MAMPF: An Intelligent Cooking Agent for Zoneless Stoves

Sven Reichel, Timo Müller, Oliver Stamm, Fabian Groh, Björn Wiedersheim, and Michael Weber
Ulm University,
Institute of Media Informatics,
James-Franck-Ring, 89081 Ulm, Germany
email: mampf@uni-ulm.de
video: http://vimeo.com/19938951

***Abstract*—The Multifunctional and Adaptive Meal Preparation Facility (MAMPF) is a versatile and adaptive kitchen system allowing even an inexperienced user to create ambitious dishes. On the hardware side, the system controls hotplates by regulating the temperature of heated regions to turn the attention to the food actually being heated instead of the hotplate. Furthermore, the user interaction with the computer system is adapted to the requirements in a kitchen. On the work flow side, MAMPF converts a recipe into a task tree which is used to guide the user through the cooking process. By knowing the recipe, based on a formal description specially developed for this system, the optimal sequence of actions can be calculated.**

The system is designed to support the user without limiting spontaneity or creativity and can - if necessary - be used just like any usual stove. Consequentially, the user always retains control over the system, all settings can be overwritten at any point, and the order of the tasks can be dynamically rearranged.

In order to prove the correctness of this concept, a prototype has been implemented.

***Index Terms*—Smart kitchen, intelligent environment, cooking support.**

I. Introduction

The kitchen is still one of the most traditional places in our modern society. Handwritten recipes and kitchen-secrets are shared within families and are passed on from one generation to another. Professional cooks prepare meals you would not be able to imagine, rather than being able to reproduce them. Those meals may not only catch others' attention but even attract and emotionally move them. MAMPF can even assist beginner chefs with their cooking careers by adjusting the level of detail used in the cooking instructions and thus replace a cookery course.

Considering that many parts of our daily activities are rapidly changing and being digitalized, cooking still is like it was centuries ago. Of course there are new machines that help ease the act of cooking in various ways but the act of combining dozens of ingredients into a delicious meal is still something which requires a lot of skill and experience. There are many ways intelligent and ubiquitous computer systems can help to ease the preparation of food, making it possible for everyone to cook a large variety of dishes.

A system offering recipes and supporting the user lowers the barriers of cooking unfamiliar dishes. Preparing a meal instead of reheating frozen food or eating at the next fast food restaurant may lead to a more healthy diet. By the selection of ingredients based on the user's preferences it is possible to cook a larger variety of meals with less effort. However, cooking itself does not have to be reinvented. Online communities sharing recipes in combination with an intelligent kitchen could make this vision come true.

MAMPF is a concept showing that ubiquitous computing methods integrated into a kitchen are an effective way to simplify the cooking process. It is the result of analyzing the way of meal preparation and identifying steps, tasks, and methods which can be extended and supported by computer interaction and automation.

Thus, we convert the recipes into a machine-readable form, which allows classification, sharing, rating, and comfortable modification. After choosing the desired recipe, the instructions are displayed on a screen or explained via voice. We invented a system that relieves the user of all tasks which can be automated. Furthermore, it monitors the user in order to recognize actions and purposes allowing him to work freely without restrictions. For example, bringing water to boil usually begins with filling a pot with water followed by putting the pot onto an appropriate hotplate and adjusting the desired heat level. Our concept reduces the explicit input in such a way that the area covered by the pot is recognized and temperature is automatically regulated.

First of all, we put our work in context to already existing systems and ideas. After that we present a redesign of user interaction in Section III, followed by theoretical aspects of cooking needed to supply an intelligent cooking agent. Section V shows a prototypical implementation enhanced by a recipe guidance with modified induction cookers which are controlled automatically according to a recipe. MAMPF is only a first step to provide a rich, user supporting, intelligent environment, thus we give a short prospect how improvements can further automate the cooking process.

II. Related Work

Our work can be roughly classified into three different domains: the hardware with induction hot plates, cooking theory including specification of recipes, and intelligent agent-systems for user interaction. Various works have been considered for each of these categories.

The stove represents the essential part of the hardware domain. Currently, more and more nearly-zoneless stoves with multiple induction plates are proposed, as described by Forest [5] or in "Rendez-vous", a commercial concept by AEG [4]. Furthermore, the control mechanisms are undergoing significant changes as AEG already uses touch screen interaction [1]. We expand the stove control by replacing it with an automation based camera system for pot tracking. To identify the cooking utensils a combination of template matching [19] and SURF [3], [9] is used.

More and more kitchens are going to be equipped with sensors, actors, and intelligent systems. Those systems are able to analyze the context, monitor the users actions, and adapt the cooking processes if necessary [17]. Different agent systems have been developed to display recipe information to the chef [7], [18], [11]. We extend these approaches by not only presenting the cooking process but automatically adjusting the stove and subsequently propose a dynamic order of the actions to prepare a meal.

To be processed by a computer system, the recipe has to be based on a formal specification, like suggested in [6]. Nevertheless, we aim to keep it as simple as possible and analyzed cooking sequences with the help of [12], [15] and [11]. Thus, we can not only specify a recipe but present an elaborate representation which determines dependencies of different actions, ingredients and cooking utensils.

Object detection, recognition, and tracking with the aid of cameras is already wide-spread and different methods are established [9]. Camera independent solutions are another possibility as induction cookers can sense the presence of a magnetic vessel [21], [16].

III. Redesign of user interaction

Heating a pot is commonly associated with lighting up a fireplace or switching on a hotplate. In both cases the interaction is bound to that specific spot which emits heat and a pot needs to be put on top.

The technical possibilities have been developing and, instead of primitive fireplaces or standard stoves, high-tech kitchens containing multiple sensors and actors are available. Combined with some computing power, the different facilities of modern user interaction can be implemented and enable a new cooking experience.

Once the cooking process has been analyzed and split up in disjoint actions and items (see Section IV-A) the steps can be mapped to new ways of interaction like speech recognition or touch-panels. The latter are currently finding their way into consumer induction stoves.

Beside the different controlling options, the stove itself can be modified as well. Nowadays, stoves mostly consist of four explicitly defined hotplates. Breaking up this structure and thereby the forced interaction with the hotplate was one of the main research goals. As we regard cooking as a learned ability, we assume that developments and changes concerning the control of a stove will be adopted by users.

In the following the new approach is presented. The advantages of a zoneless stove lead to the idea of implicit interaction. Finally, we suggest methods of modern interaction which allow explicit input where implicitness might not work.

A. Zoneless cooking

The current standard stoves impose the following major restrictions:

- The number of hotplates limits the number of vessels which can be heated at the same time.
- Positioning vessels freely on the stove is not possible.
- Moving a vessel from one hotplate to another means manually changing the heating settings.

The first and the second problem are correlated and can be solved by using multiple modular hotplates which then - aligned like honeycombs - can be used to cover the whole stove [5]. With this technique, vessels may have unrestricted movement and thus will always be heated in an optimal way as heat is emitted across their entire footprint. Still, the hotplates themselves have to be activated and deactivated. Handling such a stove with usual switches would be unacceptably complex as there are too many hotplates to control and no specific classified regions. As a result, the vessels on the stove stay under control of the user while covered hotplates are automatically recognized and activated by the system. Thus, a relocation of a pot or pan triggers a change of hotplates being used to heat that vessel. As a result, all hotplates covered by pots and pans might be heated while the others stay cold.

It is necessary to identify and track the vessels and map pot positions to the hotplates underneath. Object detection, recognition, and tracking with the aid of cameras is already wide-spread and different methods are established [9]. Other solutions take advantage of the fact that the power consumption of the hotplate depends on the cooking utensils [21], [16]. However, identifying and tracking a vessel using only one monitored value is more complex and other local information is difficult to extract. Due to high temperatures and effects from induction hotplates the use of electronic devices, bar codes or other kinds of tags to identify different cooking utilities is very problematic. Tags made of plastic or paper might start to burn while RFID tags and readers are additionally affected by electromagnetic fields which disturb the communication or even destroy electrical parts. So there is a need to recognize the cooking tools from distance.

We consider pot detection and tracking via camera a useful and cheap solution, as a standard webcam is sufficient (see Section V-B). However, our concept works with resistance sensors as well which might be more elegant for commercial products as sensors are embedded and no calibration is needed.

B. Focusing the vessel and implicit interaction

Given a zoneless stove with detection and tracking capability, the computer is aware of the location and mapping information of each pot and is able to adjust its heat. With even more contextual information, and a set of rules, the user's

intention can be derived and thus implicit interaction may be achieved.

A system that is not only aware of what is being cooked and which ingredients and cooking utensils are used, but even knows the actual progress within the cooking process, opens up a further context of temporal-semantic quality. This context allows the software to derive the most probable interaction intended by the user, which is the basis for implicit interaction. I. e. when the recipe demands a pot to be heated up to a specific temperature the system triggers the object recognition to wait for a pot to be put on the stove. The placement of the pot on the stove is the only action the user has to execute. Afterwards the system controls the pot autonomously according to the recipe. As a result, the heat setting is associated to the pot instead of the hotplate, which allows the chef to concentrate on the pot and does not have to cope with the technical details of the stove.

Summing up, we facilitate the cooking process by removing the intermediate stage of explicit temperature control at the stove and instead enable the direct manipulation of cooking utensils.

The next step would be to extend the interaction with pots to the interaction with single ingredients. By recognizing them as described in Section VI-A this can be done implicitly as well. This would lead to the tracking of the entire cooking process beginning with the preparation phase with a minimum of explicit interaction with the system.

C. Necessity of explicit control

In any case, the cook needs to stay in control and has to be able to adjust the settings manually. Especially if recipes are faulty or cooking without any recipe is desired. Furthermore, the ability of applying changes to the cooking workflow is essential (e.g. to extend the heating time of a pot if the software turns it off too early).

Then again, the software might not detect a pot which has been put on the stove or might not be able to track an already detected pot anymore. Generally, this should not terminate the whole process. Therefore, explicit user input is necessary, i. e. to (re-)register the vessel.

However, in some other cases implicit interaction is not sufficient. Setting up the environment or even handling emergency cases like overboiling and burning require explicit input. Automatic detection might fail but control over the system has to be guaranteed. These inputs can be realized by standard point-and-click devices, touch panels or pads, and speech dialog systems. We suggest a context-sensitive speech recognition which is capable to interpret each command according to the current situation (see V-D) and allows hand-free control.

IV. Intelligent Cooking Agent

MAMPF is equipped with an intelligent agent which guides a user through the cooking process. The system gives clear advice and automates as much steps as possible - for example heating a pot on time according to the recipe. Furthermore, it tracks time and status information during the preparation and presents it to the user to achieve an overview of the actual cooking process. That is why the system needs to know each step in a recipe as well as ingredients, cooking utensils, and time information. In order to provide all this in a computer readable format, we developed a formal specification for recipes. The guidance always supports the users and does not restrict their abilities, so changes can be made dynamically. This prevents the failure of a recipe even if it contains a mistake or a discrepancy between the specified requirements and the actual resources. Additionally, if the user misses a cooking step the agent will indicate it, thus preventing the user from making mistakes.

A. Cooking theory

As described by David A. Mundie in "Computerized Cooking" [12], recipes contain a way to describe the preparation of a meal and can be expressed in a mathematical formula, but are usually authored in colloquial prose. This form is not compatible to computer systems so heuristics to parse prose or a special computer readable format for recipes is needed.

One major purpose of MAMPF is that it is easy to use, so a recipe should only contain essential information, making it as easy as possible for the user to specify. As described in Section II, other solutions like recipeML already exist, but according to our experiences they are too complicated for the users to work with. So we had to develop a new recipe specification for our purposes.

The cooking domain consists mainly of four parts: actions, food, recipes, and utensils [15]. At the beginning, meta information of a recipe like title, author, categorization, and number of people considered are specified. Secondly, it contains the needed ingredients and for each a certain amount and a unit, e.g. *500g Spaghetti*. Furthermore, to each ingredient a recipe unique identification number is added to obtain references between ingredients and a recipe's step. Moreover, the ingredients are treated by kitchen utensils which we call tools. Each tool has its own identification number. To express the recipe's workflow we use eight different kinds of steps. Each one references different numbers of tools and ingredients - this is expressed in brackets - e.g. (1/2) means this step needs one ingredient and two tools. Thus, we obtain context information of the step.

Here is a list of actions we distinguish, including the maximum amount of ingredients and tools involved:

- **PUT** ingredient into tool (1/1)
- **OUT** remove anything from tool (1/1)
- **JOIN** the content of two tools (0/2)
- **ON_STOVE** put a tool onto the stove (0/1)
- **OFF_STOVE** remove a tool from the stove (0/1)
- **DO_TASK** do a task (e.g. stir) (2/2)
- **DO_CONTENT** do a task with tools (0/2)

DO_TASK and DO_CONTENT have optional tools and ingredients as well as a required plain text field. This contains action information to explain the action to the cook, for example cutting, stirring, or mixing. It is not possible to predefine these actions since they are uncountable. There is no requirement

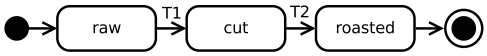

Fig. 1. Onion state machine

that the system needs to understand these actions, so they are kept is a plain text string. This also makes it possible for the user to define actions as precise or vague as they are needed, depending on the skill level of the user to whom the recipe is addressed.

As described in Section IV-C, each step contains its average preparation time. Additionally, it can have three different triggers, which describe the system's reaction to the step. They are defined in the following way:

- **TEMP** sets the temperature of a tool
- **TIME** starts a timer after which a message is displayed and the used tools are turned off
- **CONFIRMATION** displays a message to remind the user observing something and has to be confirmed when the described state is reached (e.g. onions are golden brown)

The cooking process is based on state machines of ingredients and tools. That means each of them has a different current state and transitions representing actions, leading via other states to the final result. Figure 1 shows a simple example of an onion's state machine which has three states and two transitions, though more states can be added, e.g. sliced, diced. As an action can perform a change of up to two ingredients and two tools, the confirmation of a step can trigger up to four transitions of different state machines.

The processing order of steps needs to respect each state machine of a tool and an ingredient, for example the onion in Figure 1 has to be cut before getting roasted. Therefore, a step can only be confirmed if all concerning state machines are in a state where each transition can effect a state change. This guarantees that no state of a state machine is skipped. For this purpose we use a dependency tree which expresses the dependencies of steps. The leaves of the tree reflect the raw ingredients, the root the final meal. To create this tree, the order of steps in a recipe's specification is essential. Based on that we construct the state machines and furthermore the connections of the tree. So each recipe will create one tree. Figure 2 shows a recipe with five steps. Step1 uses tool $t1$ and ingredients $i1$ and $i2$. As Step3 processes the same ingredient as Step1 ($i1$) and as Step2 ($i3$), it depends on these two steps and therefore can only be processed if Step1 and Step2 are confirmed. When cooking more than one recipe simultaneously, it is possible to connect the trees with a special join step in parallel or as a sequence - in that case we have a graph.

B. Processing

One major goal of our system is to keep things simple for the user and not to dictate the cooking process. That means the user can rearrange the tasks as far as they do not depend on others. By presenting all executable steps at the same

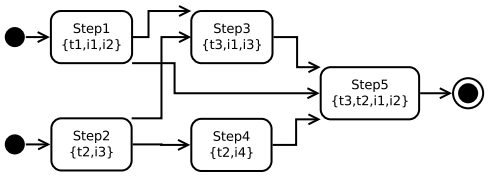

Fig. 2. Dependency tree of one recipe

time, the user might get confused. Therefore, we propose a sequential step list but allow scrolling the list to choose which step to perform next. The step with the highest priority is always highlighted (see Section V for the System's display possibilities). Each step is displayed until the user explicitly confirms the step. After that, all triggers are processed and the next step is highlighted. As a result, the dependency graph needs to be transposed into a sequential list. This is done according to the recipe's specification and can be optimized according to the preparation times derived from the user profile and the available cooking devices and tools. Thus, the idle time in a recipe can be minimized and the shortest preparation duration can be reached.

During the execution of a cooking workflow each step can have five different states:

- **waiting** not active, can be selected manually by the chef via the scrollable step list
- **pre-execution** check whether all dependent steps are in state *done*
- **execution** show actions to do, including used tools and ingredients. Currently, this is clarified by images and spoken language but a video or any other graphical help, as suggested in [18], may be possible.
- **post-execution** all triggers are executed, that means the temperature is set and timers are started
- **done** a step is confirmed and all TIME and CONFIRMATION triggers are finished

A user interaction is necessary to transfer a step from the state execution to the state post-execution. If there are any active triggers attached, the user needs to validate or cancel them explicitly. This can be done by touch, gestures, or speech but as mentioned in Section III also by implicit interactions. Furthermore, this confirmation can be done by recognizing the current cooking state with different kinds of sensors in the kitchen [11].

During the cooking process, the system measures time information and thus can predict the finishing time. Additionally, the user can always monitor the current process and active cooking tools. For example, in a short overview he sees which tool is heated for how long and which one needs special attention so that the ingredients do not get scorched. The tools on the stove have to be mapped to tools in a recipe, in order to display a label for each tool and preventing a mix up.

C. Learning

For reliable prediction of the preparation time and providing meaningful timing cues it is necessary to have detailed timing

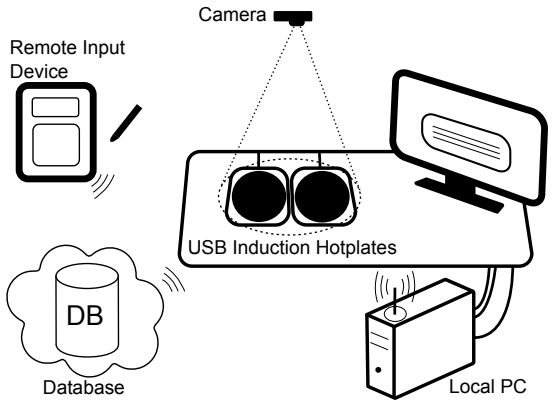

Fig. 3. MAMPF overview: How the components are connected to the system

information for each step. A cook's skill ranging from amateur to professional and also the equipment of a kitchen have great influence on the preparation times. It is necessary to measure the actual duration of each step and continuously update the overall duration. Additionally, we store the required cooking time for each user and adapt the predicted duration each time the recipe is executed according to following formula:

$$T_{new} = \frac{T_{old} \cdot PrepCount + \Delta t}{PrepCount + 1}$$

Where $PrepCount$ includes the number of times a recipe was already prepared by this user.

In addition to the preparation time of each step, it would also be possible to adapt the heating times of the vessels on the stove. They may vary according to the amount of ingredients, the material of a pot respectively pan, and other kind of influences. At this stage, the user has to monitor the stove if one cooks with a different configuration than specified and eventually turn down the heat or increase the time before the ingredients are done. To predict a precise cooking time on varying configurations, more research is necessary on how the amount of ingredients change the heating time or how different pans or pots affects it.

V. Implementation

The MAMPF prototype consists of the following components: the main system running on an ordinary PC, a display presenting the cooking agent screen, three induction hotplates with 2000W each connected to the PC via USB, a camera which is directed towards the stove, a touchscreen device for remote control, as well as a sound system for speech synthesis and recognition (see Figure 3 and Figure 4). A presentation video of the prototype can be found online[14].

Due to limited resources, it was not possible to develop a zoneless cooking area for our prototype as proposed in Section III-A, so we had to draw on three ordinary hotplates. Anyway we implemented a pot detection and therefore our prototype tracks the movement of vessels from one hotplate to another and automatically regulates the temperature of the hotplates. In our implementation, the amount of hotplates which can be connected to the system is not limited.

For safety reasons, we preferred induction cookers equipped with temperature sensors for our prototype, which were enhanced by micro controllers and USB ports (Fig. 7). The hotplates are able to detect a pot, and turn themselves off when there is no vessel located in the range of the heating area. The vessel is directly heated by an oscillating magnetic field, so the surface of the devices is just heated indirectly by the vessel, and cools down immediately after removing the pot, minimizing the risk of injuries. But also energy efficiency and quick response times to temperature adaptation are appreciable advantages of induction cookers.

One innovation of our system is the substitution of hotplate interaction by directly using pots and pans. Subsequently, jobs have to be assigned to heatable tools which are located on the stove region. Additionally, dynamic information which changes due to cooking processes and static information like hotplate positions are considered during calibration. In the calibration process the user can capture images of kitchen tools as templates for the pot detection via webcam and customize the individual hotplate characteristics like number of, sizes and positions.

The architecture is based on autonomous modules written in C#, using a publish/subscribe-event-driven model for communication. Each event that is published to the event flow is received by all subscribing modules individually. Figure 4 illustrates an overview.

In the following sections, the individual software modules the MAMPF prototype consists of are described in more detail.

A. Intelligent Agent Module

As described in Section IV, our system is able to guide the user through the cooking process. Recipes are saved on a local database and can be exported as XML-Files supporting an easy exchange between different users.

Different tasks need to be completed to enable the cooking support. The first step is to select and load a recipe from the database before starting the cooking process. Then, the recipe is transferred into a dependency tree as well as a step-list representation. This step list is displayed during the cooking process. Furthermore, once a step is accomplished the next step is read aloud to keep the user focused on ongoing tasks. According to the recipe's specification, all steps are executed and temperature instructions are sent to the hardware module.

Time triggers are tasks that need to be executed after a certain period of time and are automatically managed by the system. Each time trigger starts a callback timer, which is initialized with the specified time and executed after its expiration.

As seen in Figure 5, the main components of the intelligent agent's UI are the serial step list and the time triggers. As an experimental case we cooked Kaiserschmarrn, a typical Austrian dish. For the step list we used a scrollable coverflow, which presents a description of the current step and a brief title of the following. In the shown example, the user should put the

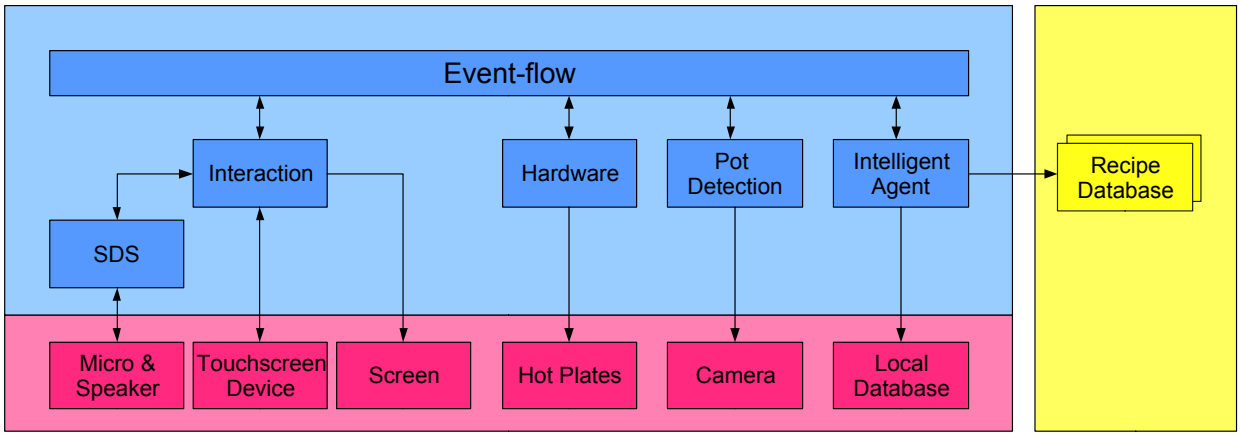

Fig. 4. Implementation overview: **Blue:** MAMPF software modules **Red:** Hardware components **Yellow:** Recipe databases e.g. web servers

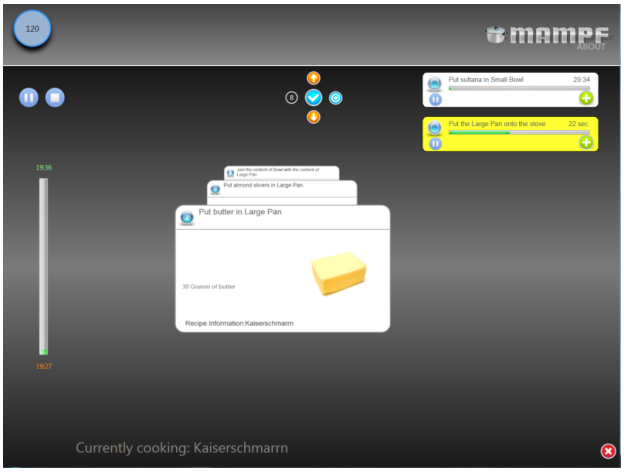

Fig. 5. Screenshot of the Intelligent Agent presenting a recipe

butter into a pan. After confirmation of the step it disappears and "Put almond slivers in Large Pan" moves forward showing a detailed description. On the right hand side we see the time triggers, which means the sultanas need to stay in the small bowl for 29:34 minutes.

B. Pot Detection

Generally, there are many algorithms to find and track objects depending on the input type and affordances like real time capability, maximum error, and object properties. Considering that the implementation goal was to build a system with limited resources, only one camera on top of the stove captures images which are then interpreted by a processing chain as shown in Figure 6. Pots are finally extracted using a SURF-template approach [3]. Hereby, a state machine controls searching areas and frequency for each pot depending on four different pot states (see Figure 8). Pots which are in the ON STOVE state are not searched for, since the position is known and updates are only made if movement around that position is recognized. If a pot is OFF STOVE, the system compares it to whatever object enters the stove. FOUND and LOST are transfer states in which pots reside for a certain amount of time. This assures that pots do not mistakenly change their state due to bad frames containing occlusions or image noise, which influence the feature recognition. A linear least squares approach is used to finally estimate and track the exact pot position from the matching features.

C. Hardware

The main goal of the hardware module is to autonomously execute the tasks which are defined by the cooking agent or the user. Therefore, the module is equipped with a ticket system which stores a maximum of one ticket for each registered tool. Tickets precisely define the jobs, which have to be executed by the hardware system, e.g. temperature and duration settings. Hence, each task received by the module results in creating a ticket or in updating an existing one.

The hardware module implementation also includes an algorithm, which matches a given pot location to the subjacent hotplates. However, the approach of using precise positions even allows the replacement of those hotplates with arbitrarily small induction patches which form a zoneless stove, enabling unrestricted cooking on the entire surface area.

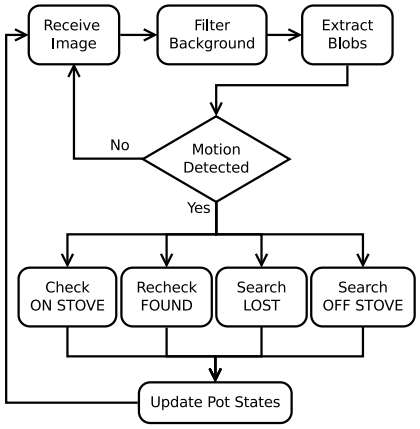

Fig. 6. Processing chain of the Pot Detection Module

D. Interaction

The MAMPF prototype offers two different interaction possibilities, a touch screen device and a speech dialog system. The remote device is connected via WLAN and provides interaction possibilities for the intelligent agent as well as the complete stove and grants the user full control of the system. Nevertheless, the intelligent agent has the possibility to change the instruction set during cooking to adapt to the user interface.

The touch device is only useful up to a certain point as it needs to be operated by hand and is sensitive to heat that might evolve from splashy oil. For this reason we also implemented a speech dialog system using the abilities of Windows 7 with Microsoft's Speech API to recognize speech and give auditory feedback to the user. A predefined grammar according to the system's context is used for recognition. The instructions are produced by a text-to-speech system that allows each module to send arbitrary text. Currently, only the cooking agent uses these features. As a first implementation it lacks the possibilities of natural communication and needs improvements in the future.

Further, the system enables different users to create their own profile to adaptively provide assistance according to the users' personal skills. The history of each user is stored within the local database. Thereby attained information about step durations is then used to achieve better time estimation.

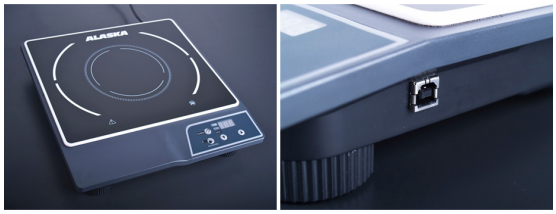

Fig. 7. Standard induction hotplate enhanced with USB port

VI. Future Work

Our first vision combined stove and dining table without the limitation of hotplate regions. It provided the ability to cook food, keep plates warm, support the users by guiding them through recipes and regulating temperatures and timers.

The implemented system described in Section V presents a first step towards this direction. It gives an idea of how our approach may be used to simplify the selection of recipes, the choice of ingredients and cooking utensils, the interaction with the stove and the entire cooking process. Still, there are lots of possibilities to make the system more intuitive and further improve functionality and usability: the different types of context can be utilized to track ingredients, utensils and user interaction as it is mentioned in Section III, while different system components can enhance the application spectrum.

A. Object Recognition

The camera provides information about pots, pans, positions and movements to the system. A more sophisticated sensor network covering most parts of the kitchen and consisting of infrared and CCD cameras as well as microphones and RFID sensors could be used to recognize and track not only cooking vessels but all kinds of objects e.g. by using RFID-tags, particle filters or multi-input hidden Markov Models which are extracted from the different visual input devices [2]. Enhancing this approach with the recognition of cooking actions would result in the possibility to spot and track ingredients throughout the whole preparation. Figure 9 shows two ways to support ingredient detection throughout time.

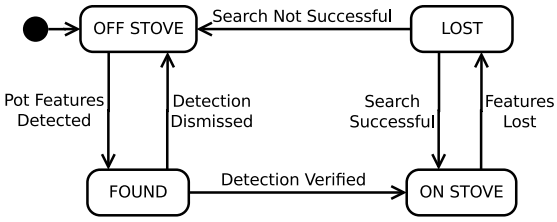

Fig. 8. State machine of each pot

By the help off this information, the intelligent agent is able to interpret the preparation progress and can automatically confirm steps as it holds the current recipe state and therefore knows which step should be performed next. A similar method that uses the idea of recognizing the base of an object and keeping the identity throughout the manufacturing procedures is presented in [20].

B. Interaction Recognition

As recipes represent a major part of the supportive system, they need to be well defined to guarantee a good understanding and easy handling. Machine Learning algorithms could be used to enhance the usability and implicitness of recipe creation and editing. Each interaction of users, utensils and ingredients could be recognized and classified. This way, the system could record and interpret the cooking steps and put them into a semantic and temporal context [13], [8].

C. Failure Avoidance

Cooking leaves a lot of space for failure. Food might be prepared too early or too late, burn or boil over. A combination of heat sensors, cameras and ionization detectors could prevent those types of mistakes and automatically regulate the cooking temperature or warn the users. For example, if the temperature of a pot filled with milk gets dangerously close to the burning temperature or the milk starts to build foam, the system could cool down that pot and save the user from the trouble of cleaning or even a fire breaking out.

D. Flow Graph Optimization

Cooking implies consistently rearranging preparation steps. On the one hand, most dependencies of different tasks are parallel and do not need to be fulfilled at a specific point in time. On the other hand, some steps cannot wait or need to take place as early as possible. Therefore the optimization of the recipe workflow is an important issue to provide sufficient help for the users and needs to be adapted to the number of

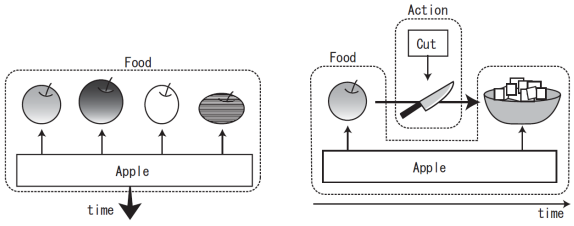

Fig. 9. Object recognition and tracking on an object base [20] **Left:** Recognition of an apple by its homogeneity **Right:** Recognition of an apple by its identity

chefs and their cooking skills. A combination of flow graphs and state-machine-diagrams provide a good base for a user adaptive cooking model as it is proposed in [10]. This type of model would enable the system to calculate the starting time of preparation only by knowing the time the user wants the food to be done.

VII. Conclusion

The Multifunctional and Adaptive Meal Preparation Facility is a feasibility study, examining the amount of facilitation concerning cooking processes that can be achieved by computer support. We concentrated on three core-elements: "zoneless cooking" with focus on the vessel, implicit interaction, and recipe guidance.

"Zoneless cooking" can be achieved with multiple sufficiently sized hotplates linked together and controlled globally. Context awareness allows implicit input, the cook can concentrate on the ingredients and not on controlling the stove. Thus, different types of sensors provide a representation of the kitchen's state. In order to heat a pot, and not a hotplate, a mapping between the pots on the surface and the hotplate-array beneath it was achieved. This concedes the possibility of implicitly interacting with vessels as they are tracked and a limitation of possible intentions can be generated from a formal translation of a recipe. The system contains several state machines and is aware of the cooking utensils, ingredients and actions to be taken. By these means it cannot only literally navigate the cook through the cooking process but also decide whether a vessel has to be heated or not. So, interaction with the stove already starts with putting the pot onto it.

The exemplary implementation illustrates that the initial aims can be achieved with simple methods and limited equipment resources. As a next step other kitchen utensils as ovens, microwaves, and fridges need to be implemented to create a comprehensive kitchen support. The architecture of MAMPF is designed to control a all kinds of devices, but yet the number of suitable kitchen tools that can be connected to a computer system is very limited. Obviously, after installing an intelligent kitchen with connected devices, a user study is inevitable and needs to be considered in future development of MAMPF.

Summing up, lots of improvement is to be done in view of the long-term objective of a modern and networked kitchen equipped with a sophisticated, stable, and implicit Multifunctional and Adaptive Meal Preparation Facility.

References

[1] AEG Electrolux. (2009) Aeg-electrolux maxisight mit multicolor-tft-display. [Online]. Available: http://newsroom.electrolux.com/at/2009/09/17/aeg-electrolux-maxisight-mit-multicolor-tft-display/

[2] T. Aono, H. Kimura, and Y. Yamauchi, "A food recognition algorithm based on dish recognition," in *IECON 02 [Industrial Electronics Society, IEEE 2002 28th Annual Conference of the]*, 2002, pp. 1445 – 1450.

[3] H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool, "Surf: Speeded up robust features," in *Computer Vision and Image Understanding (CVIU), Vol. 110, No. 3*, 2008, pp. 346–359.

[4] Electrolux Design. (2010) Electrolux rendez-vous concept. [Online]. Available: http://www.electroluxdesignlab.com/2010/01/electrolux-rendez-vous/

[5] F. Forest, "Frequency-synchronized resonant converters for the supply of multiwinding coils in induction cooking appliances," in *IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS, VOL. 54, NO. 1*, 2007.

[6] *The Recipe Markup Language (RecipeML)*, FormatData, 4918 Schuyler Drive, Annandale, VA, USA, Apr. 2002. [Online]. Available: http://www.formatdata.com/recipeml/index.html

[7] W. Ju, R. Hurwitz, T. Judd, and B. Lee, "Counteractive: an interactive cookbook for the kitchen counter," in *CHI '01 extended abstracts on Human factors in computing systems*, ser. CHI EA '01. New York, NY, USA: ACM, 2001, pp. 269–270.

[8] P. Lade, N. C. Krishnan, and S. Panchanathan, "Task prediction in cooking activities using hierarchical state space markov chain and object based task grouping," in *2010 IEEE International Symposium on Multimedia*, Taichung, Taiwan, 2010, pp. 284–289.

[9] O. G. Luo Juan, "A comparison of sift, pca-sift and surf," *International Journal of Image Processing (IJIP)*, vol. 3, pp. 143–152, 2009.

[10] K. Miyawaki and M. Sano, "A user adaptive cooking navigation system using ubiquitous sensing environment," in *Applications of Digital Information and Web Technologies, 2008. ICADIWT 2008. First International Conference on the*, Faculty of Information Science and Technology, Osaka Institute of Technology, Osaka, 2008, pp. 378–383.

[11] ——, "A virtual agent for a cooking navigation system using augmented reality," in *Intelligent Virtual Agents*, ser. Lecture Notes in Computer Science, no. 5208. Springer Berlin / Heidelberg, Aug. 2008, pp. 97–103.

[12] D. A. Mundie. (1985) Computerized cooking. Culinary Softwary Systems. [Online]. Available: http://www.anthus.com/Recipes/CompCook.html

[13] Y. Nakauchi, T. Suzuki, A. Tokumasu, and S. Murakami, "Cooking procedure recognition and inference in sensor embedded kitchen," in *Robot and Human Interactive Communication, 2009. RO-MAN 2009. The 18th IEEE International Symposium on*, Dept. of Intell. Interaction Technol., Univ. of Tsukuba, Tsukuba, Japan, 2009, pp. 593–600.

[14] S. Reichel, T. Mueller, O. Stamm, F. Groh, and S. Scherle. (2010) Mampf: Video presentation of the prototype. [Online]. Available: http://vimeo.com/19938951

[15] R. Ribeiro, F. Batista, J. P. Pardal, and N. J. Mamede, "Cooking an ontology," in *Artificial Intelligence: Methodology, Systems, and Applications*, ser. Lecture Notes in Computer Science, no. 4183. Rua Alves Redol, 9: Springer Berlin / Heidelberg, Sep. 2006, pp. 213–221.

[16] W. Schilling, R. Dorwarth, M. Volk, and T. Schoenherr. (2005, Oct.) Verfahren zur Topferkennung. [Online]. Available: http://www.patent-de.com/20070719/DE102005050035A1.html

[17] M. Schneider, "The semantic cookbook: sharing cooking experiences in the smart kitchen," in *Intelligent Environments, 2007. IE 07. 3rd IET International Conference on*, Sept. 2007, pp. 416–423.

[18] K. Tee, K. Moffatt, L. Findlater, E. MacGregor, J. McGrenere, B. Purves, and S. S. Fels, "A visual recipe book for persons with language impairments," in *CHI '05: Proceedings of the SIGCHI conference on Human factors in computing systems*. New York, NY, USA: ACM, 2005, pp. 501–510.

[19] C. Wang and F. Wang, "A knowledge-based strategy for object recognition and reconstruction," *Information Technology and Computer Science, International Conference on*, vol. 1, pp. 387–391, 2009.

[20] Y. Yamakata, K. Kakusho, and M. Minoh, "Object recognition based on object's identity for cooking recognition task," in *2010 IEEE International Symposium on Multimedia*, Taichung, Taiwan, 2010, pp. 278–283.

[21] ZF Electronics GmbH - Cherry, *Pot Detection Sensor*, 2006.