}_{t_1,\mathrm{upright}})$ for the upright scenario. For the avoidance, the robot is aware of the upright constraint at all times.

additional safety. In this scenario, the robot is aware that the mug should be upright during the entire transfer according to $\mathcal{L}_{t_0,\mathrm{avoid}}$, but the laptop avoidance is added online. The trajectories in the x-y-plane in Fig. 7 show that the motion planner is able to adapt its trajectory online to account for the added task constraint. The replanning happens at pose $\mathbf{p}_{1,\mathrm{avoid}}$ shortly before the robot's end effector reaches the space above the laptop. This requires a fast, safe, and decisive reaction from the robot. The added obstacle is only relevant to the robot's end effector pose but does not affect the position of other parts of the kinematic chain.

C. Experiment 3: Handover of a screwdriver

In a human-robot interaction scenario, the robot needs to be able to adapt to human preferences. This scenario is visualized in Fig. 8 and assumes that a human works on a task for which a screwdriver is needed, but the screwdriver is not accessible to the human due to obstacles. The human instructs the robot with the utterance $\mathcal{L}_{t_0}$ = *"Hand me the screwdriver"* at time t_0 to help with the task. The robot plans to pick up the screwdriver and provide it to the human at a predetermined handover location $\mathbf{p}_\mathrm{f}$. The human expects this task to take a certain time, but the robot moves too slowly to adhere to these expectations. Thus, at time t_1, the human specifies the utterance at time t_1 to $\mathcal{L}_{t_1}$ = *"but move faster"*. This prompts the robot to change the cost function parameters $\phi(\mathcal{L}_{t_i})$ in (2) to value speed more strongly.

Remark. *Human-robot handovers are complicated interactions requiring many factors to be considered. This experiment simplifies this scenario by disregarding, e.g., human motion prediction and psychological factors, for exemplary reasons. A more rigorous approach is presented in [30].*

With and without the speedup instruction $\mathcal{L}_{t_1}$, the robot is able to finish the task and reach $\mathbf{p}_\mathrm{f}$ as seen in Fig. 8. However, the robot takes considerably longer without the speedup, as shown by the norm of the Cartesian end effector velocity $\mathbf{v}$ in Fig. 9. In both cases, the robot avoids obstacle collisions and safely reaches $\mathbf{p}_\mathrm{f}$. Thus, our framework enables online modifications of the objective to account for human preferences.

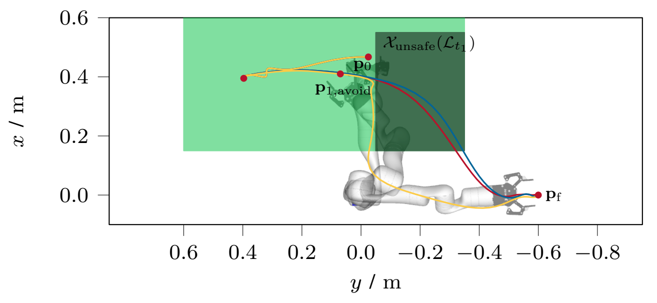

Fig. 7. Experiment 2: Trajectories for the laptop avoidance during the mug transfer in the x-y-plane. The set of disallowed states $\mathcal{X}_\mathrm{unsafe}(\mathcal{L}_{t_1})$ is shaded and constitutes the space above the laptop.

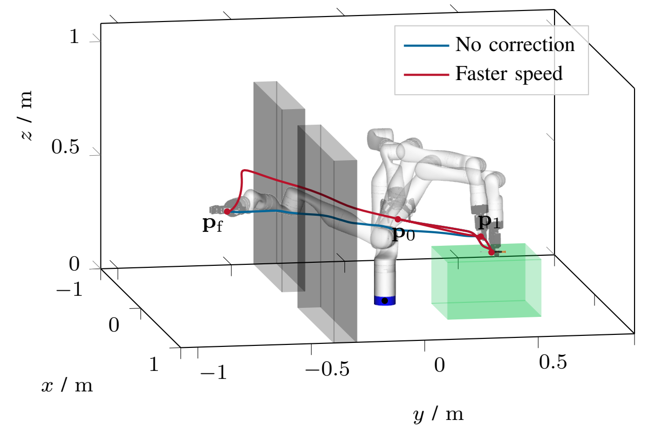

Fig. 8. Experiment 3: The robot is visualized during the handover of the screwdriver at the initial pose $\mathbf{p}_0$, the pickup pose, and the handover pose $\mathbf{p}_\mathrm{f}$. The screwdriver rests on the green table and the robot has to avoid the black walls while transitioning to the handover location $\mathbf{p}_\mathrm{f}$.

VI. CONCLUSION

This paper presents an incremental language processing system integrated with an online motion planner for real-time human-robot interaction. Our approach enables a robotic agent to dynamically interpret linguistic instructions,

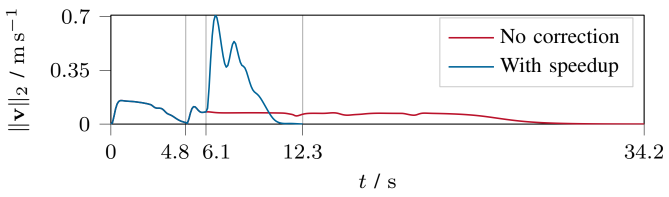

Fig. 9. Experiment 3: Norm of the end-effector velocity $\mathbf{v}$ during the screwdriver handover with and without the speedup. The speedup is instructed at $t_1 = 6.1\,\mathrm{s}$ using $\mathcal{L}_{t_1}$.

continuously update motion plans, and refine its actions based on evolving constraints, unlike previous methods that assume fully specified language input. The system adapts to speech corrections and new task constraints in real time, such as modifying grasp strategies or avoiding obstacles, reflecting more naturalistic communication and interactions. The effectiveness of our approach was demonstrated in the real world using a 7-DoF robot manipulator. Future work will focus on extending the system to incorporate visual information, refining adaptation mechanisms to support more complex task hierarchies, and conducting a thorough human subject evaluation study.

References

[1] T. Oelerich, C. Hartl-Nesic, and A. Kugi, "Language-guided Manipulator Motion Planning with Bounded Task Space," in *Proc. Conference on Robot Learning*, 2024.

[2] K. Lin, C. Agia, T. Migimatsu, M. Pavone, and J. Bohg, "Text2Motion: From Natural Language Instructions to Feasible Plans," *Autonomous Robots*, vol. 47, no. 8, pp. 1345–1365, 2023.

[3] W. Huang, C. Wang, R. Zhang, Y. Li, J. Wu, and L. Fei-Fei, "Voxposer: Composable 3d value maps for robotic manipulation with language models," in *Proc. Conference on Robot Learning*, 2023, pp. 540–562.

[4] C. Lynch, A. Wahid, J. Tompson, T. Ding, J. Betker, R. Baruch, T. Armstrong, and P. Florence, "Interactive language: Talking to robots in real time," *IEEE Robotics and Automation Letters*, pp. 1–8, 2023.

[5] Y. Cui, S. Karamcheti, R. Palleti, N. Shivakumar, P. Liang, and D. Sadigh, ""No, to the Right" – Online Language Corrections for Robotic Manipulation via Shared Autonomy," in *Proc. Conference on Human-Robot Interaction*, 2023, pp. 93–101.

[6] T. Oelerich, C. Hartl-Nesic, F. Beck, and A. Kugi, "BoundPlanner: A convex-set-based approach to bounded manipulator trajectory planning," *Preprint arXiv:2502.13286*, 2025.

[7] T. Oelerich, F. Beck, C. Hartl-Nesic, and A. Kugi, "BoundMPC: Cartesian path following with error bounds based on model predictive control in the joint space," *The International Journal of Robotics Research*, p. 02783649241309354, 2025.

[8] M. K. Tanenhaus and J. C. Trueswell, "Sentence comprehension." in *Speech, Language, and Communication.*, ser. Handbook of Perception and Cognition (2nd Ed.), Vol. 11. Academic Press, 1995, pp. 217–262.

[9] W. J. Levelt, "Speaking: From intention to articulation," *MITPress, Cambridge, Mass.*, 1989.

[10] A. Peldszus and D. Schlangen, "Incremental construction of robust but deep semantic representations for use in responsive dialogue systems," in *Proc. Workshop on Advances in Discourse Analysis and its Computational Aspects*, 2012, pp. 59–76.

[11] S. Constantin, J. Niehues, and A. Waibel, "Incremental processing of noisy user utterances in the spoken language understanding task," *Preprint arXiv:1909.13790*, 2019.

[12] D. DeVault and M. Stone, "Domain inference in incremental interpretation," in *Proc. ICoS*, 2003, pp. 73–87.

[13] C. Kennington, D. Moro, L. Marchand, J. Carns, and D. McNeill, "rrsds: Towards a robot-ready spoken dialogue system," in *Proceedings of the 21th annual meeting of the special interest group on discourse and dialogue*, 2020, pp. 132–135.

[14] T. Baumann and D. Schlangen, "The inprotk 2012 release," in *NAACL-HLT Workshop on Future directions and needs in the Spoken Dialog Community: Tools and Data (SDCTD 2012)*, 2012, pp. 29–32.

[15] T. Michael and S. Möller, "Retico: An open-source framework for modeling real-time conversations in spoken dialogue systems," *Studientexte zur Sprachkommunikation: Elektronische Sprachsignalverarbeitung 2019*, pp. 134–140, 2019.

[16] M. Abrams, C. Bonial, and L. Donatelli, "Graph-to-graph meaning representation transformations for human-robot dialogue," in *Proceedings of the Society for Computation in Linguistics 2020*, 2020, pp. 250–253.

[17] J. Nivre, "Dependency parsing," *Language and Linguistics Compass*, vol. 4, no. 3, pp. 138–152, 2010.

[18] K. Lin, C. Agia, T. Migimatsu, M. Pavone, and J. Bohg, "Text2motion: From natural language instructions to feasible plans," *Autonomous Robots*, vol. 47, no. 8, pp. 1345–1365, 2023.

[19] M. K. Tanenhaus and S. Brown-Schmidt, "Language processing in the natural world," *Philosophical Transactions of the Royal Society B: Biological Sciences*, vol. 363, no. 1493, pp. 1105–1122, 2008.

[20] M. J. Steedman, "Grammar, interpretation, and processing from the lexicon," in *Lexical representation and process*, 1989, pp. 463–504.

[21] T. Berg and W. J. M. Levelt, "Speaking: From intention to articulation," *American Journal of Psychology*, vol. 103, p. 409, 1990.

[22] G. Skantze and D. Schlangen, "Incremental dialogue processing in a micro-domain," in *Proc. Conference of the European Chapter of the Association for Computational Linguistics*, 2009, pp. 745–753.

[23] W. Schuler, S. Wu, and L. Schwartz, "A framework for fast incremental interpretation during speech decoding," *Computational Linguistics*, vol. 35, no. 3, pp. 313–343, 2009.

[24] M. Poesio and H. Rieser, "An incremental model of anaphora and reference resolution based on resource situations." *Dialogue Discourse*, vol. 2, pp. 235–277, 2011.

[25] C. Kennington and D. Schlangen, "A simple generative model of incremental reference resolution for situated dialogue," *Computer Speech & Language*, vol. 41, pp. 43–67, 2017.

[26] T. Brick and M. Scheutz, "Incremental natural language processing for hri," in *Proc. Conference on Human-Robot Interaction*, 2007, pp. 263–270.

[27] T. Marcucci, M. Petersen, D. von Wrangel, and R. Tedrake, "Motion planning around obstacles with convex optimization," *Science Robotics*, vol. 8, no. 84, p. eadf7843, 2023.

[28] J. Jankowski, L. Brudermüller, N. Hawes, and S. Calinon, "VP-STO: Via-point-based Stochastic Trajectory Optimization for Reactive Robot Behavior," in *Proc. International Conference on Robotics and Automation*, 2023, pp. 10 125–10 131.

[29] F. Beck, M. N. Vu, C. Hartl-Nesic, and A. Kugi, "Model Predictive Trajectory Optimization With Dynamically Changing Waypoints for Serial Manipulators," *IEEE Robotics and Automation Letters*, vol. 9, no. 7, pp. 6488–6495, 2024.

[30] T. Oelerich, C. Hartl-Nesic, and A. Kugi, "Model Predictive Trajectory Planning for Human-Robot Handovers ," in *Proceedings of the VDI Mechatronics Conference*, 2024, pp. 65–72.

[31] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "STOMP: Stochastic trajectory optimization for motion planning," in *Proc. International Conference on Robotics and Automation*, 2011, pp. 4569–4574.

[32] P. S. Schmitt, F. Wirnshofer, K. M. Wurm, G. V. Wichert, and W. Burgard, "Planning Reactive Manipulation in Dynamic Environments," *Proc. International Conference on Intelligent Robots and Systems*, pp. 136–143, 2019.

[33] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[34] Y. Li, C. Zheng, K. Chen, Y. Xie, X. Tang, M. Y. Wang, and J. Ma, "Collision-Free Trajectory Optimization in Cluttered Environments Using Sums-of-Squares Programming," *IEEE Robotics and Automation Letters*, vol. 9, no. 12, pp. 11 026–11 033, 2024.

[35] D. Schlangen and G. Skantze, "A general, abstract model of incremental dialogue processing," *Dialogue & Discourse*, vol. 2, no. 1, pp. 83–111, 2011.

[36] D. H. Younger, "Recognition and parsing of context-free languages in time n3," *Information and Control*, vol. 10, no. 2, pp. 189–208, 1967.

[37] T. Schoels, L. Palmieri, K. O. Arras, and M. Diehl, "An NMPC Approach using Convex Inner Approximations for Online Motion

Planning with Guaranteed Collision Avoidance," in *Proc. International Conference on Robotics and Automation*, 2020, pp. 3574–3580.