2013 IEEE/RSJ International Conference on
Intelligent Robots and Systems (IROS)
November 3-7, 2013. Tokyo, Japan

Automated Alignment of Specifications of Everyday Manipulation Tasks

Moritz Tenorth
Institute for Artificial Intelligence & TZI*
University of Bremen, Germany
tenorth@cs.uni-bremen.de

Johannes Ziegltrum
Technische Universität München, Germany
johannes.ziegltrum@mytum.de

Michael Beetz
Institute for Artificial Intelligence & TZI*
University of Bremen, Germany
beetz@cs.uni-bremen.de

***Abstract*— Recently, there has been growing interest in enabling robots to use task instructions from the Internet and to share tasks they have learned with each other. To competently use, select and combine such instructions, robots need to be able to find out if different instructions describe the same task, which parts of them are similar and which ones differ. In this paper, we investigate techniques for automatically aligning symbolic task descriptions. We propose to adapt and extend established algorithms for sequence alignment that are commonly used in bioinformatics in order to make them applicable to robot action specifications. The extensions include methods for the comparison of complex sequence elements, for taking the semantic similarity of actions into account, and for aligning descriptions at different levels of granularity. We evaluate the algorithm on two large datasets of observations of human everyday tasks and show that they are able to align action sequences performed by different subjects in very different ways.**

I. Introduction

Over the past years, there has been increasing work investigating how robots can be instructed to do novel tasks. The approaches include understanding directions given by humans [1], using natural-language instructions from the Internet [2], or building a Web-based platform on which robots can exchange tasks among each other [3]. One problem that needs to be solved is the translation from natural language into parametrizations of robot actions. Another one, that we address in this paper, is how to compare instructions to find out if they describe the same task, if parts of them are similar, or which parts are different. Comparing the action sequences in these instructions is challenging because many tasks can be described in various different ways, at different levels of detail, using different verbs for the same actions and different names for the same objects. For instance, the instructions "stir the egg with a fork" or "beat the egg in a bowl" can describe the same action if both occur in a similar task context. In addition to different ways of *describing* a task, there are also different ways of *performing* a tasks – an egg, for example, can be beaten using a fork, a spoon, or a whisk.

As part of the RoboEarth project [3], we are working on a Web-based knowledge base of robot task instructions. The vision of the project is a "Wikipedia for robots" through which information about actions, objects and environment maps can be shared between robots. In this context, reasoning about instructions is important both in the Web-based knowledge base and on the robots that search for information. The knowledge base should be able to find similar tasks and to eliminate or merge duplicate descriptions of the same task. If several tasks share common sub-sequences, these are probably important, and introducing a separate, reusable description for these sequences may be beneficial. A robot that is to use the database needs to select among the results and possibly merge several instructions given for a task.

In this paper, we present methods for comparing and aligning formal task descriptions in order to determine which parts match and which parts differ. We propose to adapt and extend algorithms for sequence alignment that are commonly used in the field of bio-informatics for aligning sequences of nucleobases to match DNA strings, or sequences of amino acids to compare proteins. Compared to robot actions, applications in bioinformatics have different requirements though: While robot action sequences are rather short, DNA sequences can become extremely long, but are of simpler structure. Their elements are usually considered as atomic entities and are described by single letters (A, C, G, T). Everyday tasks, in contrast, have a complex structure that relates action classes to the involved objects, tools and locations, and can also be hierarchically structured and composed of sub-actions. We therefore introduce different extensions to account for this more complex structure and present results how the algorithms can be applied to the comparison of action sequences in everyday manipulation tasks.

The main contributions of this paper are (1) the proposal to apply and extend *sequence alignment algorithms* for reasoning about robot action descriptions; extensions of these algorithms to account for (2) the *complex structure of robot actions* when comparing sequence elements, to include (3) background knowledge about the *semantic similarity* of actions and objects for more flexible matching, and to handle (4) action descriptions at *different levels of abstraction*.

II. Related work

Sequence analysis is one of the central research topics in the field of bioinformatics and deals with the structural analysis, comparison and matching of sequence of proteins or nucleobases [4], [5]. Current software programs [6] can perform search for and matching between common subsequences on extremely large data sets. However, the considered elements are usually quite simple, e.g. the four atomic nucleobases, as oppposed to complex robot action specifications. Similar things hold for string matching algorithms [7] and the computation of string edit distances [8] – related algorithms in computer science that also consider

*The Centre for Computing Technologies (TZI).

only simple elements and that use constant costs for the insertion, deletion, and replacement of characters. In the area of action recognition, there is research on aligning annotations, like scripts of movies, with video sequences [9], [10]. Again, the actions are considered atomic, and no deeper reasoning about their properties is performed. In robotics and AI, the focus has mostly been on the *generation* of action sequences using planning [11] or on understanding and executing instructions [1], [2], but much less on the problem of aligning and merging instructions that are already given.

III. Action representation

In this work, we compare action sequences represented in the formal RoboEarth language [12]. In previous work, we have shown how such descriptions can be generated from natural-language task instructions found on web pages like *wikihow.com* [2], and also how they can be created by action recognition and segmentation methods applied to observations of human manipulation tasks [13], so we assume these descriptions to be given.

The RoboEarth language provides a formal, expressive, and flexible specification of robot action plans in terms of abstract action classes. These descriptions of action schemata specify the abstract structure of actions as well as the relations to the manipulated objects and the respective locations. Complementary to this class-level representation is the description of observed actions as instances of these classes that inherit all the properties described at the class level. For instance, observations of human activities would be described as a set of action instances, while robot plans are stored as abstract action schemata. The action specifications in the RoboEarth language are represented in terms of Description Logic in the Web Ontology Language (OWL) [14]. The ontology of actions we use is part of the KnowRob ontology [15] and currently contains class descriptions of more than 130 actions commonly observed in human everyday activities, e.g. in household tasks. These classes serve as the building blocks for describing complex tasks.

Our action representation makes strong use of hierarchical structures: On the one hand, the action ontology arranges action classes in a hierarchy of more and more specialized classes (e.g. stating that *PickingUpAnObject* is a specialization of *ActionOnObject*). On the other hand, there is also a temporal hierarchy describing the composition of complex actions from sub-actions. For example, the action *PuttingSomethingSomewhere* for transporting an object from one position to another involves the sub-actions of picking up the object, moving to the goal position, and putting the object down again. An example of the description of sub-actions is given in the following OWL fragment:

```
Class: PuttingSomethingSomewhere
  SubClassOf:
      Movement-TranslationEvent
      TransportationEvent
      subAction some PickingUpAnObject
      subAction some CarryingWhileLocomoting
      subAction some PuttingDownAnObject
```

Action plans derive task-specific action classes from the abstract classes in the ontology and specify their properties like the *objectActedOn* or the *fromLocation*:

```
Class: PutCupToTable
  SubClassOf:
      PuttingSomethingSomewhere
      objectActedOn some Cup
      fromLocation some Cupboard
      toLocation some EatingTable
```

These constructs can be used to describe hierarchical robot plans that are composed of several actions which interact with different objects, and to specify further action properties like constraints on the motions to perform. Information about the action hierarchies from the robot's ontology is used to compute similarities between actions and objects and to translate between descriptions at different levels of the composition hierarchy.

IV. Sequence Alignment Algorithms

When comparing sequences, one needs to take both the order, the type and the properties of the sequence elements into account. Two sequence elements at a given position can thereby either be equal in both sequences (match), or unequal (mismatch), or an element exists in only one of the sequences, but not in the other (insertion/gap). Given that a cost can be computed for the match between two single elements, the problem of aligning two sequences can be formulated as finding the minimal set of atomic transformations that are needed to translate one sequence into the other.

This alignment problem is very common in bioinformatics, which has led to the development of efficient algorithms based on Dynamic Programming for computing the optimal alignment between two sequences. The two main algorithms are the Needleman-Wunsch algorithm [4] and the Smith-Waterman algorithm [5]. The former computes a global alignment of two complete sequences, while the latter tries to find the optimal local alignment, i.e. the best alignment of a subsequence, and ignores non-matching parts before or after that sequence. In order to apply these algorithms to the alignment of robot action specifications, different extensions are required to account for their more complex structure and the fact that actions can not only be strictly equal or not, but also similar to some degree. These extensions are described in more detail in Section V.

A. Needleman-Wunsch algorithm

The Needleman-Wunsch algorithm progressively computes the optimal global alignment of two sequences from the alignments of sub-sequences. Let $\mathbf{x}$ and $\mathbf{y}$ be sequences of length n and m, respectively, while x_i is the ith symbol in x, and y_j the jth symbol in y. The algorithm computes the optimal alignment score for all prefixes of both sequences in a matrix F. The entry $F(i,j)$ contains the optimal solution for the alignment of the sub-sequences $x_0...x_i$ and $y_0...y_j$. After the algorithm has been executed, the optimal alignment score can thus be found in the field $F(n,m)$. The algorithm consists of three main steps:

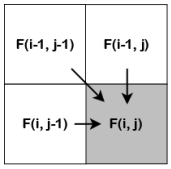

Fig. 1. Computing the cost of an alignment score based on the comparison of the current sequence elements and the neighboring matrix elements.

a) Matrix initialization: The matrix is initialized in such a way that the first row and first column correspond to an alignment of sequence **x** with only gaps in sequence **y** and vice versa. The element $F(0,0)$, corresponding to an alignment of the two first sequence elements, is initialized with zero, while the other fields are set as follows (with gap costs d):

$$F(0,0) = 0 \tag{1}$$

$$F(i,0) = -i \cdot d \tag{2}$$

$$F(0,j) = -j \cdot d \tag{3}$$

b) Recursive computation of alignment scores: The other alignment scores can be computed from the matching cost $S(x_i, y_j)$ of the sequence elements x_i and y_j and from the neighboring matrix elements that contain the alignment scores of the preceding part of the sequence. Any field $F(i,j)$ can be reached by either from the upper left $F(i-1, j-1)$, the left $F(i-1, j)$, or the top $F(i, j-1)$ (Figure 1), and its value can be computed as

$$F(i,j) = max \begin{cases} F(i-1,j-1) + S(x_i, y_j) \\ F(i-1,j) - d \\ F(i,j-1) - d \end{cases} \tag{4}$$

c) Traceback: Each field $F(i,j)$ contains a pointer to the field from which it has been reached. Once the matrix has been completely built up, these pointers can be tracked back from the final element to the origin to reconstruct the path corresponding to the optimal alignment of the two sequences.

B. Smith-Waterman algorithm

The Smith-Waterman algorithm computes the optimal alignment between sub-sequences. It introduces the additional condition that $F(i,j) = 0$ if all three alignment options yield a negative score:

$$F(i,j) = max \begin{cases} 0 \\ F(i-1,j-1) + S(x_i, y_j) \\ F(i-1,j) - d \\ F(i,j-1) - d \end{cases} \tag{5}$$

As a result, there is no longer a path leading through the whole matrix because paths are finished once their score falls to zero. The optimal alignment score can therefore be found anywhere in the matrix, so the algorithm needs to search for the largest matrix element and track its path to the next zero-entry. Multiple optimal local alignments may exist [5].

V. Extensions for Aligning Action Sequences

The sequence alignment algorithms were developed for aligning DNA sequences consisting of nucleobases that are commonly represented as single letters A, C, G, and T. In order to apply these algorithms to action sequences, they need to be extended to the complex structure of robot action that are usually not atomic, but composed of a type and several properties. Furthermore, actions can also match partially and can be described at different levels of abstraction. We have therefore modified the algorithms to make them applicable to the alignment of robot action sequences.

A. Accounting for complex sequence elements

Action sequences often differ only in parts, for example if a different tool is used for an action. Not all parts are equally important, the action type and the main object are normally more relevant than e.g. the location where the action is performed. This should be taken into account when comparing two actions, so we compute the match value by a weighted combination of the matches of their components. Assume an action act_i is described by its type a_i and a set of n role-value pairs:

$$act_i \ = \langle a_i, \langle p_i^1, o_i^1 \rangle, \ldots \langle p_i^n, o_i^n \rangle \rangle \tag{6}$$

The match between two actions can then be computed by the weighted match values of the action types plus the weighted sum of the matches between the respective roles p_i^k and values o_i^k (e.g. objects or locations):

$$sim(act_i, act_j) = \tag{7}$$
$$\alpha \cdot sim(a_i, a_j) + \sum_{k=1}^{n} \beta_k \cdot sim(p_i^k, p_j^k) + \gamma_k \cdot sim(o_i^k, o_j^k)$$

The weighting parameters α, β_k and γ_k determine the influence of the individual components. For example, the roles p_i^k are often correlated with the action types a_i^k (e.g. "take" and "from") and should therefore have less influence. For the experiments, we empirically chose the parameters $\alpha = 0.3$, $\beta_1 = 0.5$, $\gamma_1 = 0.6$ (for the *objectActedOn* as the most important object the action is performed on), and $\beta_k = 0.1$, $\gamma_k = 0.3$ for all other properties and values. This weights the three main components (action, first and second object) with a factor of 0.3, while the preposition's weight is 0.1. The resulting values are normalized to the range $[-1, 1]$ to be compatible with the costs for match and mismatch.

B. Semantic similarity computation

The computation of how well two descriptions match should further take knowledge about the actions and objects into account: If a similar action is performed on a similar object, it should not be treated as a complete mismatch. Based on the ontology of action classes, the system can compute the similarity between concepts by their closeness in the ontology. The "WUP similarity" was originally proposed by Wu and Palmer in [16]. For two concepts in an ontology, it defines a similarity value in the interval $[0; 1]$, considering the depth of the concepts and the depth of their lowest common super-concept (LCS):

$$wupSim(C_1, C_2) = \frac{depth(LCS(C_1, C_2))}{\frac{1}{2}\left(depth(C_1) + depth(C_2)\right)} \tag{8}$$

The reflexive case is defined as $wupSim(C, C) = 1$. The computation is illustrated in Figure 2, showing a snippet of our ontology. Concepts in the ontology can have multiple

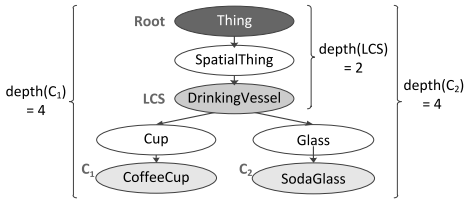

Fig. 2. Computation of the WUP similarity metric. The similarity of the concepts *wupSim*(CoffeeCup and SodaGlass) is computed as 0.5.

super-concepts – the WUP similarity computes the minimum of all these distances since it considers the lowest common super-concept (LCS). The structure of the ontology therefore influences the similarity values. In our experiments, the KNOWROB ontology, though not specifically created for this problem, yielded very reasonable results. In order to increase the weight of higher similarities, we use the *wupSim* to the power of three for computing the alignment costs.

C. Hierarchical structure analysis

Actions can be described at different levels of abstraction and different granularity. Instructions often mix different levels to describe parts that are quite common, like fetching items from the refrigerator, on a coarse level, while using more details for parts that are more specific to the action at hand and therefore potentially more difficult. The proposed algorithm employs information about sub-action relations from the KNOWROB ontology (see Section III) to expand all action descriptions to the most detailed level before performing the alignment. It remembers the association between the actions in the original and the expanded sequences and can therefore translate them back to the original level of detail after the matching.

VI. EXPERIMENTS

We evaluated the algorithms on two data sets of observations of humans performing everyday manipulation activities like setting a table or preparing simple meals. These datasets provide realistic data of complex object manipulation tasks and contain sufficient variability to give a realistic impression of how the algorithms handle uncertain and noisy data. Part of the variability results from the different ways how different subjects performed the activities, more variability has been introduced because different people have labeled the data in slightly different ways. The data we used for evaluation as well as the source code and the complete results can be found online[1].

A. Evaluation data sets

We used the TUM Kitchen data set [17] and the CMU MMAC data set [18] for evaluation. The former provides data of 20 table setting episodes performed by different subjects in a kitchen environment. The latter consists of observations of different subjects making brownies (16 episodes) and scrambled eggs (17 episodes). Both datasets come with manually created labels that annotate which actions are performed with which properties, e.g. the object that is manipulated or the location from which an object is picked up. We used those labels for our experiments and excluded aspects like the segmentation and classification of the motions which are not the focus of this paper.

The annotations of the TUM data set describe only the type of the action, but are very fine-grained and therefore well usable to test the matching using hierarchical abstraction. The CMU data set provides more complex labels in the form "verb [object1 [preposition object2]]". This structured format can easily be converted into the representation in the ROBOEARTH language. The 'object1' is implicitly considered the *objectActedOn* of the action verb, the relation between the action and 'object2' is determined depending on the 'preposition'.

B. Similarities of action sequences

In a first experiment, we computed the pair-wise alignment scores of the 16 sequences of making brownies and the 17 sequences of making scrambled eggs (called 'eggs' in the data) from the CMU data set. While both tasks are different, they share common sub-sequences like cracking eggs into a bowl and stirring them. Each of the sequences has been performed by a different subject with minimal instructions on how to do the task. The different ways how the subjects performed the task result in much variability between the sequences, which also differ quite significantly in length (from 39 to 107 actions for making brownies, from 44 to 127 for making eggs). The algorithm thus needs to compute the optimal alignment between each pair of the 33 sequences that each consist of 39–127 elements that are each composed of the action type, the *objectActedOn*, and another prepositional relation (e.g. *from, to, on*, etc). The matching costs between these sequence elements are determined based on a weighted sum (Eq. 7) of the semantic similarities (Eq. 8).

Figure 3 shows the alignment scores that have been computed using the WUP-similarity in the form of a matrix. The diagonal, corresponding to the alignment of a sequence with itself, is always one, so a perfect self-alignment could be found. The algorithm can well distinguish between the activities, indicated by the clearly visible blocks in the upper left and lower right, despite significant differences in how they are performed by the different subjects, and despite common parts in both activities (cracking eggs, mixing ingredients in a bowl). The alignment scores for Bro_{S13}, Egg_{S13}, Egg_{S16}, and Egg_{S20} are lower than the others, visible as the lighter-gray cross in the darker blocks, which is because they are about twice as long as the other sequences and therefore differ substantially.

C. Alignment with and without semantic similarities

Quantitatively comparing alignments is difficult: First of all, there is no gold-standard alignment. For example, if a sub-sequence is matched to a gap in the other sequence, it does not matter if a non-matching action is put before or after that gap. Apart from that, the decision if actions are sufficiently similar to be matched needs to be done on a case-by-case basis. We therefore present qualitative

[1] https://github.com/knowrob/action_alignment

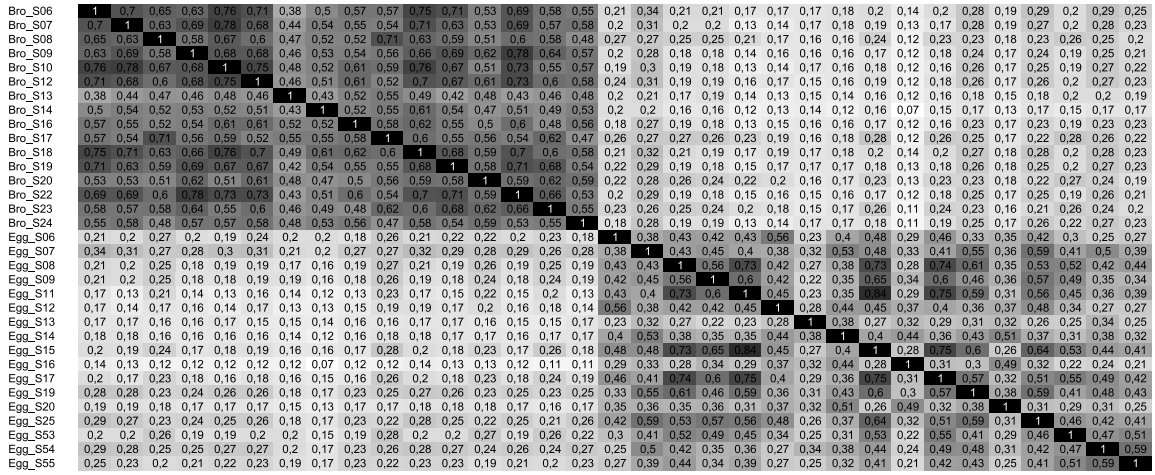

Fig. 3. Matrix of the similarities between all 33 action sequences for making brownies and scrambled eggs. The clearly visible blocks indicate that actions inside one activity are more similar to each other than across activities.

results. Figure 4 shows part of a global alignment of the sequences Egg_{S20} and Egg_{S25} without (left) and with similarity-based matching (right). For improved readability, the figure uses natural-language labels instead of the formal representation. One can see that the matching became more flexible, associating *beat egg with fork* and *stir small-bowl with fork*. A better alignment of one action can also lead to a better alignment of the neighboring actions. In this example, the action *switch-on stove* can be matched because the correspondence between the two other actions was found. On the other hand, the more flexible matching can also lead to mis-alignments depending on how the similarities are weighted. The impact of incidental mis-alignments versus a globally better alignment has to be traded off for each scenario. We found that the improvement achieved by a similarity-based comparison strongly depends on how much variability is in the annotations. The brownie-sequences have been annotated by a different person than the eggs-sequences and are labeled more consistently, which can also be seen by the darker upper left block in Figure 3. As a result, there are very much fewer differences between the hard and the similarity-based matching.

D. Hierarchical abstraction

Since the available data sets were all annotated at a fixed level of abstraction, we synthetically created test data at a coarser level. The abstraction can best be shown in the TUM data set, because it is annotated at a more detailed level than the CMU data and therefore leaves more room for abstraction. The following sub-action relations were read from the KNOWROB ontology:

```
PuttingSomethingSomewhere
  - PickingUpAnObject
  - CarryingWhileLocomoting
  - PuttingDownAnObject

PickingUpAnObject
  - Reaching
  - TakingSomething
PuttingDownAnObject
  - LoweringAnObject
  - ReleasingGraspOfSth
```

Using these relations, the system can flexibly convert between annotations and align sequences that have been annotated at different levels of abstraction. In the example below, we have created an abstracted version of the annotations for the right hand in sequence *0-7* of the TUM data set (right side), and aligned it with the right-hand annotations of sequence *0-11* (left side). The abstracted labels are on either the medium level (*PickingUpAnObject* and *PuttingDownAnObject*) or the most abstract one (*PuttingSomethingSomewhere*).

```
CarryingWhileLocomoting - CarryingWhileLocomoting
              Reaching - PickingUpAnObject
       TakingSomething - ()
CarryingWhileLocomoting - CarryingWhileLocomoting
      LoweringAnObject - PuttingDownAnObject
  ReleasingGraspOfSth - ()
CarryingWhileLocomoting - CarryingWhileLocomoting
              Reaching - PuttingSomethingSomewhere
       TakingSomething - ()
CarryingWhileLocomoting - ()
      LoweringAnObject - ()
  ReleasingGraspOfSth - ()
CarryingWhileLocomoting - CarryingWhileLocomoting
              Reaching - PuttingSomethingSomewhere
CarryingWhileLocomoting - |
              Reaching - |
       TakingSomething - ()
CarryingWhileLocomoting - ()
      LoweringAnObject - ()
  ReleasingGraspOfSth - ()
CarryingWhileLocomoting - CarryingWhileLocomoting
                    [...]
```

While the sequences on the left and on the right side are substantially different, the system is able to align them based on its knowledge about the composition of actions from sub-actions.

E. Discovery of frequent sub-sequences

If the system detects that some actions are regularly performed after each other, for example the sequence of walking to the refrigerator, opening the door, taking some object out, and closing the door, it could make use of this observation to automatically perform an abstraction. In the context of the ROBOEARTH knowledge repository, creating a specialized "action recipe" for such a sequence will allow to re-use it in other contexts. Similar to the global alignment, the local alignment profits as well from a more flexible matching based on semantic similarities. For example, the optimal local alignment found by the Smith-Waterman algorithm for the sequences Egg_{S55} and Egg_{S06} is:

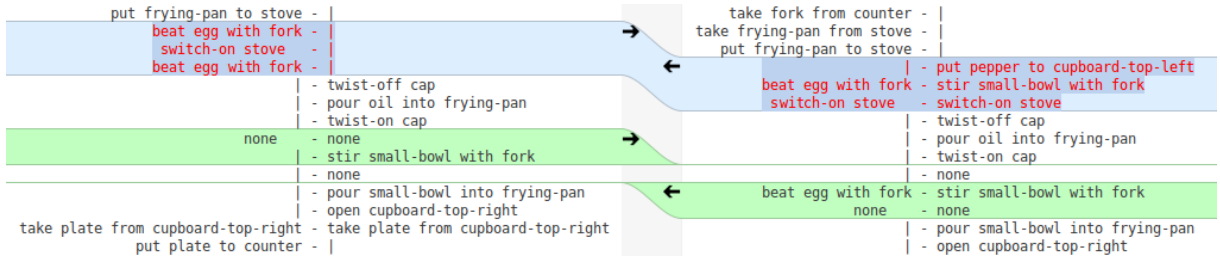

Fig. 4. Alignment without (left) and with WUP similarity-based matching (right). Using this more flexible matching approach, the system can find the correspondence between non-equal action descriptions.

```
none          - none
walk to fridge - walk to fridge
open fridge    - open fridge
```

Such a short subsequence is of quite limited use. Using the WUP similarity-based matching, the system is able to generate much longer and more meaningful alignments:

```
none          - none
walk to fridge - walk to fridge
open fridge    - open fridge
open egg-box   - |
take egg from egg-box - take egg from fridge
close egg-box  - |
close fridge   - close fridge
walk to counter - walk to counter
```

VII. Conclusions

In this paper, we presented extensions to sequence alignment algorithms to make them applicable to the alignment of robot action specifications. We adapted the methods for comparing sequence elements to account for the complex relational structure of action specifications and included semantic similarity measures to achieve more flexible matching. Knowledge about the hierarchical structure of actions can be exploited to align sequences described at different levels of granularity. We evaluated the methods on two large real-world datasets of observations of making brownies, making an omelette, and setting a table. The system was able to align the sequences and to determine common subsequences as well as parts that differ between the examples.

In our current work, we are integrating the presented methods into the RoboEarth system to find duplicates in action descriptions and merge similar descriptions of the same task. In a related project, we have investigated methods for learning expressive action models that can also represent the dependency structure between the actions [19]. While those models are more expressive than the sequence-based models of this paper, their complexity is also much higher. We plan to bring these approaches together to see whether both can profit from each other.

Acknowledgments

This work is supported in part by the EU FP7 Projects *RoboEarth* (grant number 248942) and *RoboHow* (grant number 288533).

References

[1] C. Matuszek, E. Herbst, L. Zettlemoyer, and D. Fox, "Learning to parse natural language commands to a robot control system," in *Proc. of the 13th International Symposium on Experimental Robotics (ISER)*, June 2012.

[2] M. Tenorth, D. Nyga, and M. Beetz, "Understanding and Executing Instructions for Everyday Manipulation Tasks from the World Wide Web," in *IEEE International Conference on Robotics and Automation (ICRA)*, Anchorage, AK, USA, May 3–8 2010, pp. 1486–1491.

[3] M. Waibel, M. Beetz, R. D'Andrea, R. Janssen, M. Tenorth, J. Civera, J. Elfring, D. Gálvez-López, K. Häussermann, J. Montiel, A. Perzylo, B. Schießle, O. Zweigle, and R. van de Molengraft, "RoboEarth - A World Wide Web for Robots," *Robotics & Automation Magazine*, vol. 18, no. 2, pp. 69–82, 2011.

[4] S. B. Needleman and C. D. Wunsch, "A general method applicable to the search for similarities in the amino acid sequence of two proteins," *Journal of Molecular Biology*, vol. 48, no. 3, pp. 443 – 453, 1970.

[5] T. F. Smith and M. S. Waterman, "Identification of common molecular subsequences," *Journal of Molecular Biology*, vol. 147, no. 1, pp. 195–197, 1981.

[6] S. Altschul, T. Madden, A. Schäffer, J. Zhang, Z. Zhang, W. Miller, and D. Lipman, "Gapped blast and psi-blast: a new generation of protein database search programs," *Nucleic acids research*, vol. 25, no. 17, pp. 3389–3402, 1997.

[7] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *String Matching*. MIT Press, 2009, ch. 32, pp. 906–932, in Introduction to Algorithms, Second Edition.

[8] V. I.Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," in *Soviet Physics Doklady*, vol. 10, no. 8, 1966, pp. 707–710.

[9] O. Duchenne, I. Laptev, J. Sivic, F. Bach, and J. Ponce, "Automatic annotation of human actions in video," in *2009 IEEE 12th International Conference on Computer Vision*, September 2009, pp. 1491–1498.

[10] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld, "Learning realistic human actions from movies," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2008, pp. 1–8.

[11] D. McDermott, "Robot Planning," *AI Magazine*, vol. 13, no. 2, pp. 55–79, 1992.

[12] M. Tenorth, A. C. Perzylo, R. Lafrenz, and M. Beetz, "The RoboEarth language: Representing and Exchanging Knowledge about Actions, Objects, and Environments," in *IEEE International Conference on Robotics and Automation (ICRA)*, St. Paul, MN, USA, May 14–18 2012.

[13] M. Beetz, M. Tenorth, D. Jain, and J. Bandouch, "Towards Automated Models of Activities of Daily Life," *Technology and Disability*, vol. 22, no. 1-2, pp. 27–40, 2010.

[14] W3C, *OWL 2 Web Ontology Language: Structural Specification and Functional-Style Syntax*. World Wide Web Consortium, 2009, http://www.w3.org/TR/2009/REC-owl2-syntax-20091027.

[15] M. Tenorth, "Knowledge processing for autonomous robots," Ph.D. dissertation, Technische Universität München, 2011. [Online]. Available: http://nbn-resolving.de/urn/resolver.pl?urn:nbn:de:bvb:91-diss-20111125-1079930-1-7

[16] Z. Wu and M. Palmer, "Verbs semantics and lexical selection," in *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*. Association for Computational Linguistics Morristown, NJ, USA, 1994, pp. 133–138.

[17] M. Tenorth, J. Bandouch, and M. Beetz, "The TUM Kitchen Data Set of Everyday Manipulation Activities for Motion Tracking and Action Recognition," in *IEEE International Workshop on Tracking Humans for the Evaluation of their Motion in Image Sequences (THEMIS), in conjunction with ICCV2009*, 2009.

[18] F. De la Torre, J. Hodgins, J. Montano, S. Valcarcel, and J. Macey, "Guide to the Carnegie Mellon University Multimodal Activity (CMU-MMAC) Database," CMU-RI-TR-08-22, Robotics Institute, Carnegie Mellon University, Tech. Rep., 2009.

[19] M. Tenorth, F. D. la Torre, and M. Beetz, "Learning probability distributions over partially-ordered human everyday activities," in *IEEE International Conference on Robotics and Automation (ICRA)*, Karlsruhe, Germany, May 6–10 2013.