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 16. Januar 2024

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.

Introduction to Hardware and Software components

This chapter serves as an introduction to the frameworks, software, and libraries we use. The work we present can be simplified explained as a knowledge base containing parameters that a robot can use to perform specific actions.

Firstly, we would like to introduce the robotic component, which acts as the interface between the knowledge base we have created and the robot, followed by an introduction to Ontologies and Knowledgebases.

Robotic section: ROS and PyCram

The main framework we use is PyCram, which serves as an interface for various software components such as knowledge, perception, or manipulation. This framework utilizes another framework, ROS, to communicate with the different robot components. These two frameworks are now introduced in the following.

ROS

ROS, which stands for Robot Operating System, is an open-source middleware framework designed to develop and control robots. Despite its name, ROS is not a traditional operating system but rather a set of software libraries and tools that help in building and managing robot software. It provides a standardized and modular approach to developing robotic systems, allowing for easier collaboration and code reuse in the robotics community. Key features and components of ROS include:

- Nodes: ROS systems are organized into nodes, which are individual processes that

perform specific tasks. Nodes communicate with each other by passing messages over topics, creating a decentralized and modular architecture.

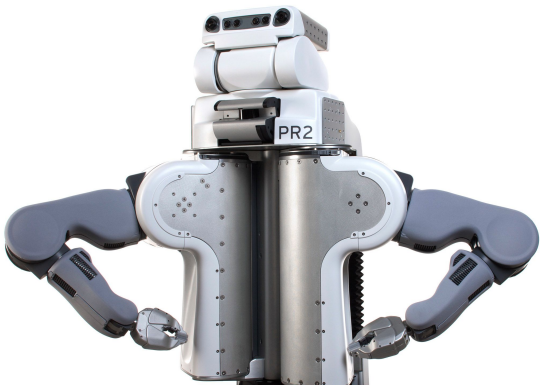
- Topics: Nodes exchange data through topics, which are named buses over which messages are passed. This publish/subscribe communication model allows for asynchronous and loosely coupled interactions between nodes.

- Launch files: ROS uses launch files to specify how to start multiple nodes and configure the system. This helps simplify the setup of complex robotic systems.

- Master: The ROS Master is responsible for managing the communication between nodes by keeping track of publishers, subscribers, and services. It facilitates the discovery and connection of nodes within a ROS network.

Pr2

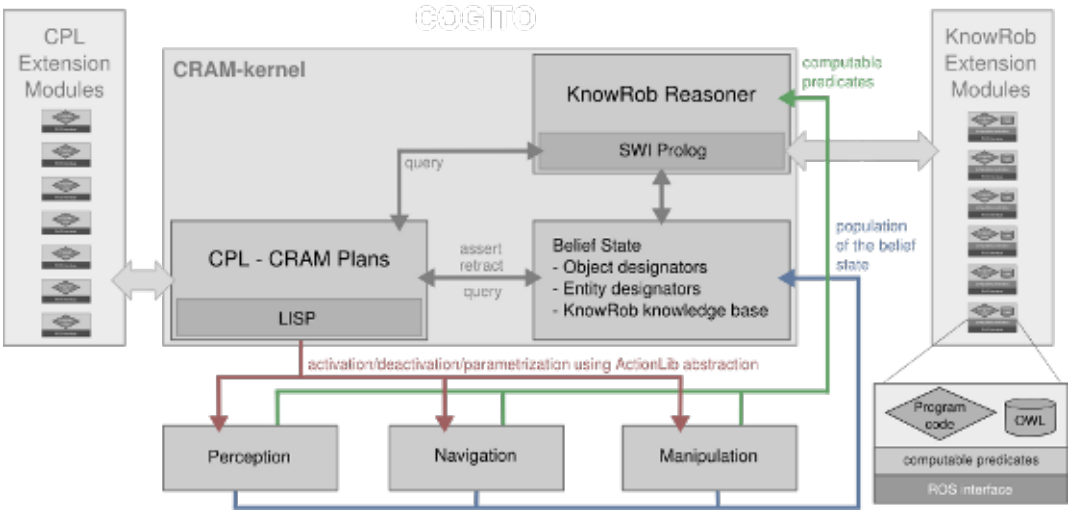
Introduced in 2010 by Willow Garage (https://robotsguide.com/robots/pr2), the PR2 stands as an advanced research robot. Boasting multiple joints and 20 degrees of freedom, this robot excels in autonomous navigation and the manipulation of a diverse array of objects, making it an ideal choice for our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



PyCram

VON PYCRAM: PyCRAM is a toolbox for designing, implementing and deploying software on autonomous robots. The framework provides various tools and libraries for aiding in robot software development as well as geometric reasoning and fast simulation mechanisms to develop cognition-enabled control programs that achieve high levels of robot autonomy. PyCRAM is developed in Python with support for the ROS middleware which is used for communication with different software components as well as the robot.

VON CRAM: CRAM (Cognitive Robot Abstract Machine) is a software toolbox for the design, the implementation, and the deployment of cognition-enabled autonomous robots performing everyday manipulation activities. CRAM equips autonomous robots with lightweight reasoning mechanisms that can infer control decisions rather than requiring the decisions to be pre-programmed. This way CRAM-programmed autonomous robots are much more flexible, reliable, and general than control programs that lack such cognitive capabilities. CRAM does not require the whole domain to be stated explicitly in an abstract knowledge base. Rather, it grounds symbolic expressions in the knowledge representation into the perception and actuation routines and into the essential data structures of the control programs.

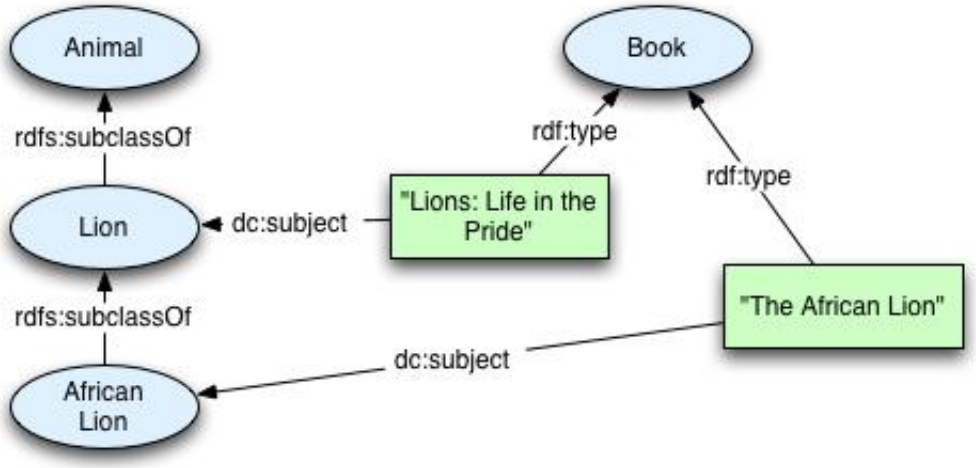


Knowledge Section: Ontologies and Rules

The parameters inferred for various robot actions come from a knowledge base. In the following, the principle of an ontology, as well as the concept of rules, which play a crucial role in parameter inference, will be introduced.

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.

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.

Libraries

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.

WikiHow

WikiHow is an extraction tool, that can collect informations from the WikiHow Corpus. The goal of this tool is to analyse a WikiHow corpus using basic NLP techniques to gather information about Everyday tasks like "Pouring", "Cuttingör "Discarding". These information should support cognitive robots in understanding and parameterizing these tasks to better handle unknown tasks, working in underspecified environments and handling common task-object combinations. Jedes Verb hat eine eigene Klasse, worin das Verb und die zusätzlich gewünschten Hyponyme/Synonyme definiert sind. Diese Verben dienen das als Schlüsselwort, wonach in den WikiHow Artikeln letzendlich gesucht wird. Außerdem kann man verschiedene Parameter einstellen, wie zum Beispiel das Ausschließen verschiedener Kategorien, welche für die Suche nicht von Relevanz sind.

Data acquisition

ToDo:

- Aus deutsch ins englische übersetzen.
- Beim vorletzten Punkt, die Zutaten Tools und Container besser beschreiben.
- Tabelle der Videos vervollständigen.
- Code Bild um Stir vervollständigen.

A part of this master's thesis involves creating a knowledge base that includes certain queryable parameters, allowing the robot to perform specific actions. The initial step in building the knowledge base is to gather data. In this chapter, we elaborate on our data acquisition strategy.

Task variations

As our main focus is to represent the knowledge about mixing, first we had to acquire the different types and variations of mixing in order to create a complete Knowldegerepresentation. 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. But first we have to ask ourselves What is Mixing?

What is Mixing?

The definition provided by the Oxford Dictionary is as follows: To put together or combine (two or more substances or things) so that the constituents or particles of each are interspersed or diffused more or less evenly among those of the rest; to unite (one or more substances or things) in this manner with another or others; to make a mixture of, to mingle, blend. Ultimately, this definition conveys that mixing requires at least two elements or substances, which are then combined (evenly) with each other, resulting in a (new) substance. This definition is general and can be applied to various contexts. For our work, the aspect of cooking or mixing different (cooking) ingredients is important. Therefore, we

consider some hyponyms of the verb "mixingïrrelevant for our cause and do not take them into account. An adapted definition for our work could be: Mixing is the combination of various (cooking) ingredients through different motions in a container.

Mixing hyponyms analysis

Hyponyms are subordered words of a given word, for example one hyponym of mixing could be beating. To conduct this analysis, one can utilize tools from various websites, such as FrameNet and WordNet. These platforms provide users with the ability to search for specific words and obtain various associations for those words, including synonyms, acronyms, or, crucial for our case, hyponyms.

WikiHow extraction

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. Damit wir die Suche ausführen können definieren wir eine neue Klasse "MixingVerb", welche das Verb Mix, sowie die von uns gefundenen hyponyme und Synonyme enthält.

```
MIX( verb: "mix", past: "mixed", participle: "mixing"),
no usages
AMALGAMATE( verb: "amalgamate", past: "amalgamated", participle: "amalgamating"),
no usages
BLEND( verb: "blend", past: "blent", participle: "blending"),
no usages
COALESCE( verb: "coalesce", past: "coalesced", participle: "coalescing"),
no usages
COMBINE( verb: "combine", past: "combined", participle: "combining"),
no usages
COMMINGLE( verb: "commingle", past: "commingled", participle: "commingling"),
no usages
COMPOUND( verb: "compound", past: "compouded", participle: "compounding"),
no usages
CONFLATE( verb: "conflate", past: "conflated", participle: "consolidated"),
no usages
FOLD( verb: "fold", past: "folded", participle: "folding"),
no usages
FUSE( verb: "fuse", past: "fused", participle: "fusing"),
no usages
INTERMIX( verb: "intermix", past: "intermixed", participle: "intermixing"),

JOIN( verb: "join", past: "joined", participle: "joining"),
no usages
JUMBLE( verb: "jumble", past: "jumbled", participle: "jumbling"),
no usages
LUMP( verb: "lump", past: "lumped", participle: "lumping"),
no usages
MERGE( verb: "merge", past: "merged", participle: "merging"),
no usages
PAIR( verb: "pair", past: "paired", participle: "pairing"),
no usages
UNIFY( verb: "unify", past: "unified", participle: "unifying"),
no usages
UNITE( verb: "unite", past: "united", participle: "uniting"),
no usages
BEAT( verb: "beat", past: "beat", participle: "beating"),
no usages
WHIP( verb: "whip", past: "whipped", participle: "whipping"),
no usages
WHISK( verb: "whisk", past: "whisked", participle: "whisking");
```

IN DEM BILD FEHLT STIRRING Da wir nun die gewünschten Verben, wonach wir

letzendlich suchen möchten definiert haben, passen wir einige Suchparameter für die Suche an. Die wichtigen Parameter sind das Filtern der Kategorien, in unserem Fall möchten wir uns auf das Mixen in dem Kochbereich fokussieren, deshalb filtern wir alle Artikel, welche nicht in dem Bereich Food and Entertaining existieren, aus.

Hyponyms occurance

Für jedes definierte Verb, wird eine Suche gestartet, wie oft dieses Verb in den WikiHow Artikeln vorkommt, dadurch wollen wir in Erfahrung bringen, welche Verben letzendlich für unsere Implentierung relevant sind und welche wir ausschließen können, da sie selten vorkommen im alltäglichen Gebrauch. In the table below the results can be seen.

Hyponym	**Occurance**
Mix	5300
Amalgamate	0
Beat	956
Blend	1041
Coalesce	1
Combining	3591
Coommingle	0
Compound	0
Conflate	0
Folding	821
Fuse	17
Intermix	0
Join	53
Jumble	0
Lump	7
Merge	6
Pair	352
Stir	6027
Unify	2
Unite	2
Whip	863
Whisk	2267

Tabelle 1: Mix synoyms/hyponyms occurance

Further examination and conclusion

Diese Suche soll uns letzendlich Auskunft darüber geben, welche Verben wir in der Wissensbasis repräsentieren wollen, daher entscheiden wir uns für gewisse Verben unter 2

Bedingungen, Häufigkeit und Ausführbarkeit. Die erste Bedingung ist leicht zu verstehen, Verben die gar nicht oder so gut wie gar nicht vorkommen, werden nicht in Betracht gezogen. Die zweite Bedingung bezieht sich auf die Ausführbarkeit des Verbs im Kontext von Roboternbewegungen. Außerdem werden einige Verben mit relativ hoher Häufigkeit auch etwas genauer untersucht, da im Englischen der past tense auch als adjektiv benutzt wird unter gewissen Umständen. Unter Berücksichtigung der ersten Bedingung werden folgende Verben nicht in Betracht gezogen: Amalgamate, Coalesce, Comingle, Compound, Conflate, Fuse, Intermix, Join, Jumble, Lump, Merge, Unify und Unite. Die zweite Bedingung schließt ein weiteres Verb aus: Blend. Blend wird meistens nur im Kontext mit einer Blendmaschine benutzt, welches vom Roboter nicht gehandhabt wird. Ohne diese Maschine kann die Blend-Ausführung nicht richtig durchgeführt werden und somit ist für uns dieses Verb ausgeschlossen. Unter genauer Betrachtung werden wir auch das Verb Whip nicht in Betracht ziehen, da dieses meist als adjektiv von Zutaten genutzt wird, wie zum Beispiel whipped Cream. Dies stellt heraus, dass nur das Verb Whip eine relativ gerine Häufigkeit hat. Das selbe gilt für pair, wo das past tense dazu genutzt wird eine Verbindung von verschiedenen Zutaten zu beschreiben, Wine paired with cheese.

Somit sind die von uns in Betracht gezogenen Verben: Mix, Combine, Beat, Fold, Stir und Whisk.

Task analysis and defintion

Da wir nun die Tasks ausgewählt haben, müssen diese Tasks anaylsiert werden und geschaut werden in welchem Kontext sie letzendlich genutzt werden. Unser Ziel ist es, dass der Roboter diese Tasks so ausführen kann, wie ein Mensch es tun würde. Um dies zu bewerkstelligen müssen wir im nächsten Schritt diese Tasks näher untersuchen. Dazu empfiehlt es sich Videos auf WikiHow oder andere Quellen zu analysieren, wo diese Tasks als Aufgabe vorkommen. Die Analyse besteht daraus zu gucken mit welchen Bewegungen die Tasks in Verbindung gebracht werden. Diese Analysen stellen wir tabellarisch im Folgenden dar. Untersucht wird die Task an sich, die jeweiligen Zutaten die verarbeitet werden, die Tools die dafür genutzt werden, sowie der Behälter worin die Task ausgeführt wird.

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 2: Video analysis

NOCH UNVOLLSTÄDNIG

Nach einer genauen Analyse der Videos und der in den Videos bereitgestellten Informationen, schlussfolgern wir, dass die ausgeführte Bewegung nicht nur im Zusammenhang mit dem Task steht, sondern auch mit den genutzten Zutaten. Einige Tasks sind allerdings determinierend, im Sinne dass sie schon ohne unbedingt auf die Zutaten zu achten, die Bewegung zwingend ausgeführt wird. Im Folgenden wollen wir extrahierten Informationen aus den Videos strukturieren und unsere Ergebnisse vorstellen

Video analysis conclusion and results

Basierend auf den extrahierten Informaitonen schlussfolgern wir dass für unsere Ziele folgende Aspekte, neben der Task, wichtig sind und weiter betrachtet werden:

- Ingredients: Die Zutaten spielen bei der Bewegungsentscheidung im Zusammenhang mit den Tasks eine wichtige Rolle und werden genauer definiert.

- Tools: Die Tools die genutzt werden sind bei der Bewegungsentscheidung nicht unbedingt ausschlaggebend, diese sind jedoch wichtig für die Paramter des Roboters????

- Container: Das selbe wie oben.

- Motions: Die Motions stellen letzendlich die Bewegung des Roboters für unsere Implentierung dar. Diese Bewegungen werden aus den Videos extrahiert und auf Roboternbewegungen abstimmend definiert.

Ingredients

Quellen:

- https://ceur-ws.org/Vol-2028/paper28.pdf

- https://www.freshfarm.org/app/uploads/2020/05/Mixing-Measuring-Wet-and-Dry-Ingredients.pdf

- https://www.cookingforengineers.com/article/280/Analyzing-a-Baking-Recipe/print

Durch die Videoanalyse gelangen wir zu der Schlussfolgerung, dass die Zutaten wichtig sind gegeben dessen Art. Hauptsächlich können Zutaten in zwei Hauptkategorien unterteilt werden, Dry and Wet. Diese Unterteilung wird auch wichtig hinsichtlich der Messung der Mengen der gegebenen Zutatenart. Für unseren Fall ist eine etwas feinere Art der Unterteilung der Zutaten notwendig, um die verschiedenen Bewegungen vernünftig auf die gegebenen Arten abzubilden. Wir abstrahieren neben der Wet Ingredients, eine neue Art: Liquid, diese entspricht den Zutaten, die unter den Wet Ingredients fallen, aber dessen Aggregatzustand flüssig ist, wie Milch, Wasser und Öl. Dies ist wichtig, da einige Bewegungen sich unterscheiden, wenn die gegebenen Zutaten Wet oder Liquid sind. Eine weitere Kategorie, die wir einfügen ist die Solid Kategorie. Diese unterscheidet sich zu der Dry Kategorie, darin, dass sie aus festen Komponenten besteht, während wir die Dry Kategorie so definieren, dass dort Pulverartige Substanzen unterklassifiziert werden. Die Solid Kategorie beinhaltet Nahrung wie Gemüse, Obst, Fleisch. Durch diese Unterscheidungen können wir eine Definition erstellen, wobei die Ingredientkategorien etwas verfeinert werden.

Definition: Zutaten werden Hauptsächlich in "Dryünd "Wetünterteilt, wobei eine feinere Differenzierung notwendig ist. Eine abstrahierte Kategorie namens "Liquid"wird eingeführt, um flüssige Zustände wie Milch und Wasser zu erfassen. Eine zusätzliche Kategorie, SSolid", differenziert sich von "Dry"durch feste Bestandteile wie Gemüse und Fleisch. Diese Unterscheidungen ermöglichen eine präzise Definition der Zutaten, verbessern die Analyse kulinarischer Bewegungsabläufe und deren Darstellung.

In erster Version von uns definierten Zutaten sind:

- Liquid: Milk, Oil, Water, Vinegar, Vanilla Extract, Sauces.

- Wet: Egg White, Egg yolk, Butter, Whipped Cream, (mashed fruits?).

- Dry: Flour, Salt, Sugar, Baking Soda, Cocoa Powder.
- Solid: Onions, Pork, Chicken, Minced meat, Bacon.

Hier eventuell nochmal begründen wieso wir die nehmen??.

1. Ansatz: Zutaten aus den Videoanylysen.

2. Ansatz WikiHowExtraction.

Tools and Container

Das kommt von Soma, muss man schauen wie man das beschreiben möchten

Motions

From this videos we extracted informations about the executed motions. The following motions can be 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.

Data Acquision Conclusion

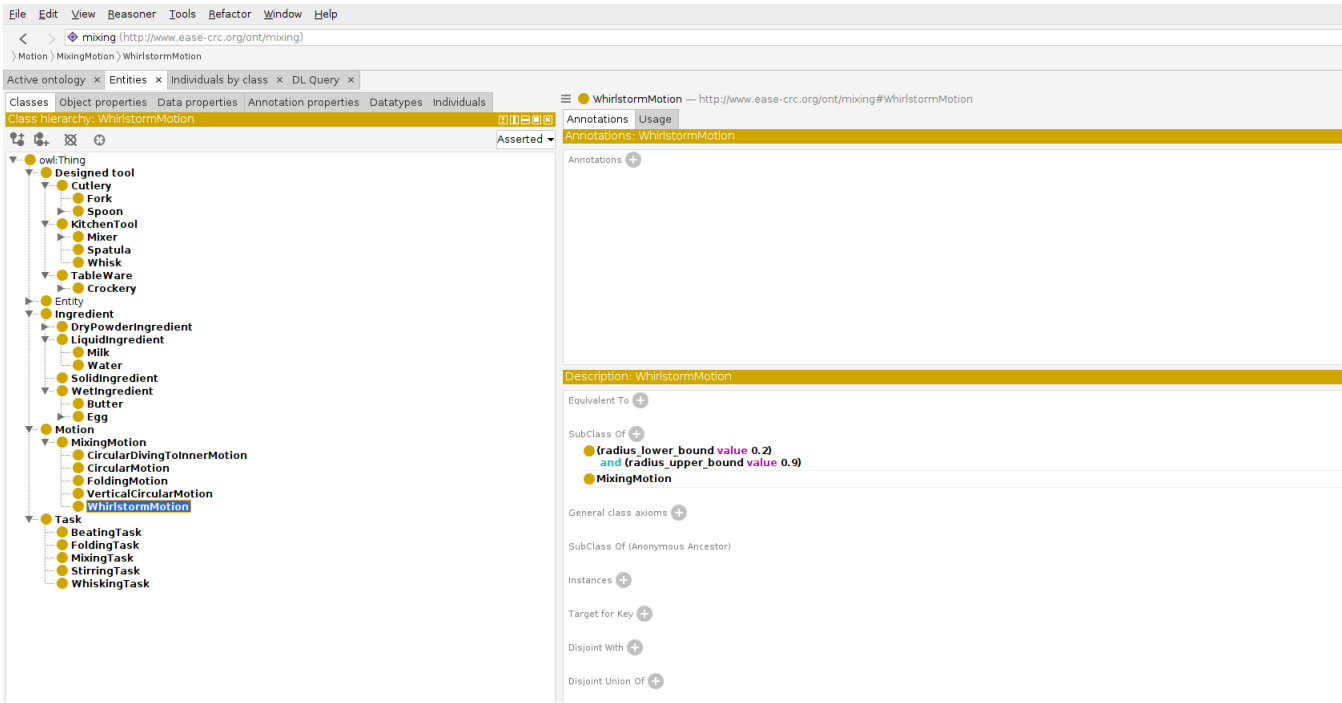
Bei der Datenacquise fokussierten wir uns darauf eine Grundlage der benötigen Daten und Informationen zu erschaffen, welche für einen Roboter benötigt werden verschiedene Mixing Tasks auszuführen. Nach der Feststellung der Tasks, konnten wir Videos analysieren um die Bewegungen zu untersuchen die in Zusammenhang mit den jeweiligen Tasks stehen. Dabei sind wir zu dem Entschluss gekommen, dass nicht nur die Task ausschlaggebend für die Bewegung sei, sondern auch die Zutatenauswahl. Die Zutaten werden in 4 verschiedenen Kategorien unterteilt und wir konnten 5 Motions definieren, welche vom Roboter so ausgeführt werden sollen, wie es im Alltag gebräuchlich ist. Im nächsten Kapitel wollen wir vorstellen, wie wir dieses Wissen nun repräsentieren und ein System erstellen, wodurch der Roboter in der Lage ist zu Wissen welche Bewegung es ausführen soll basierend auf einer gegebenen Task und Zutaten.

Data Representation

In this chapter we want to introduce our main ideology behind the Data Representation. Our main goal is to determine different motions for different given scenarios, while keeping it as simplified as possible. The main idea behind simplifying the decisions, is motivated through given situations in which the robot would be to dependent on multiple parameters in order to suceed and execute its motion. By simplifying these scenarios the robots decision will rely on less parameters thus the succes rate on making decisions will rise.

The Knowledgebase

We designed a Ontology in which we illustrated our needed classes. The Knowledgebase is divided upon the superclasses Ingredient, Tools, Tasks, Motions and Container.



Ingredients

The Ingredient has 4 subclasses: WetIngredients, SolidIngredients, LiquidIngredients and DryPowderIngredients. The WetIngredients are all the ingredients that can not be considered liquids but also not considered dry and solid, like (melted) butter or eggyolk. Liquid ingredients consists of liquids such as water and milk. Powder ingredients are the ingredients which are dry but also consists of small particles like sugar, salt and flour. Last we have the solid ingredients which consists of ingredients with solid material like vegetables, fruits and meat.

Tools and containers

The tools can be also divided in multiple categories. Cutlery consists of Fork and Spoon, while Spoon also has subclasses like a wooden spoon and tea spoon. Then we have kitchen tools where we can find different types of mixers and whisks. Last we got the superclass Crockery in which our containers we will be saved, like different bowls, pot and mugs.

Tasks

Different Tasks will be saved under the Task superclass. The tasks consists of Mixing, which can be regarded as the umbrela term of all tasks, Stirring which is mostly a task that includes a circular motion, Beating will be mostly used in context with eggs and other wet ingredients, Folding which represents a gentle type of mixing and the last task is whisking which is similar to the beating task. While some of these tasks can be regarded as deterministic when considering the motion decision, like the folding task will always rely on a folding motion, we discovered that the task itself is not as ausschlaggebend, but in combination with the ingredients, one can decide upon a better motion decision.

Motions

The motions are necessary for the robotic system to know what has to be done?. The motions are inferred from Rules that regards a task component, combined with ingredients. The motion also contain parameters, which should determine the moving space available for the robotic system. As in the mixing world, most of the container have a circular formed base, our motions are defined with a radius. The most important parameters are:

- Radius Lower Bound: This parameter describes the smallest possible radius for a motion. For example if the used container is a bowl, that has a radius of 10 cm, a radius lower bound of 0.1 would imply that the smallest possible radius on which the robot can perform its motion is 1 cm.

- Radius Upper Bound: Similar to the radius lower bound, the upper bound determines the maximum radius on which the robot can execute motions.

Our implemented motions are:

- Circular Motion: A circular motion is a motion defined on a circular with a constant radius. This motion basically sets the 2 parameters radius upper bound and radius lower bound on the same value.
  HIER BILD

- Whirlstorm Motion: The Whirlstorm motion covers multiple parts of the container, the motion starts on the center of the container and increments its radius until it reaches the radius upper bound parameter, before turning back to the center.
  HIER BILD

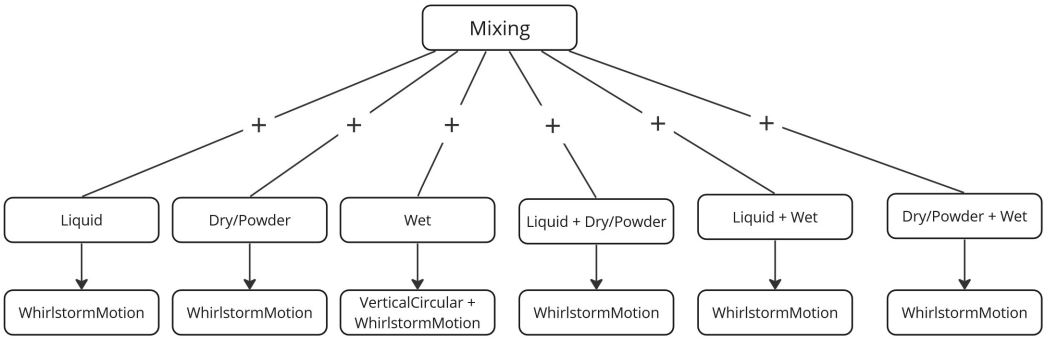
- Folding Motion: Folding is a special motion which is mostly used in the context of the folding task. The motion's goal is to mix the ingredients gently without overmixing the mixture. The motion starts on a point on the circle with the radius of the upper bound parameter, from there the motion draws a straight line to the middle point of the container, then getting back to the start point, next the tool will be moved 90 degrees on the defined circle and it moves back to the middle, this motion is repeteadet 4 times, and then you move the tool 20 degrees to cover all of the ingredients before starting the cycle again.
  HIER BILD

- Vertical Circular Motion: The last motion can also be seen as the most difficult one. The main idea of this motion is to 'wildly' mix the ingredients. Beschreibung fehlt, Bild hier rein.

Rules

In order to infer the right motion based on the ingredients and task input, rules have to be defined. This rules are written in SWRL, which was presented in the USEDLIBRARIES section. In this section we want to illustrate the inference on a high level, while the implementation of it will be shown in the implementation chapter. The rules can be thoguht was if conditions, which will then result in a motion. For example, if we regard the combination of the task Mixing and the ingredient type liquid, we infer the motion Whirlstormmotion. This can be done for every task and ingredient combination. The inference can be illustrated with decision trees which will be shown in the next images for every task and ingredient type combination available. Hier die decision trees?.

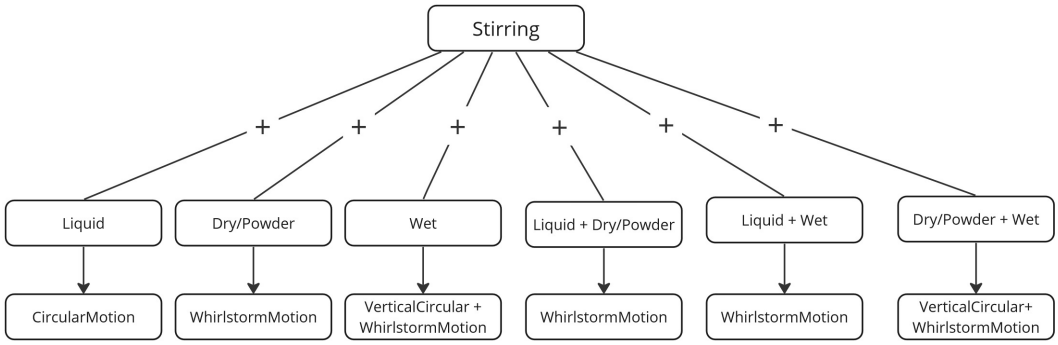
Mixing



Definition: In the context of baking or cooking, a mixing task refers to the process of combining multiple ingredients thoroughly to create a homogeneous mixture. The goal is to distribute the ingredients evenly, ensuring that each component contributes to the overall texture, flavor, and consistency of the final dish or baked good. Mixing is a fundamental step in many recipes and is essential for achieving a balanced and cohesive result.

Wie man anhand der Abbildung erkennen kann, kann die Mixing Task hauptsächlich auf die Whirlstormmotion abgebildet werden. Dies wundert letzendlich nicht, wenn man sich die Definion anschaut, denn diese Motion führt dazu, dass alle Zutaten in dem Behälter gleichmäßig verteilt werden.

Stirring

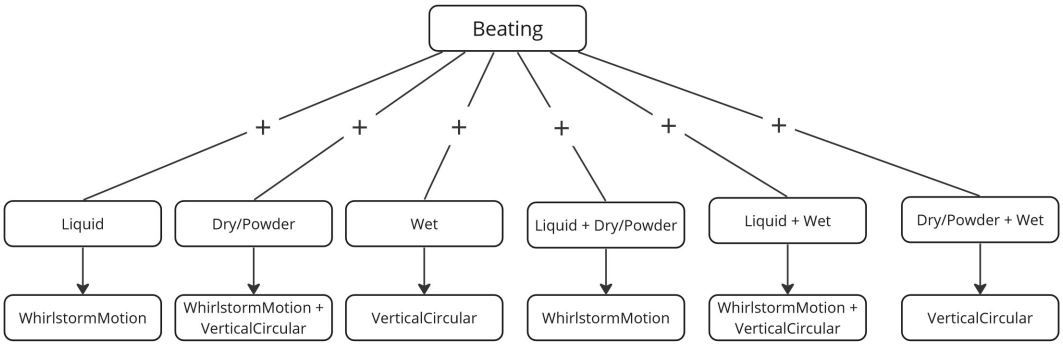


Definition: In the context of baking or cooking, a stirring task involves using a utensil, such as a spoon, spatula, or whisk, to agitate and circulate the ingredients within a mixture. The purpose of stirring is to achieve a uniform distribution of ingredients

Im Vergleich zu Mixing bildet die Stirring Task abhängig von den Zutaten auf eine breitere Anzahl von Motions ab. Neben der Whirlstormmotion wird hier zum ersten Mal die

CircularMotion genutzt. Diese ist besonders wichtig in Anbetracht der Stirring Task mit dem Ingredienttyp Liquid, da man nicht möchte, wie bei der Whirlstormmotion der Fall wäre, dass die Zutaten äufgerührt"werden, sondern ledigilich umgerührt.

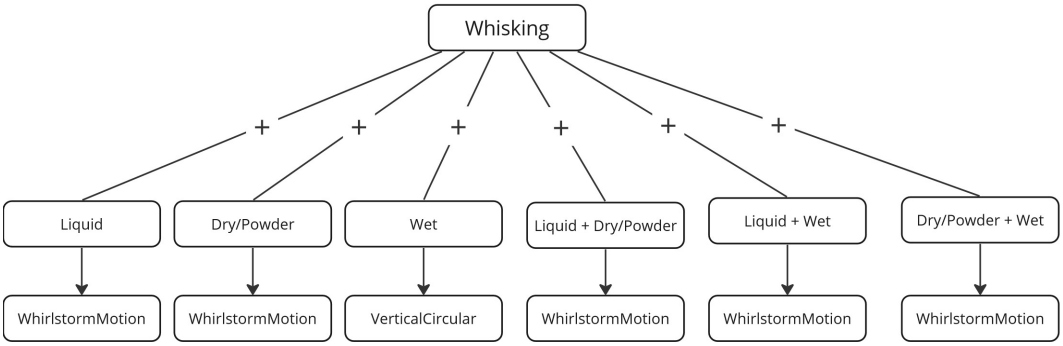
Beating



Definition: In the context of baking and cooking, a "beating"task refers to the process of vigorously stirring or mixing ingredients to achieve a specific texture or consistency. Beating is often done to incorporate air into the mixture, create smooth and uniform blends, or alter the physical properties of certain ingredients.

Neben der WhirlstormMotion prädominiert hier auch die VerticalCircular Motion, dies ist auch aus der Definition abzuleiten, da diese Motion eine willkürliche Art des Mixen erfordert, worauf die VerticalCircular Motion passt.

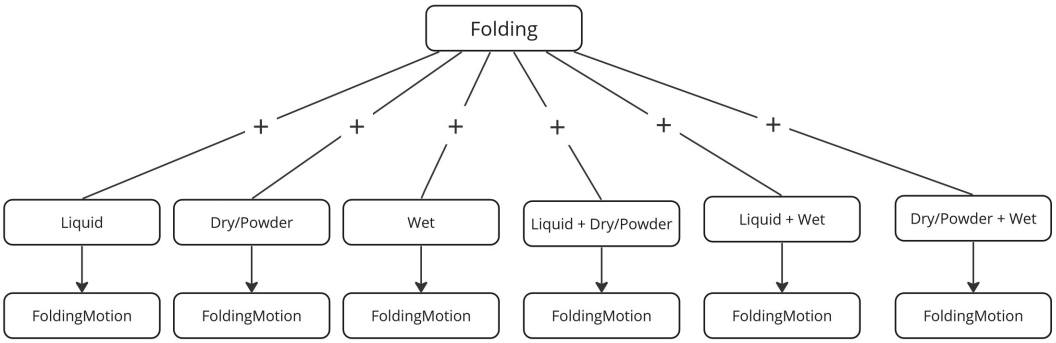
Whisking



Definion: In the context of baking and cooking, a "whisking"task involves using a kitchen utensil called a whisk to mix, blend, or beat ingredients. A whisk typically consists of wire

loops or a coil attached to a handle, and it is designed to incorporate air into mixtures, break up clumps, and create a smooth and uniform texture.

Folding



Definition: In the context of baking and cooking, a "folding"task refers to a gentle mixing technique used to incorporate ingredients without deflating or destroying the air bubbles that have been created. Folding is often employed when combining a lighter mixture (such as whipped cream or beaten egg whites) with a denser one (such as a batter or a heavier mixture). The goal is to maintain the desired texture, lightness, or fluffiness in the final dish.

Theoretisch müssten wir die Folding Task nicht grafisch darstellen, da jede Task und Ingredients Kombination auf eine einzige Motion abbildet, die FoldingMotion. Die Grafik wurde trotzdem der vervollstädnigkeitshalber eingefügt. Die FoldingTask erfodert eine spezifische Bewegung, die die Luft in der Mixtur nicht entfernt, wobei jede andere Bewegung scheitern würde.

How to expand the Knowledgebase

Simulation

In diesem Kapitel stellen wir die Simulation vor, indem der Agent die definierten Motions ausführen kann und dadurch als Proof of Concept dient. Zunächst werden wir die Simulationsumgebung Bulletworld vorstellen, gefolgt von einigen Beispielsaktionen die dort ausgeführt werden können. Anschließend zeigen wir, wie die Parameter über Queries inferriert werden und mit der Visulasierung der Motions zeigen wir, dass die von uns definierten Motions + Parameter vom Roboter in der Simulation ausgeführt werden können.

Simulation Environment

Die Simulationsumgebung ist Bulletworld, welche als Simulationsumgebung für das Framework PyCram dient. Dies ist sehr günstig, denn somit können die in PyCram definierten Aktionen direkt simuliert werden und müssen nicht extra geportet werden. Als Schnittstelle zu den Roboter wird ROS1 genutzt, um mit den Joints zu kommunizieren. Dies geschieht über RosNodes, darüber kann man Informationen über den Zustand der Joints (oder andere Komponente) erhalten, sowie mit diesen Komponenten kommunizieren um Aktionen zu befehlen.

Der für unsere Zwecken genutzten Roboter ist der PR2, dieser ist als Modell schon in der Bulletworld implementiert und ist für unser Fall geeignet, da für die Motions, 2 Arme benötigt werden, eins um die jeweiligen Container zu halten und den anderen Arm, um das Tool, welches für die Aktionen genutzt wird, bewegt wird.

Die Umgebung, welche für unsere definierten Aktionen genutzt wird, ist eine Küche, die Möbel besteht aus einem Tisch, worauf die genutzten Container platziert werden und die Motions dementsprechend ausgeführt werden, sowie weitere Küchenmöbel, welche für unsere Fälle nicht relevant sind. BILDER KÜCHESIMULATION

Die von uns genutzten Objekte sind:

- Container: Bowl (klein und groß), Pfanne, Topf und Tasse. BILDER
- Tools: Whisk, Löffel (klein und groß), Holzlöffel. BILDER

PyCram bietet schon eine betrachtliche Anzahl an implementierten Aktionen, womit der Roboter manevriert werden kann. Unter diesen Aktionen befinden sich notwendige Na-

vigationaktionen, sowie manipulative Aktionen wie Greifen und Platzieren. Außerdem bietet PyCram schon Schnittstellen bereit womit die Joints des Roboters bewegt werden können, diese Funktionen heißen dann zum Beispiel moveTorso(). BILDER CODE FUNKtIONEN

HIER STELLEN WIR UNSERE MOTIONS VOR

Hier Vanessa fragen, wie wir ihre Motion referenzieren sollen, einfahc sagen es war vorhanden oder es detaillierten angeben?

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.

Motions

In this chapter we'll describe on an abstract level how the motions are implemented in pycram2. To simplify the implementation of motions, we hold the following assumptions about the motions which are performed on containers. Firstly we assume, the container is symmetric in the x,y direction. We don't consider the descent of the container, we keep the height for the tool constant over the performed motion. We don't have collision detection.

Circular Motion

Be $container\_center = (container\_center\_x, container\_center\_y)$ Generating points lying on a circle: $x\_coordinates = container\_center + radius\_bounds * cos(radian)$ $y\_coordinates = container\_center + radius\_bounds * sin(radian)$ Where an array of angle values is created and converted to radian values.

INSERT Plot

0.0.1 Folding Motion

Be l an array of x,y coordinates lying on a line, where the starting point of the line is not the center of the container but rather its endpoint. The line is turned via 2D transformation via the rotation matrix:

$$R(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$$

The transformation $\begin{bmatrix} x' \\ y' \end{bmatrix} = R(\theta) * \begin{bmatrix} x \\ y \end{bmatrix}$, where $\theta = 90$ in degrees.

This results in the tool being moved 90 degrees from the containers center point, to perform on another area inside the container. The folding motion is repeated 4 times.

Afterwards we transform the line $\begin{bmatrix} x' \\ y' \end{bmatrix} = R(\theta) * \begin{bmatrix} x \\ y \end{bmatrix}$, where $\theta = 22.5$ in degrees, to cover a similar area of the container.

INSERT Plot

Vertical Circular Motion

The vertical circular motion is implemented in the following way. By using the formular to compute points on a circle, we adjust the radius for x,y, where one of the values is set to 0.01, the other one is sampled evenly from the interval $semi\_major\_radius = (radius\_upper\_bound, radius\_upper\_bound/2)$. We take the maximum sampled radius which fulfills the following condition. $\sqrt{(x\_i - center\_x)^2 + (y\_i - center\_y)^2} < radius_upper_bound$ Additionally we decrement and increment y coordinates. If the condition: $\sqrt{(y\_i - center\_y)^2} > radius_upper_bound$ is met we change from decrementing to incrementing and vice versa until this condition is fulfilled again.

Whirlstorm Motion

Using the formular to generate coordinates lying inside a circle we sample radiuses from the interval $radiuses = (radius\_upper\_bound, radius\_lower\_bound)$ This causes the motion to go from outer part of the container to the inner.

Implementation

- OWLReady
- SWRL
- Inferring parameters for the actual plan

Evaluation

- Evaluation of multiple motions multiple times
- Results of that

Summary / Fazit

avc