RC25632 (WAT1610-084) October 25, 2016
Computer Science

IBM Research Report

Teaching and Checking of Constraints for Surgical Tray Layout

Lanbo She
Michigan State University
East Lansing, MI 48824 USA

Jonathan Connell
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598 USA

Research Division
Almaden – Austin – Beijing – Brazil – Cambridge – Dublin – Haifa – India – Kenya – Melbourne – T.J. Watson – Tokyo – Zurich

Teaching and Checking of Constraints for Surgical Tray Layout

Lanbo She
Department of Computer Science and Engineering
Michigan State University
East Lansing, Michigan 48824, USA
shelanbo@cse.msu.edu

Jonathan Connell
IBM T.J. Watson Research
Yorktown Heights, NY 10598, USA
jconnell@us.ibm.com

Abstract

We introduce an automatic agent that, through linguistic communication with a human, can check constraints for object layouts as well as learn new constraints. This is applied to a surgical instrument preparation task, where the system interacts with an expert to learn both general instrument arrangement regulations and special layouts customized to a particular doctor. When supervising a non-expert human preparing the tools for a surgery, the system can provide suggestions by giving verbal feedback about how to improve the setup while simultaneously highlighting corresponding objects through a graphical interface.

Introduction

In recent years the application of artificial intelligence technologies to the field of medicine has been actively studied. Example AI systems have been successfully applied for disease diagnosis, to help reduce errors related to human fatigue, to generally improved health care, and for educational purposes in medical schools. In this paper we propose a cognitive agent that works in operating rooms, specifically an agent that can help with the surgery preparation process. It does this by giving advice as a surgical technician prepares instruments on a tray.

Usually, before a surgery, an assistant or nurse will set out the instruments used in a surgery in the order in which they are commonly used. A well organized setup can expedite the exchange of instruments between the assistant and the doctor, and obviate searching the tray for an out-of-place instrument. This saves time, which can be critical during a procedure. On the other hand, a disorganized instrument layout can have a significant adverse effect on the outcome of an operation.

An example of a somewhat disorganized layout is shown in Figure 1. Note that this image is sparser than a real surgical tray, but is fairly representative of a dental tray. In this example, a large pair of scissors v_0 is misplaced between two smaller pairs of scissors (v_2 and v_4). As a result, when the assistant casually tries to grab a small pair of scissors, he might accidentally grab a large pair instead.

Becoming a well-trained technician requires memorizing the necessary instruments for different surgeries, as well as

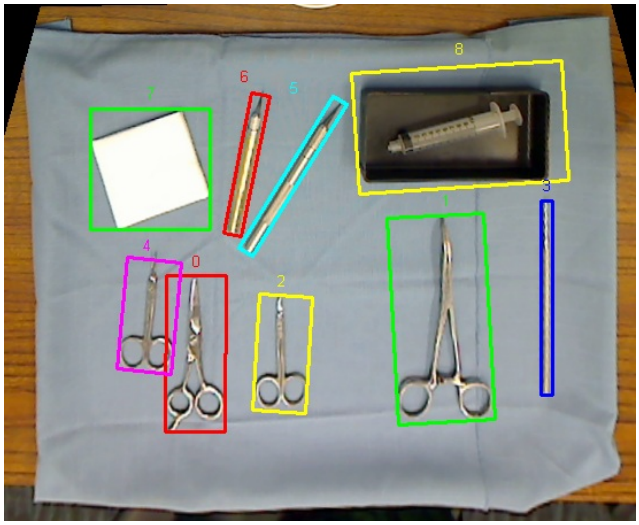

Figure 1: Rectified image of surgical instrument arrangement.

extensive training in how to set up a tray properly. In addition, sometimes particular doctors may have preferences for specific tools or the order of different tools. However, even well-trained technicians may make mistakes due to fatigue caused by the large number of operations in a day. Moreover, in some cases well-trained assistants may not be available (e.g., emergency situations, battle fields) and lesser trained personnel attempt to fill in.

Taking these concerns into consideration, we propose a cognitive agent that can learn how to properly arrange a tray as a set of rules acquired through spoken interaction with one or more human experts. Having been taught the rules, it can then supervise a non-expert human while they prepare surgical instruments, like a "super spell checker".

Specifically, images of instruments on a tray are sensed by an overhead-mounted camera, further processed by computer vision modules, and finally represented as a set of logical predicates. This discretized state is checked against a set of setup rules, essentially a collection of first order logic formulae. Those rules violated by current state are reported to the human partner (e.g., non-expert assistant) in the form of a natural language suggestion augmented by highlighting the

corresponding objects on the screen. To train or customize the system, setup rules are taught using linguistic input from human experts (e.g., well trained surgical technicians). The experts' spoken descriptions for layout rules are translated to formal representations (i.e., FOL formula) by detecting specific language patterns, and then stored for later use.

Related Work

An operating room is an example of environment which is high paced, stressful, and full of uncertainty. To ease the burden on surgeons and their assistants, researchers from HCI and the medical field have tried to build cognitive agents to aid surgical procedures. For example, the authors in (Seagull FJ 2008 Aug) have identified four pillars for an "operating room of the future" where innovative technologies can contribute to patient safety in the operating environment. Similarly, (Ellner and Joyner 2012) describes methods for applying information technology to improve safety and quality for surgical patients. The use of paper checklist in operating room is studied in detail in (Zhang et al. 2014) and used to inform the design of a cognitive agent (e.g., digital checklist) to improve the interaction and communication between surgery team members. This is taken further by the authors in (Wu et al. 2013; 2011) who have created a dynamic checklist using both a tablet and a large display. However, in contrast to this previous work focused on displaying information, our agent acts like a supervisor. It aims to prevent human errors and give advice to non-experts under emergency situation when experts are in short supply.

Another closely related research topic is knowledge acquisition from interactions. Although surgical instrument layout follows certain general rules, the instruments and their arrangement varies somewhat based on the class of surgery, and is even customized to specific surgeons on occasion. These additional constraints need to be induced on top of "factory settings" in a natural way. One actively studied human-computer interaction area in recent years has been Learning from Demonstration (LfD) (Akgun et al. 2012; Argall et al. 2009; Atkeson and Schaal 1997; Bentivegna, Atkeson, and Cheng 2004; Cakmak and Thomaz 2012). Here the agents can acquire knowledge through human illustrations (e.g., steps to perform actions, examples of previous unknown objects). Some of these systems focus on action knowledge acquisition (Matuszek et al. 2013; Misra et al. 2015; She and Chai 2016; She et al. 2014), and some of them focus on object and sequence naming (Chai et al. 2014; Connell 2014). However, none address the acquisition of proper spatial relations between objects or other configurational constraint rules.

System Architecture

A brief system work flow is shown in Figure 2. Next we will explain the key system components in detail.

Vision Processing

The real-time image of the tray is captured by an overhead camera and processed by a set of vision modules. The image is first geometrically rectified to make the tray rectangular. Next, pixels in the raw image are separated into background or objects by forming a color model for the majority of the area (i.e., surgical instruments are usually placed on a cloth with single color that contributes most of the pixels in the image). The pixels belonging to objects are grouped by connected components to form potential objects (Connell et al. 2012; Connell 2014). After removing small items and ones protruding beyond the tray boundary, the orientation and aligned bounding box are determined for each remaining object. These are plotted in Figure 1.

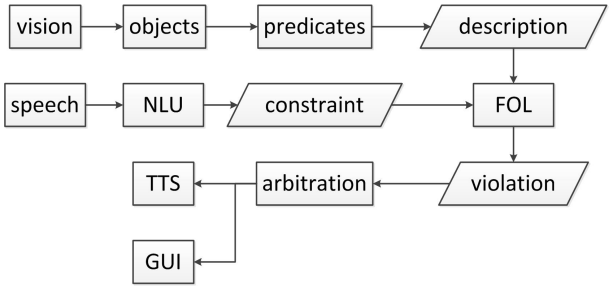

Figure 2: The system receives input via speech and vision.

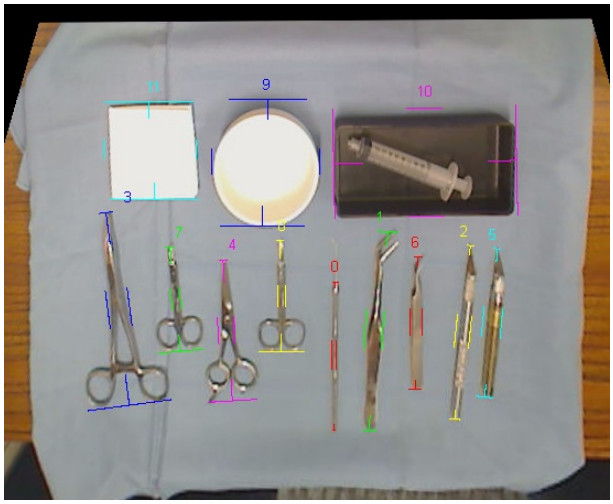

Figure 3: Each object is analyzed to yield a collection of three widths.

In addition, the rough shape of the object is captured by calculating the width of the tip, center, and bottom of each object separately. This, along with the position of each subpart, can be seen in Figure 3. Using these measurements, as well as the overall bounding box, a heuristic classifier is applied to ascertain the identity of each potential object. Each of the known surgical instruments is represented by a simple vector model of the its parameters, so rule-based matching is fairly straightforward.

Given a set of recognized objects, an environment description is built to capture information important to the domain. For the surgical instrument arrangement application, the important information includes state terms describ-

ing either 1) a single object (e.g., object type: *scissors*(v_0), *forcep*(v_0)), or 2) relations between objects (e.g., spatial relation: *parallel*(v_0, v_1) meaning v_0 is parallel with v_1, *centeralign*(v_0, v_2) meaning v_0 and v_2 are center aligned), or 3) relations between an object and the tray (e.g., object location on the tray: *at_part*(v_0, L) meaning v_0 is at the left part of the tray). Each such state predicate is formed by instantiating a pre-defined state atom using one or more real-time detected objects.

The set of current pre-defined state atoms is listed in Figure 4. Each state atom is associated with a function that can tell, when the atom is instantiated with certain objects, whether it is true or not relative to the perceived environment. For example, the predicate "*parallel(x,y)*" is associated with function $(x_{orient} - y_{orient}) < \delta ? True : False$, which is true if the orientations of x and y are similar, and false otherwise. For the scene in Figure 1 "$parallel(v_0, v_2)$" is true while "$parallel(v_5, v_6)$" is false.

Unary predicate:
TYPE(x), SHAPE(x)
Binary relation relative to the tray:
at_corner(*x, corner*), at_part(*x, part*), parallel(*x, border*), perpen(*x, border*)
Binary relation between objects:
bottomalign(*x,y*), centeralign(*x,y*), topalign(*x,y*), parallel(*x,y*), perpen(*x,y*), leftof(*x,y*), rightof(*x,y*), above(*x,y*), below(*x,y*), keepdistance(*x,y*), closeto(*x,y*)

$TYPE \in \{scissor, scalpel, gauzepad \ldots\}$	$SHAPE \in \{square, circle \ldots\}$
$x, y \in O$ objects on the tray	$corner \in \{bl, br, tl, tr\}$
$border \in \{left, right, top, bottom\}$	$part \in \{l, r, b, t\}$

Figure 4: The set of state atoms that are used both in the discrete state representation and the First-Order Logic constraints.

Some predicates have more complicated geometric interpretations. For instance, the "*keepdistance*" relation says whether two objects have been placed too close together. As show in Figure 5 this is based on the standard width $d1$ of a human finger. Objects are too close when the user cannot insert his fingers to grasp one object without hitting the other one. This predicate depends on the dimensions of the two objects, their separation, and their relative orientation.

The final result of visual processing is a discretized symbolic representation embodied by a set of logical state predicates. That is, the complete environment description is the conjunction of all the state terms that are currently true. By using such a conjunction of state terms to represent a scene, the system is able to check the environment against the collection of setup constraints and thereby identify the constraints that are violated.

Language Processing

Human voice is captured through a microphone and then the Microsoft ASR Engine under Windows 7 is used to perform speech recognition. To improve accuracy, we constrain the acoustic utterances with a defined language grammar. This is composed of a set of CFG-style rules, each specifying

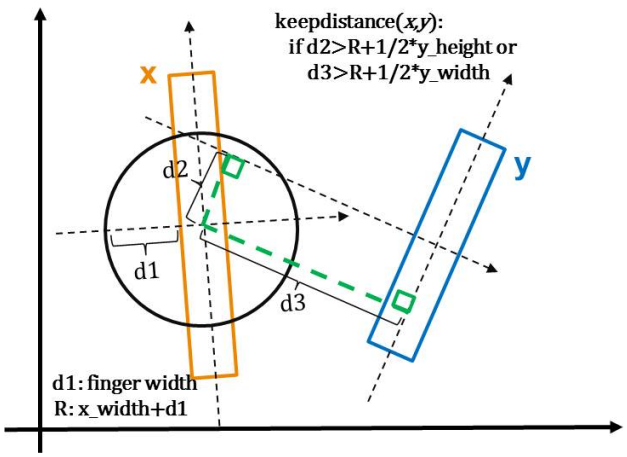

Figure 5: Many semantic predicates depend on the scene geometry.

how syntax elements are expanded. An example grammar is shown in Figure 6. The first rule in the example means "*this is a type of sentence that describes an object*" and the second rule means "*the DescribeObject sentence is composed of drawing the agent's attention (Attn) followed by propounding an object description (obj)*". Another benefit of utilizing a language grammar is that each recognized utterance can be parsed by any chart parser (e.g., Earley parser, CYK parsing) to generate a tree, which can have embedded task-related information. More details about a similar language grammar and parse tree analysis can be found in (Connell 2014).

=[S] <Intent-DescribeObject>	=[Def] <DET> | <QUANT>
=[Intent-DescribeObject] <Attn> <obj>	=[obj] <Def><SHAPE><TYPE>(<PPMod>)
=[Attn] "robot" | "Tracy"	=[TYPE] "forcep" | "scalpel" | "gauze pad" | "hemostat"
=[DET] "a" | "an" | "the" ...	...

Figure 6: Example language grammar covering both intent and syntax.

But pure parsing is not enough. All utterances are further processed by a series of language understanding modules to extract sentence meaning. This includes several different aspects such as intention, the main objects being talked about, and setup constraints. Some example sentences and their resulting outputs are shown in Figure 7.

In particular, an intention recognizer is used to capture the purpose behind an utterance. That is, for example, whether the human is describing an object (i.e., DescribeObject), introducing a constraint rule (i.e., IntroConstraint), or just wants to say hello (i.e., Greeting). Capturing human intent helps the agent decide what response should it give (e.g.,

H1: ***Tracy, the gauze pad.***
Intent: DescribeObject **Focus**: v7 **NP**: gauze pad
Grounding: gauze-pad=>v7

H2: ***Tracy, it should be placed at the top left corner of the tray.***
Intent: IntroConstraint **Focus**: v7 **NP**: it, top left corner
Grounding: it=>v7, top left corner=>tl
Semantic: all x. (gauzepad(x) -> at_corner(x, tl))

H3: ***Tracy, what do you think about this tray?***
Intent: StartTest **Focus**: v7

Figure 7: Some example sentences and their derived meanings.

when the human is describing an object it should change its attention to that object, when human says hello it should return a greeting). Due to our narrow domain, currently we just define a fixed set of intentions which are embedded in the language grammar. Thus the intent can be conveniently recovered from the top label of the parsing tree.

Besides recognizing human intent, the agent also needs to understand the object descriptions in a human utterance (i.e., Object Grounding). Specifically, the noun phrases (in our case, phrases describing specific instruments or the tray) need to be mapped to physical objects detected in the scene. For example, the "gauze pad" should be grounded to v_7 in Figure 1, and "the leftmost scissors" should be grounded to v_4. To achieve this, we define grounding functions associated with specific words that assign a score telling how well a particular object matches with this word. For example, the word "scissors" matches better with objects v_0, v_2, and v_4 in Figure 1 than with the other objects. And the word "leftmost" assigns higher scores for objects that are farther to the left. The score for a noun phrase (e.g., "leftmost scissors") is then calculated by adding the individual scores for the descriptive terms. The highest scoring object is then assumed to be the referent for the noun phrase.

Our system also keeps track of another piece of information important for dialog: the Focus Object. By Focus Objects we mean the objects that are the center of attention in the current dialog turn. For instance, in the phrase "the dental pick to the right of the hemostat", the "dental pick" is the focus object, while the "hemostat" is a landmark used to identify that focus object. Keeping track of the focus object is useful for reference resolution. For example, when the human says "they should be grouped together", the human is trying to introduce a constraint (i.e., some objects should be placed near each other). However, to construct a constraint rule from this sentence, the objects that "they" refers to must be correctly identified. This turns out to be the focus objects of the most recent dialog turn. In our case, the focus object is either inherited from previous turn, or switched to a new object if the human decides to describe a different instrument.

For utterances trying to introduce a setup constraint, like "the scissors should be grouped together", we will also generate a first order logic formula ("Semantic" in the example). The problem of translating such natural language statements to a logic representation has been actively studied by the NLP community. Typical methods include learning this mapping from a large set of parallel data, where each language sentence is paired with a desired logic form (Kwiatkowski et al. 2011; Zettlemoyer and Collins 2009). Yet, given our narrow domain, we currently just look for specific language patterns that are commonly used by people when introducing constraints. We associate a FOL template with each of these. In this way, when the pattern is detected in a sentence, its full semantic meaning can be constructed by simply filling the template with content extracted from elsewhere in the parsing tree. This is discussed in more detail below.

Constraints and Reasoning

In our system, knowledge of how to arrange the surgical instruments is formulated as a set of *constraint*s that the object layout should satisfy. Example constraints for a specific surgery include the number of each type of instrument needed, whether instruments of the same type should be close to each other or in some use-based order, and the canonical locations that different tools should be placed on the tray (e.g., bowls with liquids should situated far from areas that have frequent interactions with the surgical assistant).

The representation for a *constraint* should 1) be able to capture the semantics of the constraint, and 2) support validity checking (i.e., whether the constraint holds in the layout) as well as violation identification (i.e., be able to determine which objects are responsible). Here we utilize *First-Order-Logic* (*FOL*) sentences to represent constraints. Example FOL sentences and their semantics in the tray environment are shown in Figure 8. Each FOL constraint is composed of a number of terms connected by logic symbols (e.g., & and , $->$ infer, $-$ negation). Variables (e.g., *x*, *y*) in the terms are bounded by quantifiers (e.g., *all*, *exist*) and can be instantiated with objects recognized from the scene. The terms used in constraints share the same set of state descriptors as was shown in Figure 4.

To perform reasoning, we use the NLTK package (Loper and Bird 2002) and theorem prover Prover9 (McCune 2005 2010). Given an environment description and a FOL constraint, the prover can tell whether each constraint is valid in the environment (i.e., the formula is true under every interpretation of variable assignments). Such a prover can check the validity of a constraint but, if a FOL formula is shown to be invalid in an environment, we cannot immediately tell which object(s) triggered this violation. To remedy this, we negate the constraint in question and find the interpretations of *this* that are satisfied in the environment. The objects involved in the bindings from these interpretations are then deemed the culprits. We use these objects, together with the violated constraint, to generate feedback to the human.

Dialog Management

The task for Dialog Management is to decide what actions the agent should take in different dialog states. The design of our dialog manager follows (Chai et al. 2014) and consists of three components: state, policy, and actions. The dialog

FOL sentences	Semantics
all x.(scissors(x) -> exist y. (scissors(y) & closeto(x,y) & -(x=y)))	scissors should be grouped together.
all x. all y. (instrument(x) & instrument(y) & -(x=y) -> keepdistance(x,y))	any two instruments should not be too close to each other.
exist x. exist y. (scalpel(x) & scalpel(y) & -(x=y))	Two scalpels are necessary in this surgery.
all x. all y. (scalpel(x) & scalpel(y) & -(x=y) -> parallel(x,y))	Scalpels should be placed parallel with each other.

Figure 8: Example FOL constraints and their semantic interpretation.

state is characterized by the parsing result, human intention, object grounding, and focus identification. The dialog policy consists of 12 rules, where each is a state-action pair. During interaction, the agent will identify the current dialog state, match it against the policy to find the best rule, and then retrieve actions to execute from the right hand side of the rule. Our current space of agent actions includes verbal feedback moves like greeting, making a confirmation, describing how to improve the layout, and explaining problems with its language understanding (e.g., cannot find focus object, failed to induce a FOL constraint, or totally don't understand). In addition, actions can be system internal operations like checking the current environment against the FOL constraints, or updating the set of known constraints.

Response Generation

Once the dialog manager decides what action(s) to take, Response Generation is responsible for executing them. If the layout is good enough, the agent will just tell the human that things are satisfactory. If violations are detected, instead, our agent will create a natural language description of its concern along with a visual indication of the problematic objects. Note that, if there are multiple violations, it is generally a bad idea to inundate the user with all the problems at once. To help structure the interaction more cleanly, we just randomly choose one violation out of the list (but indicate that there are more). Perhaps a better solution would be to employ an arbitration scheme to impose an order on these. For instance, fixing a "*parallel*" violation often serendipitously fixes a related "*keepdistance*" violation. Thus it would make sense to point out the "*parallel*" violation first.

Once a problem has been identified, a linguistic description of this violation is generated through a two step process. First, because the agent needs to help the user identify which objects to fix, a referring expression is generated for each violated object. Referring expression generation is itself an active research topic in NLP (Fang et al. 2013; Fang, Doering, and Chai 2014; Krahmer and van Deemter 2012). We utilize a strategy similar to (Haas 1992), where information about object type, absolute locations (i.e., left, right, top, bottom of the tray), and relative locations within a group of the same type of objects is used to form a unique description for each object. Thus two pairs of scissors might be denoted by "the rightmost scissors at the bottom left" and "the leftmost scissors". Second, several language templates are predefined for each type of violation. At run time, the agent randomly chooses one template and fills it with the referring expressions just generated. Considering two pairs of scissors, a violation might finally be expressed as "the rightmost scissors at the bottom left is too close to the leftmost scissors".

Finally, when an associated a template has been filled, the complete description is sent to a speech synthesizer to provide verbal feedback to the human. In addition, the specific objects that trigger a violation are highlighted on the screen. For example, looking ahead to the second image of Figure 10 the two pairs of scissors are too close to each other, which is a violation of the "*keep distance*" rule. Therefore the agent highlights these two objects on the screen (green and magenta) to draw the user's attention to the correct area.

Performance of Implemented System

A video demonstration our Tracy system can be viewed at (She and Connell 2016). Figure 10 shows a scenario from this video where the agent assists a non-expert to set up the surgical tray through dialog. When asked for advice, the agent analyzes the current tray setup with an overhead camera to generate a state representation such as in Figure 9. It then compares this against a set of layout rules to generate feedback for the user. For example, as shown in left panel of Figure 10, the scalpels and scissors are not well organized. The agent first complains that the two scalpels are not placed parallel, and labels them on the screen (green and magenta). After this problem is fixed by the human, the agent goes on to describe the second problem: that the two pairs of scissors are too close to each other. Once all the problem are fixed (i.e., the environment satisfies every constraint in agent's knowledgebase), the agent reports that the setup is fine.

```
scissors(v0), scissors(v2), forcep(v1)
...
scalpel(v5), scalpel(v6),
...
at_part(v0, L), at_part(v0, B),
parallel(v0, v1), centeralign(v0, v2),
keepdistance(v0, v2),
...
```

Figure 9: Example state representation for Figure 10.

In Figure 11 an expert (e.g., surgical assistant) teaches the agent more setup factors through language interaction. In this case, the particular operation requires more pairs of

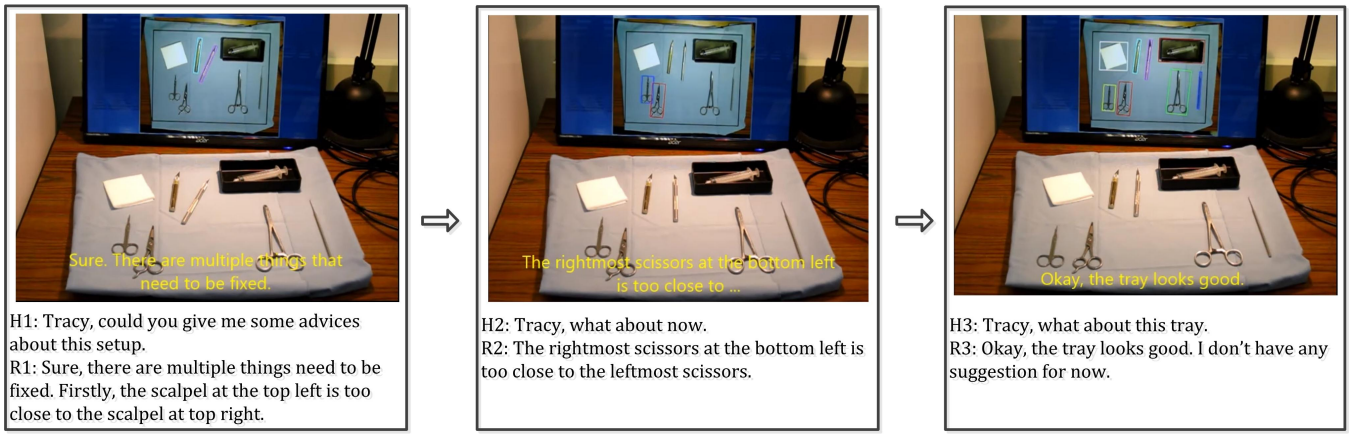

Figure 10: A constraint checking situation from the demo video.

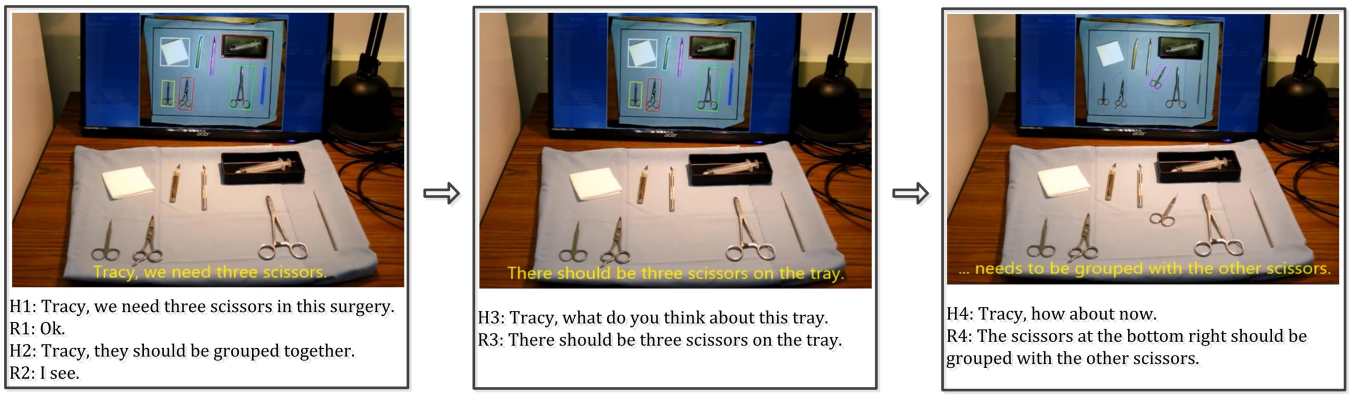

Figure 11: Teaching the system new constraints (from the demo video).

scissors than normal. In addition, for ease of access, the scissors should all be in the same portion of the tray. The human imparts this knowledge to the system by conversing with the agent, as shown in the left panel. In the middle panel, Tracy automatically detects the lack of a vital instrument. Then, when an additional pair of scissors is added (right panel), it complains that it was placed inappropriately. This proves it was able to understand and apply the added constraints it was taught.

Conclusion

In this work we have integrated many disparate AI technologies into a working real-time system, and there is a publicly viewable demonstration video. The system visually parses an instrument layout and develops an internal symbolic representation of the situation. Using this it is able to critique an instrument layout and provide corrective suggestions using speech and graphics. It is also able to accept and apply brand new constraints and surgery-dependent requirements. Imparting new rules simply by speaking like this gives non-technical users the invaluable ability to refine and extend the system in the field, something usually not possible for non-programmers.

Of course there are many ways the system could be extended. For instance, the bulk of the grounding is currently hand-coded. We are in the process of implementing a hand tracker that can determine pointing locations as well as general regions indicated by circling motions. Pointing would allow us to build new object models by indicating a name for some object, e.g. "this is a forceps". We could also use it for deictic corrections such as "this scalpel should be closer to the other one" to help tune constraints. Finally, region indication could be useful for designating (possibly unnamed) "home" locations for objects, e.g. "the gauze pads go over here".

The system could also be ported to perform a similar tutorial/checking function in other domains. The most obvious is how to properly "plate" food at a high end restaurant. Relevant instructions might be that the broccoli spears should be aligned and on the left. Similarly, the split radishes should adorn the upper margin of the plate, while the boiled artichoke is centered. Some of these specialized terms themselves might also be grounded through interactive teaching.

In general, our agent is one example of a more general class of problems where a Cognitive System watches over the shoulder of a human to help ensure compliance with safety standards and promote industrial best practices. There will likely be many more such systems in the future.

References

Akgun, B.; Cakmak, M.; Jiang, K.; and Thomaz, A. L. 2012. Keyframe-based learning from demonstration - method and evaluation. *I. J. Social Robotics* 4(4):343–355.

Argall, B. D.; Chernova, S.; Veloso, M.; and Browning, B. 2009. A survey of robot learning from demonstration. *Robot. Auton. Syst.* 57(5):469–483.

Atkeson, C. G., and Schaal, S. 1997. Robot learning from demonstration. In *Proceedings of the Fourteenth International Conference on Machine Learning*, ICML '97, 12–20. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Bentivegna, D. C.; Atkeson, C. G.; and Cheng, G. 2004. Learning tasks from observation and practice. *Robotics and Autonomous Systems* 47(23):163 – 169. Robot Learning from Demonstration.

Cakmak, M., and Thomaz, A. L. 2012. Designing robot learners that ask good questions. In *Proceedings of the Seventh Annual International Conference on Human-Robot Interaction*, HRI '12, 17–24. New York, NY, USA: ACM.

Chai, J. Y.; She, L.; Fang, R.; Ottarson, S.; Littley, C.; Liu, C.; and Hanson, K. 2014. Collaborative effort towards common ground in situated human-robot dialogue. In *Proceedings of the 2014 International Conference on Human-robot Interaction*, HRI '14, 33–40. New York, NY, USA: ACM.

Connell, J.; Marcheret, E.; Pankanti, S.; Kudoh, M.; and Nishiyama, R. 2012. *An Extensible Language Interface for Robot Manipulation*. Berlin, Heidelberg: Springer Berlin Heidelberg. 21–30.

Connell, J. H. 2014. *Extensible grounding of speech for robot instruction*.

Ellner, S. J., and Joyner, P. W. 2012. Information technologies and patient safety. *Surgical Clinics of North America* 92(1):79 – 87. Patient Safety.

Fang, R.; Liu, C.; She, L.; and Chai, J. Y. 2013. Towards situated dialogue: Revisiting referring expression generation. In *EMNLP*, 392–402. ACL.

Fang, R.; Doering, M.; and Chai, J. Y. 2014. Collaborative models for referring expression generation in situated dialogue. In *AAAI*, 1544–1550.

Haas, N. 1992. Learning by ostentation for robotic assembly.

Krahmer, E., and van Deemter, K. 2012. Computational generation of referring expressions: A survey. *Comput. Linguist.* 38(1):173–218.

Kwiatkowski, T.; Zettlemoyer, L.; Goldwater, S.; and Steedman, M. 2011. Lexical generalization in ccg grammar induction for semantic parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, 1512–1523. Stroudsburg, PA, USA: Association for Computational Linguistics.

Loper, E., and Bird, S. 2002. Nltk: The natural language toolkit. In *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics - Volume 1*, ETMTNLP '02, 63–70. Stroudsburg, PA, USA: Association for Computational Linguistics.

Matuszek, C.; Herbst, E.; Zettlemoyer, L.; and Fox, D. 2013. *Learning to Parse Natural Language Commands to a Robot Control System*. Heidelberg: Springer International Publishing. 403–415.

McCune, W. 2005–2010. Prover9 and mace4. `http://www.cs.unm.edu/~mccune/prover9/`.

Misra, D. K.; Tao, K.; Liang, P.; and Saxena, A. 2015. Environment-driven lexicon induction for high-level instructions. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 992–1002. Beijing, China: Association for Computational Linguistics.

Seagull FJ, Moses GR, P. A. 2008 Aug. Pillars of a smart, safe operating room. In Henriksen K, Battles JB, K. M., ed., *Advances in Patient Safety: New Directions and Alternative Approaches (Vol. 3: Performance and Tools)*.

She, L., and Chai, J. Y. 2016. Incremental acquisition of verb hypothesis space towards physical world interaction. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

She, L., and Connell, J. 2016. Youtube video demo. `https://www.youtube.com/watch?v=WlqZ69iuFiQ`.

She, L.; Yang, S.; Cheng, Y.; Jia, Y.; Chai, J.; and Xi, N. 2014. Back to the blocks world: Learning new actions through situated human-robot dialogue. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, 89–97. Philadelphia, PA, U.S.A.: Association for Computational Linguistics.

Wu, L.; Cirimele, J.; Card, S.; Klemmer, S.; Chu, L.; and Harrison, K. 2011. Maintaining shared mental models in anesthesia crisis care with nurse tablet input and large-screen displays. In *Proceedings of the 24th Annual ACM Symposium Adjunct on User Interface Software and Technology*, UIST '11 Adjunct, 71–72. New York, NY, USA: ACM.

Wu, L.; Cirimele, J.; Bassen, J.; Leach, K.; Card, S.; Chu, L.; Harrison, K.; and Klemmer, S. 2013. Head-mounted and multi-surface displays support emergency medical teams. In *Proceedings of the 2013 Conference on Computer Supported Cooperative Work Companion*, CSCW '13, 279–282. New York, NY, USA: ACM.

Zettlemoyer, L. S., and Collins, M. 2009. Learning context-dependent mappings from sentences to logical form. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2 - Volume 2*, ACL '09, 976–984. Stroudsburg, PA, USA: Association for Computational Linguistics.

Zhang, Z.; Sarcevic, A.; Yala, M.; and Burd, R. S. 2014. Informing digital cognitive aids design for emergency medical work by understanding paper checklist use. In *Proceedings of the 18th International Conference on Supporting Group Work*, GROUP '14, 204–214. New York, NY, USA: ACM.