A Knowledgebase with which you can generate robot plan for multiple mixing actions

Master Thesis

Naser Azizi, Sorin Arion

Prüfer der Master Thesis:	1.	Prof. Michael Beetz PhD
	2.	
Supervisor		Michaela Kümpel

Eidesstattliche Erklärung

Hiermit erklären wir, dass die vorliegende Arbeit selbstständig angefertigt, nicht anderweitig zu Prüfungszwecken vorgelegt und keine anderen als die angegebenen Hilfsmittel verwendet habe. Sämtliche wissentlich verwendete Textausschnitte, Zitate oder Inhalte anderer Verfasser wurden ausdrücklich als solche gekennzeichnet.

Bremen, den 27. Dezember 2023

Naser Azizi, Sorin Arion

Motivation

Although industrial robots have been considered standard for many years, the widespread adoption of AI-based autonomous household robots is still far from being anticipated in every household. Obstacles such as unfamiliar surroundings and a limited knowledge base can significantly hinder the ability of an autonomous robot to effectively respond in a given situation, hence, the robot must precisely plan for every minor detail, even those that might appear trivial to us humans. To achieve the defined objectives, a proficient and comprehensive robotic system including Control, Perception, Navigation, and Knowledge is necessary. An illustrative goal could be cooking. To execute this task, the robot must possess various capabilities, including placing items, grasping objects, pouring liquids, cutting ingredients, and mixing components. Those things seem trivial to us humans, but for the robots it requires a lot of Implementation and information. In this thesis, we aim to introduce an approach detailing how a robotic system can execute mixing actions effectively. The challenges arise from the diverse methods of mixing, which further vary based on the specific ingredients being combined. Additionally, the consideration of compatible Mixing Tools and Mixing Containers is crucial, as not every tool can be utilized with every container. Through the incorporation of a knowledge graph containing rules related to to various actions, we aim to empower the robotic system to make informed decisions on the appropriate motions to employ. This decision-making process will take into account the specific task at hand and the involved ingredients. By incorporating this knowledge graph, we aim to advance towards autonomous robots capable of engaging in cooking activities.

Introduction

First, we intend to analyze existing research, particularly examining studies that delve into diverse mixing motions and exploring related systems that contain a queryable knowledge graph. In the following section, we will introduce the reader with the frameworks employed to accomplish our objectives. Subsequently, we will present our approach to data acquisition and data representation. In this chapter, we aim to explain the mixing tasks under consideration and articulate our methodology for representing this data to enable effective querying. In the subsequent section, we will illustrate the implementation of our rules, with the ultimate aim of determining the appropriate motion to be employed. To validate our concept, we have opted to simulate various scenarios and assess their outcomes, demonstrating the efficacy of our implemented system. Finally, we will delve into the discussion of our results and draw conclusions to wrap up our work.

Related Work

There are multiple approaches for mixing tasks in the kitchen domain. From high level symbolic to low level motions and learning approaches, these different approaches, tries to tackle the challenge of solving mixing in the kitchen environment.

FoodCutting aims to equip robotic agents with necessary information about how to execute cutting tasks in unknown environments for the household domain. From high level plans FoodCutting breaks down the cutting task, part of some robot plan, into executable motions regarding cutting fruits and vegetables. These motions are parameterized by some technique and repetitions to achieve the agents objective. FoodCutting does not require fully available knowledge about the agents environment, instead the robot should be capable of recognizing certain objects for cutting operations.

BakeBot realised on the PR2 robotics platform attempts to achieve baking cookies. An implementation of locating relevant things for MixingTasks like a bowl and ingredients has been realised for semi-structured environments. An algorithm to perform mixing motions has been implemented as well, to mix ingredients with different characteristics into an uniform dough. These motions are limited to a simple circular and linear mixing motion. The authors follow an bottom-up approach which is inherently motion driven rather than a symbolic one which attempts to break down high level tasks into executable low level motions.

FluidLab is a simulation environment for different kinds of manipulation tasks regarding liquids. Its underlying engine uses differentiable physics enabling reinforcement learning and optimization techniques in manipulation to utilize the engine, to achieve several tasks including liquids and solids, like mixing tasks.

Our mixing approach will be most similar to the FoodCutting approach, in which we model symbolic knowledge about how to perform mixing tasks, which technique should be used and infering parameters for the execution of the underlying motion.

How does the robot works / plans, parameters, used tools, etc

In this chapter we want to introduce the reader to the frameworks and libraries we are using, also we want to briefly present the robot that should be able to perform the implemented tasks.

Pr2

Introduced in 2010 by Willow Garage (https://robotsguide.com/robots/pr2), the PR2 stands as an advanced research robot. Boasting multiple joints and 20 degrees of freedom, this robot excels in autonomous navigation and the manipulation of a diverse array of objects, making it an ideal choice for our specific needs. Additionally, it is equipped with a HeadStereoCamera that can be used to perceive the surroundings.

PyCram

Robotic systems are complex, comprising various components such as Knowledgebase, Perception, Manipulation/Control, Navigation, and more. Managing these diverse elements can pose challenges and requires multiple interfaces between them. One potential strategy is to integrate all these interfaces into a unified framework. An example of such a comprehensive framework is PyCram(https://github.com/cram2/pycram). Implemented in Python3, the PyCram framework incorporates ROS for robot communication. Utilizing PyCram, we can execute High-Level plans on the PR2 to evaluate our implementation.

Ontology

Ontologies are structured frameworks that provide a formal representation of knowledge within a specific domain. They play a crucial role in knowledge representation, facilitating the organization and sharing of information in a way that is both machine-readable and understandable by humans.

Key components of ontologies include:

- Concepts/Classes: These represent abstract or concrete entities within a domain. For example, in a medical ontology, "Patientänd "Disease"might be classes.

- Properties/Roles: These define the relationships between concepts. For instance, in a social network ontology, "FriendOf"could be a property connecting individuals.

- Instances/Individuals: These are specific members or examples of a class. In a geographical ontology, "New York Cityänd "Paris"could be instances of the class "City."

- Axioms: These are statements that describe the properties and relationships of the entities within the ontology. Axioms help define the logic and rules governing the domain.

- Hierarchy: Ontologies often organize concepts into a hierarchy, with more general concepts at the top and more specific ones below. This hierarchical structure aids in categorization and understanding.

- Inference Rules: These rules define how new information can be derived from existing information in the ontology. Inferences help systems reason and make deductions based on the knowledge encoded in the ontology.

We utilize ontologies in knowledge representation and reasoning systems to empower the robot with the ability to comprehend and handle information in a structured fashion for our specific objectives.

OWLReady

One important library used for our Implementation is OWLReady. ÖWLReadyïs a Python library designed for ontology-oriented programming. It facilitates the development, manipulation, and querying of ontologies using the Web Ontology Language (OWL), a standard for representing knowledge in a machine-readable format. OWLReady simplifies ontology-related tasks by providing a convenient and object-oriented interface for working with OWL ontologies in Python.

Key features of the OWLReady library include:

- Object-Oriented Programming (OOP): OWLReady adopts an object-oriented approach, allowing users to interact with ontology entities as Python objects. This makes it more intuitive for developers familiar with Python's OOP principles.

- Ontology Loading and Parsing: The library supports the loading and parsing of OWL ontologies, making it easy to access and manipulate ontology data within Python scripts or applications.

- Class and Individual Manipulation: OWLReady provides functionality for creating, modifying, and querying classes and individuals within an ontology. This allows for dynamic and programmatic management of ontology content.

- Reasoning Support: Depending on the version and features, OWLReady may offer support for reasoning tasks. Reasoning involves deducing implicit information based on the logical relationships defined in the ontology.

- Integration with RDFLib: OWLReady may integrate with RDFLib, another Python library commonly used for working with Resource Description Framework (RDF) data. This integration enhances the capabilities of handling semantic data.

SWRL

SWRL, which stands for Semantic Web Rule Language, is a rule language that allows users to define rules about the relationships between classes and individuals in ontologies represented in the Web Ontology Language (OWL). SWRL is part of the W3C's Semantic Web technology stack and is designed to be used in conjunction with OWL to express complex relationships and infer new information based on existing knowledge.

SWRL Rules have a specific syntax and consist of two main components:

- Antecedent (Body): This part of the rule specifies the conditions or constraints that must be satisfied for the rule to be applicable. It describes the current state of the ontology that triggers the rule.

- Consequent (Head): This part defines the actions or inferences that should be taken if the conditions specified in the antecedent are satisfied. It describes the changes or additional information that should be inferred when the rule is triggered.

SWRL supports various built-in predicates and functions, and users can create their own custom rules to suit their specific ontology. Some common elements in SWRL rules include:

- Individuals: Refers to specific instances of classes in the ontology.

- Class and Property Relationships: Describes relationships between classes and properties in the ontology.

- Built-in Predicates and Functions: Includes operations such as arithmetic, string manipulation, and comparison functions that can be used in the rule conditions.

Here's a simple example of a SWRL rule:
$Person(?x)^h asChild(?x, ?y) -> Grandparent(?x, ?z)^h asChild(?y, ?z)$
In this example:

If an individual (?x) is a Person and has a child (?y),

Then, infer that the individual (?x) is a Grandparent of another individual (?z), and the child (?y) is the parent of (?z).

SWRL rules are useful for expressing complex relationships and constraints within ontologies, enabling automated reasoning systems to make inferences and derive new knowledge from existing data.

Data acquisition

The first step in acquiring the needed data, was to acknowledge which task varations of mixing are actually important. So we had to analyze the word *Mixing* and its hyponyms. hyponyms are subordered words of a given word, for example one hyponym of mixing could be beating. By looking on different websites like "framenet, Wordnet", we created a list of possible hyponyms, that could be taken under consideration. For all those hyponyms, we delegated a WikiHow extraction search which should show us, how many times one of these words occur, in the context of cooking. In the table below the results can be seen.

Hyponym	**Occurance**
Mixing	5300
Combining	3700
Stirring	6810
Folding	970
Merging	7
Beating	1200
Whipping	1000
Join	57
Coalesce	1
Amalgamate	0
Pair	57
Blending	1400

Tabelle 0.1: Example Table

From this table one can see that many hyponyms have a low occurance and will be therefore not considered. We consider the words: Mixing, Stiring, Beating, Combining, Whiping, Whisking and Folding. Though Blending achieved a moderate occurance, we decided upon not considering it, because it involves a lot of electronic mixing machines, which the robot could not handle. From now on we will refer to these words also as Tasks. After deciding upon which tasks we consider, we have to analyze every task execution and define a structure. We started by analyzing different videos on WikiHow that included at least one of our choosen task and divided the information by:

Task, used Tools, used Container, Ingredients, Motion.

- Tasks: The tasks can differ in its execution, like the tasks Beating (Mixing can be

gentle or vigorous, while beating involves a more forceful and rapid action) and Folding (To maintain a light and fluffy texture in a cake, folding is often used to add dry ingredients without overmixing), which will ultimately lead to a decision upon which Motion should be used in the specific use case.

- Tools: Different Tools are used for different tasks, which also has a correlation with the choosen Container.

- Container: Most Mixing Tasks use a mixing bowl for container, but also container like Pot and Pan when considering cooking in boiling water.

- Ingredients: Besides Tasks, the ingredients are the most important component of our knowledgebase regarding the decision making upon motions. We divide the Ingredients in 4 subcategories: Dry/Powder, Wet, Liquid and Solid.

- Motions: The motions used in the analyzed videos will help us upon creating a knowledgebase from which the robot should be able to infer which motions he will use in different situations.

In the following we want to present our analysis results regarding the videos from WikiHow, but also from well known cooks, like Jamie Oliver:

Task	**Tool**	**Container**	**Ingredients**	**Description**
Beating	Whisk	Bowl	Egg yolk (Wet ingredient)	circular, swirling wildly around the bowl
Stirring	Whisk	Bowl	Beaten Egg Yolk (Wet), Parmesan(Powder) and Pepper (Powder)	Circular, from the inside to the outside.
Stirring	Tongs	Pan	Wet Mixture, Pasta (Solid) and Bacon (Solid)	Diving motions, circular but also straight lines.
Whisk	Fork	Bowl	Eggs (Wet)	Circular but also straight, wildly motion.
Mixing	Spatula	Pan	Eggs, melted butter (Wet)	Circular, from the inside to the outside, also diving.
Folding	Spatula	Pan	cooked eggs in melted butter (Wet)	Gently motion from the outisde to the inside straight, then moving about 90 degree before going to the inside again.
Mixing	Spoon	Cup	Dry yeast(Powder), Water (Liquid)	Circular
Mixing	Spoon	Bowl	Dry yeast, Water, Flour (Powder), Salt(Powder)	Whirlstorm-like motion.

Tabelle 0.2: Example Table

From this videos we extracted informations about the executed motions. The following motions were defined:

- **Circular**: Moving the tool in a defined circular movement in the container, not changing the radius during execution

- **Whirlstorm**: Moving from the inside to the outside of the container with the tool, by circulating in an incremented radius.

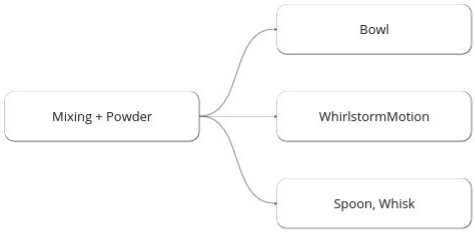
- **Folding**: Gently motion, where you start from the outside, moving one straight line to the inner side, then picking the tool up and going to the initial state before moving the tool for about 90 degrees, then going back in a straight line to the inner side of the container again.

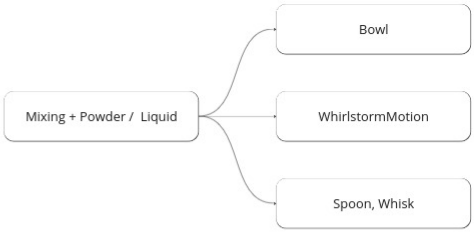
- **VerticalCircular**: Imagine a line which can be seen as the diameter of the container, from this line one can define certain regions on which you move the tool circular from side to side. This motion is used by the beating task.

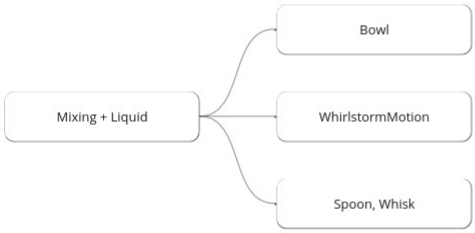
- **CircularDivingToInner**: Starting from the outerside, moving the tool in the container arround its edge for about 270 degrees, before diving to the middle of the container. This motion is used by Tasks where is required to turn the Ingredients over.

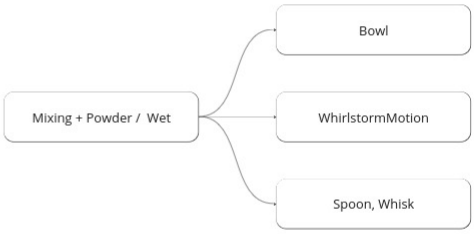
By analyzing the videos we concluded that the motion decision is based on the task and Ingredients. Upon this thoughts we decided to design a decision tree regarding how the motions will be choosen by the robot.

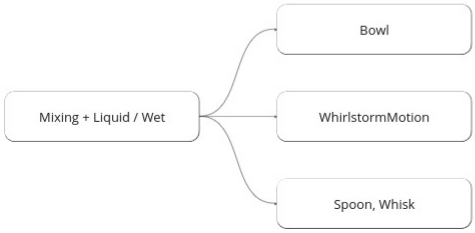
Mixing

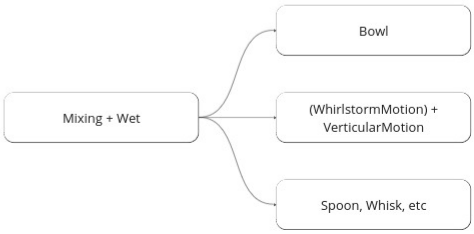

Mixing + Powder
Bowl
WhirlstormMotion
Spoon, Whisk


Mixing + Powder / Liquid
Bowl
WhirlstormMotion
Spoon, Whisk


Mixing + Liquid
Bowl
WhirlstormMotion
Spoon, Whisk


Mixing + Powder / Wet
Bowl
WhirlstormMotion
Spoon, Whisk


Mixing + Liquid / Wet
Bowl
WhirlstormMotion
Spoon, Whisk


Mixing + Wet
Bowl
(WhirlstormMotion) + VerticularMotion
Spoon, Whisk, etc

Data representation

- Ingredients, Tools, etc.
- Different Tasks/Actions maps on different Motions
- Motion dependencies
- Fobidden Tools with Containers
- How is an action defined
- Examples of using the knowledge graph

Implementation

- OWLReady
- SWRL
- Inferring parameters for the actual plan

Simulation

- BulletWorld as Simulation platform -> how is the robot actually shown, which limitations exist
- Quering the knowledge base
- Inferring the parameters in the Simulation
- Task execution with multiple sequential Motions

Simulation to RealWorld gap

- Big Problem: Uncertaininty
- Perception Solution as first approach: RoboKudo
- Data acquisition: Blenderproc
- Model Training: YoloV8
- Results showing the real world Perception.

Evaluation

- Evaluation of multiple motions multiple times
- Results of that

Summary / Fazit

avc