Article

Semantically Annotated Cooking Procedures for an Intelligent Kitchen Environment

George Kondylakis [1,2], George Galanakis [1,2,*], Nikolaos Partarakis [1] and Xenophon Zabulis [1]

1 Foundation for Research and Technology—Hellas, Institute of Computer Science, N. Plastira 100, Vassilika Vouton, GR-700 13 Heraklion, Greece
2 Computer Science Department, University of Crete, Voutes Campus, GR-700 13 Heraklion, Greece
* Correspondence: ggalan@ics.forth.gr

Abstract: Food preparation is one of the essential tasks in daily life and involves a large number of physical interactions between hands, utensils, ingredients, etc. The fundamental unit in the food preparation activity is the concept of a recipe. The recipe describes the cooking process—the way to make a dish in a sequential order of cooking steps. Frequently, following these steps can be an extremely complicated process, which requires coordination, monitoring and execution of multiple tasks simultaneously. This work introduces a cooking assistance system powered by Computer Vision techniques that provide the user with guidance in the accomplishment of a cooking activity in terms of a recipe and its correct execution. The system can provide the user with guidance for carrying out a recipe through the appropriate messages, which appear in a panel specifically designed for the user. Throughout the process, the system can validate the correctness of each step by (a) detection and motion estimation of the ingredients and utensils in the scene and (b) spatial arrangement of them in terms of where each one is located to another. The system was first evaluated on individual algorithmic steps and on the end-to-end execution of two recipes with promising results.

Keywords: cooking assistance; recipe preparation; intelligent kitchen; object detection; object tracking; computer vision; machine learning

check for updates

Citation: Kondylakis, G.; Galanakis, G.; Partarakis, N.; Zabulis, X. Semantically Annotated Cooking Procedures for an Intelligent Kitchen Environment. *Electronics* **2022**, *11*, 3148. https://doi.org/10.3390/electronics11193148

Academic Editor: Zhenhua Guo

Received: 29 August 2022
Accepted: 25 September 2022
Published: 30 September 2022

1. Introduction

Cooking is a very important aspect of everyday life—a task that usually requires a lot of practice and experience while at the same time being very time-consuming. Cooking takes place inside the kitchen, which, according to L. C. Johnson [1], "is the environment where work blends with desire, enjoyment, imagination, safety and other people, and where domestic technology, architects and designers create devices and space".

Activities in the kitchen environment are particularly complex, usually involving a large number of objects, gestures as well as cognitive relations between them. The complexity of a task such as cooking can be magnified if one thinks that an individual can slightly modify a common recipe and thus contribute to the general multiplicity of the enterprise.

A cooking recipe is the fundamental unit of a cooking activity and is the core of culinary art. It consists of a set of instructions that describes how to prepare or make something, especially a dish of prepared food. The earliest known written recipes date to 1730 BCE and were found in three cuneiform tablets in Mesopotamia [2]. Since then, the food preparation process has been commercialized and industrialized as a valuable part of food process engineering, a concept that involves the entire chain from agricultural production up to the packaging, distribution and eventually consumption of food [3].

At the industrial scale, computer-aided solutions have been present for many years and for a variety of aspects in the food industry, such as quality assurance, safety and hygiene [3]. On the other hand, with the huge advances in technology and the increasing interest in enhancing humans in their regular lives, cooking recipes started to be digitized in order to be accessible from every digital device. Researchers study the process of cooking activities in order to enhance humans in their regular lives and endeavor to enrich this process by integrating multimedia and embedding technologies in various stages. This is achieved by making these procedures more convenient, less time consuming or by enabling people through Ambient Assisted Living (AAL) to conduct otherwise unsolvable tasks. However, preparation activities for cooking in the kitchen involve physical interactions between multiple objects such as hands, utensils and ingredients, rendering it a very challenging research problem.

In the recent past, Computer Vision (CV), Robotics and Machine Learning (ML) started drastically venturing into cooking assistance. These fields provide insight into some of the most unexcogitable problems while at the same time facilitating the development of more realistic systems which could nonetheless address more comprehensively the cooking activity process. CV for human sensing has traditionally focused on using RGB images. Theoretically, fully automatic tools that integrate CV algorithms to extract and identify the elements of interest across frames could be developed. Unfortunately, state-of-the-art algorithms are not error free, and false positive and false negative detections would require a human effort to correct them in a post-processing stage.

The present study introduces an assistive cooking system that provides the user guidance in the accomplishment of a cooking activity via CV and ML. The potential of this work lies in the fact that cooking activities can be monitored and thus modeled in a dynamic infrastructure that does not rest on the use of cumbersome sensors or hardware, thus rendering it employable to the average user who cannot have access to expensive hardware, which may also require the installation of supplementary material. This does not by any means suggest that this is a standalone system that aims to replace elaborate structures deployed in smart homes or smart kitchens. On the contrary, it is a system designed specifically for the needs of ICS-FORTH's Intelligent Home and aspires to be used in the ambient facilities of the Intelligent Kitchen. Alongside other pre-existing systems, the main goal of this work is to support and guide the cooking process, decrease the perceived complexity of unfamiliar recipes, increase user confidence about the success of intermediate steps and finally reduce the user's inhibitions about using technology in a kitchen for fear of damaging it.

In summary, some of the modalities and aspects of this research work are the following:

- The system supports the spatial arrangement of objects such as utensils and food ingredients in terms of where each one is located relative to another. This can be interpreted in various ways, bearing in mind the needs of each user.
- The system supports assistance of the cooking activity in terms of recipes and their correct execution. Each recipe is imported into the system in the form of a configuration file. The system is able to construe this configuration file and translate it into human perceived knowledge. It can facilitate the cooking procedure while at the same time providing the user with advice on how to carry out the specific task (i.e., carrying out the recipe).
- The system primarily supports all the necessary transformations of a raw image into real-world case scenarios via algorithmic approaches.

2. Related Work

Many attempts have been made to provide systems that assist the user in the culinary process, both at the research and industrial level. However, few of them have tried to offer a cooking experience similar to the actual cooking assistance during the cooking process, and even less take advantage of what CV and ML have to offer.

2.1. Literature Review

In this section, we present the state-of-the-art perspective of each work. Discussion over the presented approaches is included in Section 2.2.

2.1.1. Systems

Initially, we present some more complete systems, meaning that they capture the user and/or utensils and ingredients for predicting or assisting the meal preparation. Ref. [4] approaches food preparation activities with the use of ML algorithms combined with visual and accelerometer data. The goal of multi-modal activity recognition is achieved through an accelerometer localization algorithm that facilitates the merge of accelerometer and visual data. A statistical activity model is also introduced in order to guide the construction of datasets for complex activities and give an outlook on the next steps.

The contribution of [5] is a novel method for merging accelerometers and CV for activity recognition. The multi-modal activity recognition dataset used in this work includes more than 4.5 h of annotated accelerometer and RGB-D video data. The evaluation of the experiments of the method and the comparison of various fusion methods also include a protocol for benchmarking.

Ref. [6] provides accurate cooking activities recognition as the first step to realizing cooking assistance service. The main focus is on the problem of cooking activity recognition in egocentric videos. The proposed method is based on hand detection, where the region around the hand is extracted based on the detected hand in order to facilitate detection in cluttered backgrounds. The approach of this work is to combine 2D convolutional neural networks (2DCNN) for the motion and 3D convolutional neural networks (3DCNN) for the appearance. The analysis of the first-person video is deemed very useful, especially for videos containing different hand motions and utensils. Therefore, a new dataset that consists of eight cooking-specific activities, such as "Peeling", is proposed. In this activity, the hand motion could vary on whether a knife or a peeler is used.

Dipak Surie, Saeed Partonia and Helena Lindgren proposed Kitchen As-A-Pal [7], an interactive smart kitchen with real-time human sensing capabilities. In particular, Kitchen As-A-Pal is equipped with CV systems, which are based on the fusion of fisher face recognition and skeletal tracking approaches. A Microsoft Kinect [8] is used to localize and track one or more humans, whereas the fused approach gives human identity recognition accuracy of 91.75 precision and 66 recall values for single occupant setting with good smart space coverage.

The authors in [9] proposed a use-centric cooking support system named Smart kitchen. The role of the Smart Kitchen is to know beforehand the recipe that the user wants to execute and give instructions when the cooking steps are finished or when the user asks. An important part in the aforementioned is that the user can change the cooking steps freely whenever they want. Smart Kitchen supports three main protocol functionalities: tracking food, recognizing food material and recognizing cooking actions.

Cook's Collage [10] realizes a recipe tracking service through a video-based approach. If a cook interrupts their cooking, they are provided with a memory aid in the form of a video summary of what they have already done. More precisely, the user has access to images that show the last six actions and receives information about which ingredients are used and in what quantity with respect to the order of the performed actions. An embedded camera above the preparation area records the user's actions. In stress situations, memory aids such as video and images have proven helpful, according to this work.

KogniChef [11] is a smart kitchen environment and software framework that implements a cooking recipe assistant. The prototype features a framework for the integration of multi-modal displays, a sensor layer, interfaces for control, as well as a controller that connects all these components together. The system deploys object detection and tracking as one of the most central components because it defines a fundamental basis for the tracking of cooking actions carried out by the user. These modules provide a list of Object-Beliefs, each referring to a detected and classified object in the scene. Based on an XYZ/RGB/temperature point-cloud, identical objects can be distinguished.

The work of Nguyen Thi Thanh Thuy and Nguyen Ngoc Diep [12] proposes a learning method for food preparation activity that is based on feature learning from histograms of motion primitives. Its effectiveness is validated by recognizing ten activity classes. The segmentation of sensor data streams into frames was accomplished through a sliding windows approach, following the transformation of the sensor signals into a feature vector. Afterward, the feature vector is classified using a Decision Tree classifier.

The authors in [13] proposed PIC2DISH, a cooking assistant that, given an image of food and a list of ingredients, provides the user with instructive videos demonstrating the utilization of the ingredients (e.g., how to cut a cucumber into slices) in order to complete the recipe. Initially, the system recognizes the recipe and the corresponding ingredients using a multi-task Convolutional Neural Network (CNN) and then decides if some ingredients need to be substituted because they do not exist in the user's list. This is achieved using a knowledge graph, which encodes the relationship and co-occurrence of ingredients and is learned using millions of recipes and their variants from an online source. Finally, the user is provided with the steps of the recipe along with video clips related to these ingredients.

A more recent work is HanKA [14], which is a modular cooking assistant. HanKA supports the detection and control of kitchen devices and additional user interfaces for the purpose of task planning throughout the execution of a recipe. A recipe in HanKA is represented by a knowledge graph, which includes the temporal aspect; for example, "a potato has to be cooked in the steamer for 17 min". The system expects the user to give feedback for some manual tasks (e.g., slicing) and expects that each device has a control interface such that it can be scheduled for the corresponding task (e.g., *cook for X minutes*). This way, HanKA can schedule the tasks in a smart way such that a user can do manual tasks while waiting for devices to finish theirs.

Apart from research prototypes, commercial cooking assistant products exist as well. Bosch Cookit [15] and Thermomix TM6 [16] share the same spirit, that being an all-in-one appliance for cooking, cutting and slicing and also integrates a screen for recipe guidance and programs selection. Both come with many recipes pre-installed on the device.

2.1.2. Datasets and Tools

Besides meal preparation assisting systems, some tools and datasets have been proposed in order to complement them. iVAT [17] is an interactive video annotation tool that supports manual, semi-automatic and automatic annotations with various detection algorithms. The input could be a video and a related list of items, while the output is an item annotation or even a template for that annotation. There is an option of categorizing the items to be annotated into categories such as food, kitchenware or action.

On the main screen, there is video-related information, such as a list of various shots and items, while the most important feature is a video browser that allows the user to seek through frames and sequentially browse shots. The most recently annotated frames are shown in the video frame, while at the same time, the user can have access to interaction modalities, e.g., clickable buttons, drag and drop operations, context menus and shortcuts. The lower part of the window shows an item's timeline every time the item is annotated manually or automatically. The cell in the timeline corresponds to a frame position and shows if the item is present in that frame.

The work of Marcus Rohrbach, Sikandar Amin et al. [18] proposes a new activity dataset that contains 65 activities recorded in a realistic setting. The activities are, for the most part, fine-grained and display a low inter-class variability to be able to evaluate classification and detection performance. Cooking activities are handled with extreme caution and are dissected into small differences in activities.

The contribution of this work relies on the novelty of the dataset, which introduces a classification and detection challenge together with appropriate evaluation criteria. The input data are high-resolution image and video sequences (jpg/avi), with the corresponding activity class and time interval annotations. The pre-computed mid-level representations are in the form of a pre-computed pose.

2.2. Discussion

The presented review of the literature has revealed several features that are supported by frameworks targeting the Intelligent Kitchen as well as cooking activity recognition. The majority of the mentioned research systems focus on the process of developing software that can understand or classify cooking activities via the use of various hardware, e.g., sensors, cameras, etc. [5,6,18]. At the same time, there is an attempt to present more complete systems in a way that becomes apparent, i.e., the assistance of the cooking process can be applied in the existing hardware [10]. Of course, by using the term "complete", there lies the implication that an idea or a system is better than any other that has not been applied to real-case scenarios such as a fully employable Smart Kitchen [11]. Next, there is going to be a typology of the discussed literature as it is straightforward even for the most benighted reader to follow the aforementioned literature review.

Most of the systems that utilize CV and ML aim at food preparation activity recognition or activity recognition in general. Systems like that [9] focus on solving intractable problems in CV, such as occlusion or food change of form during cooking. In the same context, Ref. [7] move toward human sensing in terms of ubiquitous computing of human activity inside the smart house, ergo focusing on face recognition and skeletal tracking. Much research is also dedicated to the fusion of CV with data provided by sensors, a step toward multi-modal activity recognition [4,5], even though it is not successful in evaluating different fusion techniques as well as leaving a lot unanswered regarding the credibility of such experimentations. Subsequently, the following research work, also published by Sebastian Stein and Stephen J. McKenna [4], introduces datasets of people performing food preparation activities as a means of evaluating different fusion techniques, additionally in the context of the recipe and ingredients acted upon. On the same note, Ref. [17] focuses on annotating cooking videos and supports manual, semi-automatic and automatic annotations obtained on the basis of the interaction of the user with various detection algorithms but does not establish a protocol for evaluating different techniques [5,7,9]. Nevertheless, it establishes an incremental learning framework that allows increasing the accuracy of the underlying algorithms over time, which is compared to the works of Thuy, Nguyen Thi Thanh et al. [12] and Urabe, Shuichi and Inoue et al. [6]. The more "machine-learning based" approaches depend on neural networks to comprehensively respond to the challenge of cooking activities recognition. While both [6,12] propose custom datasets with food preparation activities, Ref. [12] only base their results on motion primitives and do not combine them with activity streams. On the other hand, Ref. [6] combines appearance-based approaches using 2DCNN and motion-based methods using 3DCNN, which enables the improvement of the recognition performance of cooking activities on their dataset. Last but not least, KogniChef [11] is a commercial product that combines annotation modality, such as in [4,12,17], reinforced with an XYZ/RGB/temperature point-cloud, which serves as a sensor system for monitoring the food preparation activities, much like [4–6,12,17]. The KogniChef platform also enlists assistance in the cooking process using a spatio-temporal tracking layer and supports a user interface that is specialized for the kitchen configuration and guides the user through the recipe. Different to other systems, PIC2DISH [13] does not capture the cooking environment but only the photo of an existing cooked recipe. Although it is able to recognize the recipe and corresponding ingredients from the photo, its applicability is limited because, in a real scenario, photos with cooked meals that appear in online sources are usually accompanied by recipes. Therefore, the generation of the recipe can depend solely on textual information. Furthermore, it does not include real-time feedback on the existing state of the recipe. Finally, the learning part of HanKA [14] is the planning of individual tasks of the recipe. Planning is very useful, especially for an inexperienced user; however, the entire system depends on existing devices, which can be discovered and controlled externally.

There are many features presented in the above section, some of which display great diversity from one another. Each system has its own weaknesses and strengths and deserves equal and impartial study from the research world in order to gain and benefit both from the good points but also from the inadequacies. For example, HanKA can avail from a system that utilizes CV for real-time feedback, as proposed, to mitigate the need for user confirmation after completing manual tasks. It can automatically recognize that the user has finished slicing onions, rather than expecting the user to manually confirm it.

In summary, the examined systems support the following features:

1. Computer vision techniques. Includes the research based on the field of computer vision, varying from computer vision-based tracking to object detection and providing an answer to the fundamental question of what is on the scene in a food preparation activity. High intra-class variability often foreshadows these techniques' credibility. Features such as pose representation and face recognition are considered reliable for this kind of research and are often deployed in order to interpret the volitions of a user.
2. Creation and annotation of datasets. The annotation of cooking videos requires special care due to the difficulty of gathering experimental data. It is an extremely laborious process that nonetheless provides significant records on everything regarding a food preparation activity. It is to be noted that the process of annotating and documenting the results in a dataset is the utmost and most impartial way of contributing to the progress of the field.
3. Activity Recognition (AR). Activity recognition aims to recognize the actions and goals of the user in a food preparation activity such as a cooking recipe.
4. Sensor fusion. There is a lot of research in the direction of using more than one sensor and combining the obtained data to advance food preparation activities. The sensors can be cameras or accelerometers attached to the ingredients themselves.
5. Machine learning. Machine learning methods are often deployed either for enhancing computer-vision-based techniques or as standalone computational methods. Artificial Intelligence and Machine Learning have "infiltrated" horizontally various disciplines, and Computer Vision is no exception. Models such as YOLO [19] or Detectron [20], developed by the Facebook AI Research team, have improved tasks such as object detection.

The association of previously cited literature works to the above categories is presented in Table 1.

Table 1. Comparative Study of Literature work.

Related Work	ML	AR	Spatial Config.	CV	Dataset Creation	Sensor Fusion
[5]	–	✓	–	–	–	✓
[7]	–	–	–	✓	–	–
[9]	–	✓	–	✓	–	–
[10]	–	–	–	–	–	–
[12]	✓	✓	–	–	–	✓
[4]	–	✓	–	✓	–	✓
[18]	–	✓	–	✓	✓	–
[11]	–	–	–	✓	–	–
[6]	✓	✓	–	✓	✓	–
[13]	✓	–	–	✓	–	–
[14]	✓	–	–	–	–	–
Present work	✓	–	✓	✓	✓	–

3. Method

The system that we propose comprises two high-level components, as shown in Figure 1. The first component (Figure 1-left) corresponds to the cooking procedure application, which orchestrates our pipeline. The second component (Figure 1-right) is related to the object detection task. For image sensing, we utilized a single Kinect for Windows v2 RGB-D that was interfaced through the libfreenect2 driver and corresponding library [21]. We chose such a type of sensor for the reason that it offers a wide-angle lens and, therefore, is suitable for capturing larger areas even if it is installed at lower heights. Although we do not currently utilize the depth stream of the sensor, we regard it as an important cue for the setup, and it will be exploited in future work.

The cooking procedures application iteratively runs the subsequent steps. At first, an RGB image is fetched from the camera and then lens distortion correction is applied. Afterward, the image is given as input to the object detector component, and the result contains the bounding boxes and labels of the detected objects, if any. The detections are passed through object tracking, which compensates for possible transient misses. The adjusted bounding boxes are considered as moving or not by applying an optical flow algorithm. Finally, boxes, labels and flow information of them are given as input to the recipe decision logic component for the evaluation of the current state of the recipe.

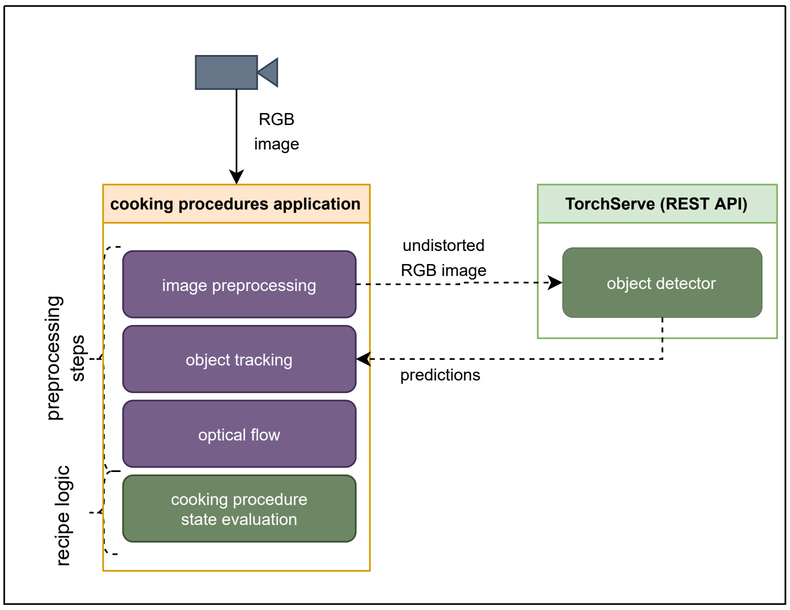


Figure 1. Application architecture of our system.

3.1. Detection of the Objects

In CV and ML, object detection refers to the two-step process of finding bounding boxes that contain exactly one type of object inside, as well as classifying this exact object and assigning it to a label. This label corresponds to the type of object that is included in the bounding box. Detecting and identifying the existing objects correctly is crucial and should be verified in every step of the procedure. The object's detection accuracy is scored.

3.1.1. Object Detection Framework

In recent years, in parallel with the emergence of ML frameworks (i.e., PyTorch [22], Tensorflow [23]), corresponding task-specific frameworks have also evolved. Their purpose is to provide a set of task-related workflows, including common configurations, pre-trained models, and training and evaluation scripts, such that the setup of a model for a new dataset is achieved with less effort. In this work, we utilized Detectron2 [20], an object detection framework implemented by Facebook AI Research (FAIR) on top of PyTorch.

Detectron2 provides state-of-the-art detection and segmentation algorithms. It supports a number of CV research projects and production applications in Facebook.

Each detection algorithm can essentially be thought of as comprising two parts. The first part is related to the extraction of feature maps from the image, and the second part is related to detection, which consists of proposing object regions (boxes) and classification of them according to the predefined set of classes.

In Detectron2, a variety of models for features map extraction are implemented, including models based on CNN [24] and Visual Transformer (ViT) [25]. In regards to the detection part, Detectron2 provides implementations for the R-CNN family of detectors. A popular one is the Faster R-CNN detector [26], which uses a novel region proposal network (RPN) for generating region proposals. Faster R-CNN was extended by Mask R-CNN [27], also included in Detectron2, which provides per-pixel predictions of the objects labels, in addition to bounding boxes. In our case, object masks are not mandatory to compute; therefore, we chose the plain Faster R-CNN detector with ResNeXt-101 CNN model [28] for feature map extraction. Although a ViT backbone would achieve better accuracy, we use the aforementioned CNN model as we found that it provides a good trade-off for speed, accuracy and GPU memory footprint.

3.1.2. The Need for a Custom Dataset

Apart from the models themselves, Detectron2 provides corresponding parameters, the so called model weights. However, in most cases, these pre-trained models are solely for demonstration purposes and have limited applicability for custom object classes, as required by a real-world scenario. The reason is that these models have been trained on publicly available datasets, specially designed for general-purpose object detection. Two of them, COCO [29] and LVIS [30], include everyday objects that correspond to some general categories such as food, furniture and animals and hundreds or thousands of images of them along with corresponding annotations. However, the obvious drawback is that they do not focus on a single category. For example, COCO includes a "bowl" class but not a "plate" one. LVIS, on the other hand, includes most cutlery and tableware objects, as required by our task, but has a limited variety of foods. Another dataset that was proposed recently, named EPIC-KITCHENS [31], focuses exactly on kitchen-related tasks and includes a large variety of kitchen utensils and food. Even such a thorough dataset has its own limitations though. We summarize them below.

- A large dataset does not and cannot include all possible ingredients required for preparing special recipes such as traditional recipes.
- Annotations about utensils are stateless. For example, there is no distinction between an empty bowl and a bowl with some ingredients in it
- For some classes we may need finer detection about specific object classes rather than relying on a single class. For example, we may need to specify which bowl is the one we are interested in, small-vs.-large or plastic-vs.-glass.

From the above remarks, we conclude that object detection for objects involved in cooking procedures can be performed as follows.

1. Utilize an existing detector as is, predict every class that it supports and discard objects not interested in.
2. Train a detector with a subset of classes of an existing dataset;
   (a) Utilize solely the existing dataset;
   (b) Augment dataset with images obtained from custom objects instances.
3. Train a detector solely with a new dataset.

In our work, we considered cases 2 and 3. In the following sections, the entire process of detecting objects used in a cooking procedure will be explained in detail. In either case, we consider that starting from a pre-trained detection model is of great benefit. Such an approach is usually adopted because it accelerates training time, provides a good starting point for the weights of the neural network and helps to avoid over-fitting, especially if training involves few images and classes, as in our case. For case 3, we also specify the procedure and tools related to the creation of a new dataset. An example output of the detector trained on our custom dataset is shown in Figure 2.

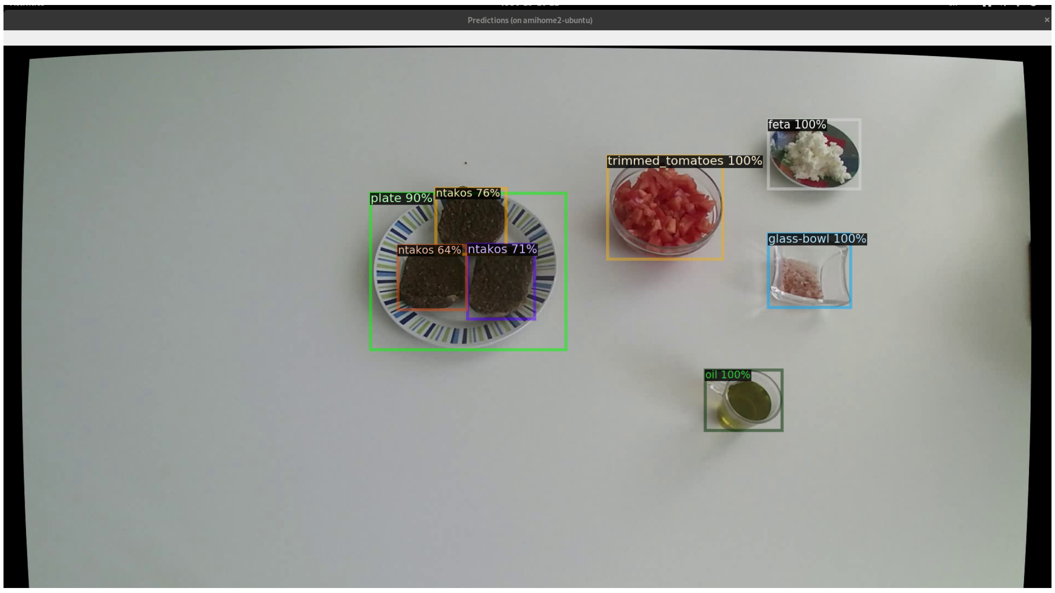


Figure 2. The output of the trained model: bounding boxes, scores and labels.

3.1.3. Utilization of Public Datasets

In this section, we investigate the utilization of existing public datasets and pre-trained models, a case which is adequate only if all recipe objects correspond to a class in the dataset. For this case, we considered the COCO dataset. From the 80 object categories, we considered seven of interest, and more specifically, “bottle”, “wine glass”, “cup”, “fork”, “knife”, “spoon” and “bowl”.

Let us say that D_0 is a detection model, pre-trained on all COCO categories. The possible options for refining D_0 for the custom classes and/or custom objects instances are the following:

1. Using a subset of COCO, keeping only classes of interest;
2. Using images of the specific objects instances that we want to further classify;
3. Combination of 1, 2.

Implementation of option 1 is straightforward. This is by creating a subset of the dataset, retaining only the relevant annotations, and training using this subset. For option 2, we considered an approach which required minimal effort, by automatically annotating videos of the custom object instances. More specifically, we recorded a video per object instance and ran a detector trained as specified in option 1 in order to obtain the pseudo ground-truth annotations. We also kept only detections with large confidence score (0.7). Afterwards, we refined D_0 using only these automatically generated annotations. For the first option, we realized that even for these few classes, the detector performed poorly in some cases. For the second case, the detector performed well when deployed under equivalent lighting conditions but very poorly if the lighting conditions were dramatically changed. An explanation for the latter is that the model overfitted not only to the custom objects but on the specific views of them. This was despite some standard data augmentation techniques that were adopted during training.

For this reason, we evaluated option 3. In this case, the dataset was created using a concatenation of the COCO subset and at most 150 images per object from the dataset that was created for option 2. The motivation about this was to instruct the model to retain some general class information, e.g., about the structure and shape of the object and simultaneously learn specific object instance information, e.g., shape and appearance. The combined model achieved a balance between generalization and specialization. However, the drawback, as mentioned before, is that the detector is limited to only the classes that are contained in the public dataset. Although we found such combined option useful, we did not further consider it through formal evaluation and we opted for the creation of a new custom dataset.

3.1.4. Training with Custom Datasets

A custom dataset is a dataset that is created by the user, meaning that the user annotates the objects that the model will be trained to. As also explained in Secion 3.1, the annotations consist of the bounding boxes that include the concerning object and the label of the object. The tool used for labeling out images to train the model is CVAT [32]. CVAT is a free, online, interactive video and image annotation tool for CV. Figure 3 depicts an example of the annotation workspace screen of CVAT.

In this case, each object and each state of each (e.g., empty bowl vs. not) can be annotated with a different label, such that they correspond to separate classes. We consider that a detector trained on such annotations is a baseline object re-identification model with object state knowledge. It is considered as a baseline, because no specific re-identification or state methodology has been applied; however, it serves such prediction information.

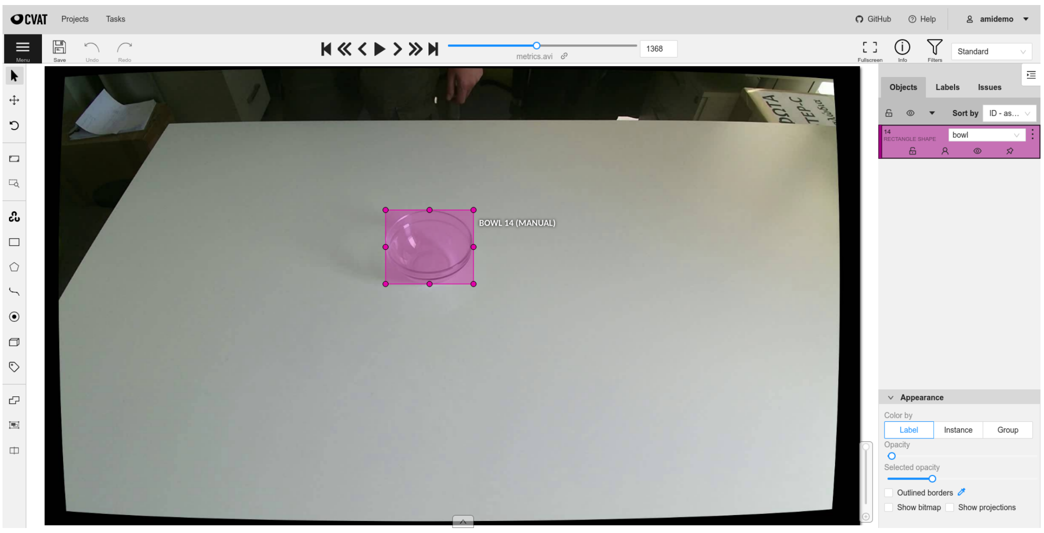


Figure 3. The annotation workspace screen of CVAT. In the particular case, the video frame contains an object corresponding to "bowl" class and a bounding box containing exactly the object that has already been created.

We considered the execution of two recipes, namely "Greek Ntakos" and "Fava". For "Greek Ntakos", the required ingredients are ntakos, feta, tomato and oil. For "Fava", the required ingredients are fava, lemon, oil and onion. Corresponding utensils such as plates, bowls and cups are also required for both recipes. The complete list of utensils and ingredients for them is presented in Tables 2 and 3 along with indicative images.

Table 2. Indicative images from the dataset for "Greek Ntakos" recipe.

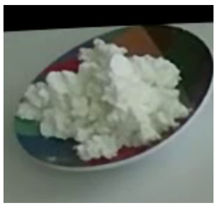
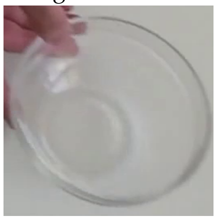
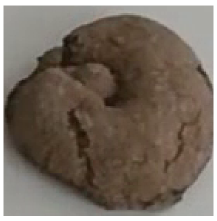
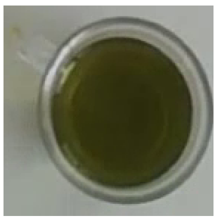
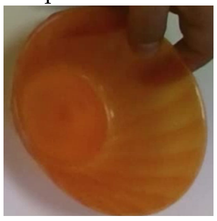
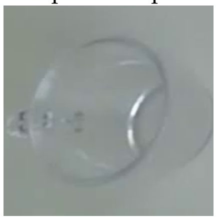
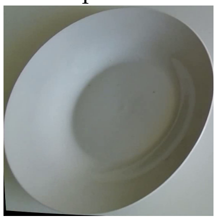
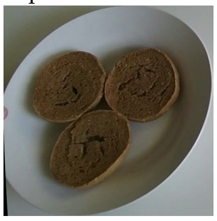
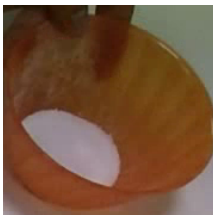
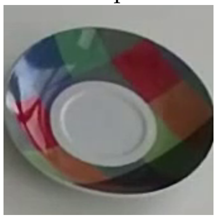
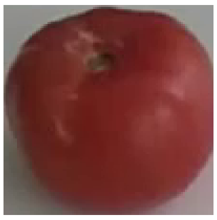
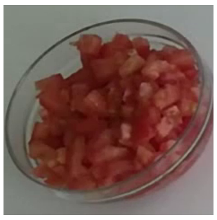
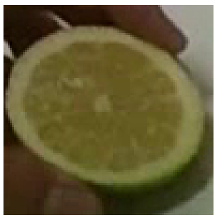
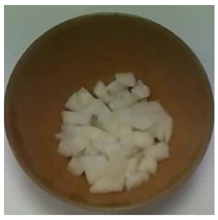
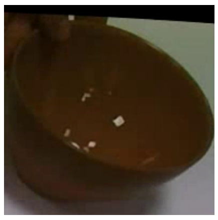
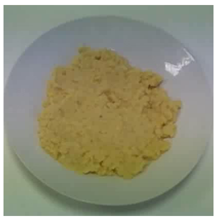
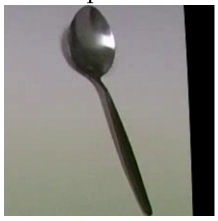
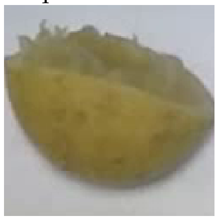
feta	glass bowl	ntakos	oil
			
plastic bowl	plastic cup	plate	plate w. ntakos
			
salt	small plate	tomato	trimmed tomatoes
			

Table 3. Indicative images from the dataset for "Fava" recipe. For the ingredients and utensils that are not shown (oil, salt, plastic bowl, plastic cup) please refer to Table 2.

lemon	onion	onion bowl
		
fava	spoon	squished lemon
		

The dataset was created as follows. For each object, a separate video was recorded. The resolution of the video was 1920 × 1080, the highest of the Kinect v2 sensor. During recording, a person was instructed to place the object at various positions on the table, such that a range of positions, poses and lighting variations was captured. This is crucial for the object detector to be able to learn object characteristics that are invariant to these changes in appearance. Each video was imported into CVAT and annotations were created from sparse frames of the video, which corresponded to the different placements of the object.

An example of the annotation process for some of the objects of “Greek Ntakos” recipe is presented in Figure 4. The number of annotations per object are shown in Table 4.

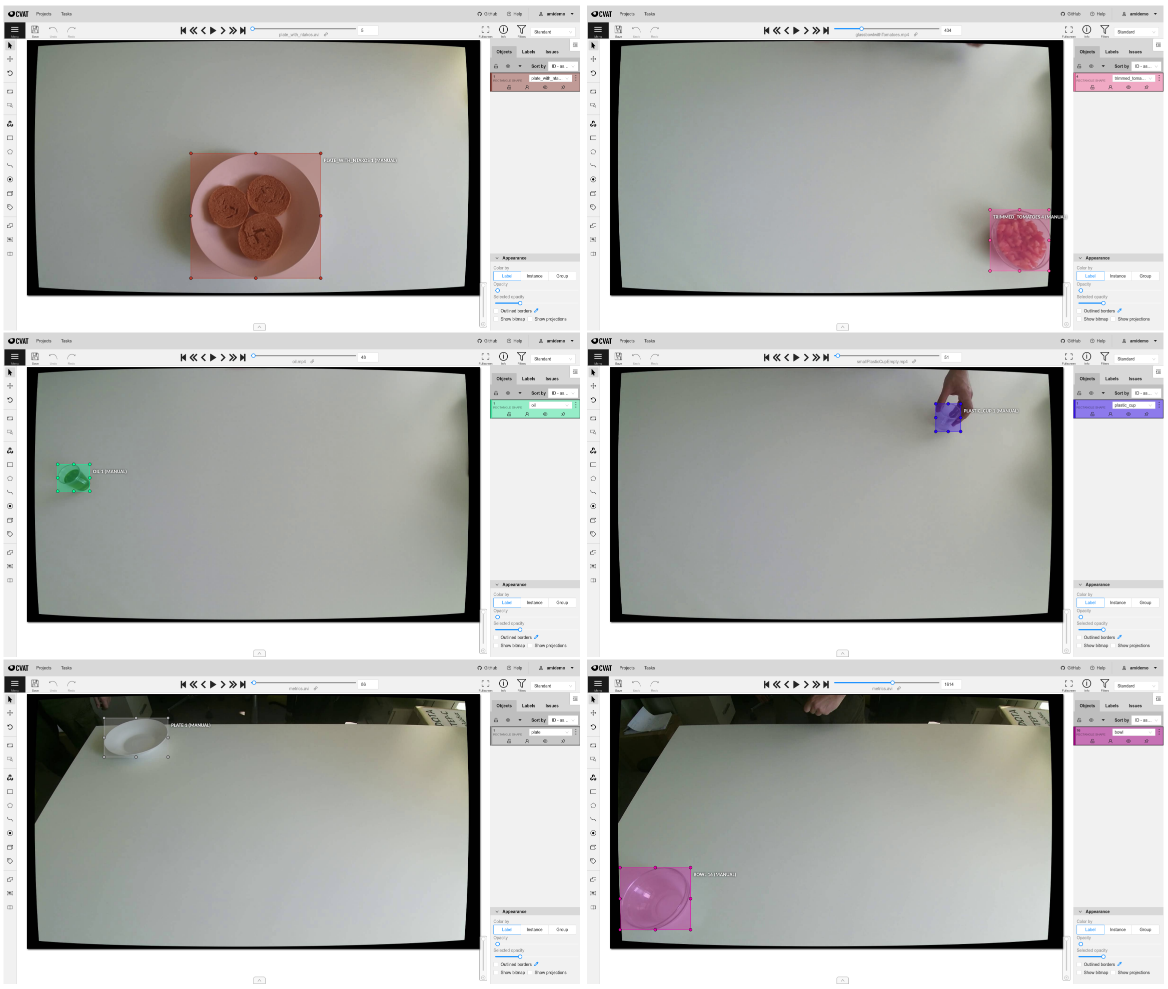


Figure 4. CVAT screens at various annotation stages. Each object was moved at various positions such that a variety of poses and locations was captured.

Table 4. The number of bounding boxes that were annotated for each of the objects. Objects from both recipes are shown.

Class Name	#bboxes	Class Name	#bboxes
fava	30	plastic cup	42
feta	46	plate with ntakos	18
glass bowl	30	plate	9
lemon	31	salt	30
ntakos	49	small plate	32
oil	52	spoon	30
onion bowl	30	squished lemon	30
onion	30	tomato	45
plastic bowl	30	trimmed tomatoes	31

3.1.5. Model Deployment

A final step beyond model training is the so-called model deployment—the integration of the model into a production environment such that it can be utilized from separate applications. For this purpose, we utilized TorchServe, an open-source model serving framework for PyTorch. TorchServe makes it easy to deploy trained PyTorch models without having to write custom code. TorchServe delivers lightweight serving with low latency, so models can be deployed for high performance inference. It provides default handlers for the most common applications such as object detection and text classification, so it is not necessary to write custom code to deploy them. It supports serving multiple models simultaneously using a RESTful API.

3.2. Object Tracking

Object tracking was implemented using the MultiTracker class and, more specifically, OpenCV's DCF-CSR tracker [33], which is available through a common multi-object tracking API. This tracker is not the fastest but it produces the best results in many cases examined.

Object tracking is used as a preventive measure in the case of failure of object detection and is deployed particularly for this scenario. In the context of this research work, object tracking is not used for tracking an object but to localize the bounding boxes of the objects in the uncommonly rare event that object detection fails. The method is deployed periodically, unlike object detection and optical flow (see Secion 3.3). Figure 5 shows simultaneous detection and tracking.

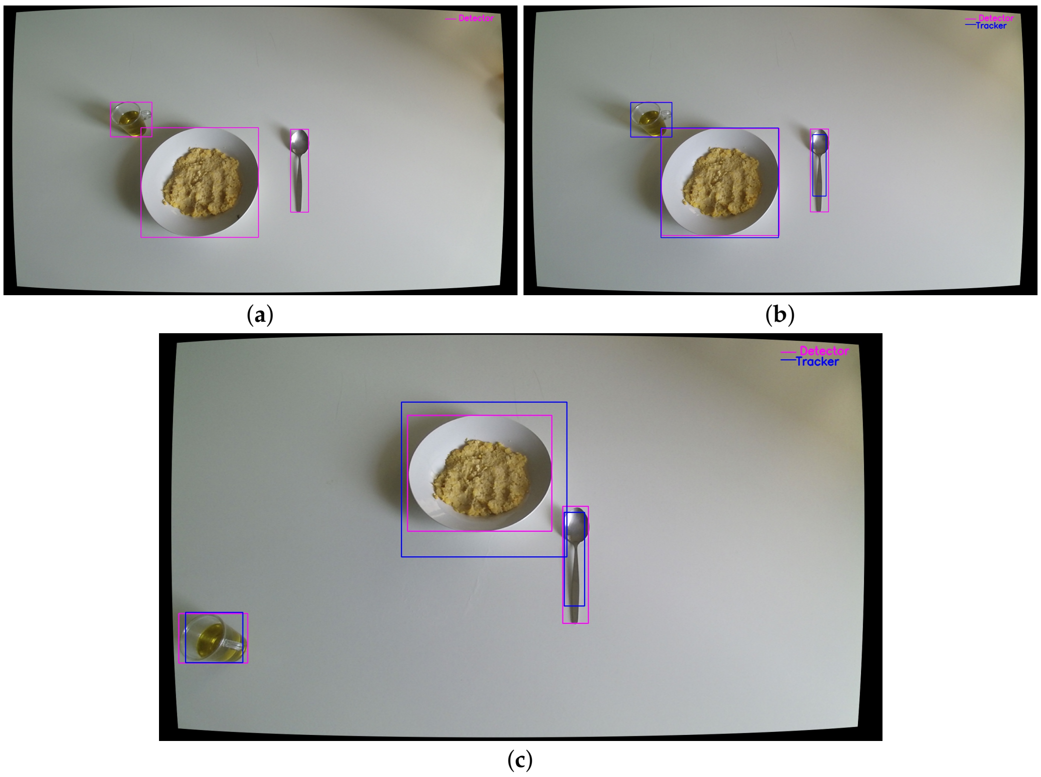


Figure 5. Showcase of simultaneous object detection and tracking: (**a**) solely object detection, (**b**) combination of object detection and tracking, (**c**) updated detection and tracking.

3.3. Optical Flow

Optical Flow provides information to the system about the motion of the objects, which can be used in conjunction with data we extract from Object Detection in order to identify which is the object that performs the movement. The relationship of Object Detection and Optical Flow plays a crucial role in this research work, as explained in Secion 3.4.3.

Three methods of Optical Flow were implemented. An experiment was designed, where each method was tested with the same data set (see Figure 6), in pursuance of the best of these three methods in terms of performance and accuracy (see Secion 4.4.1).

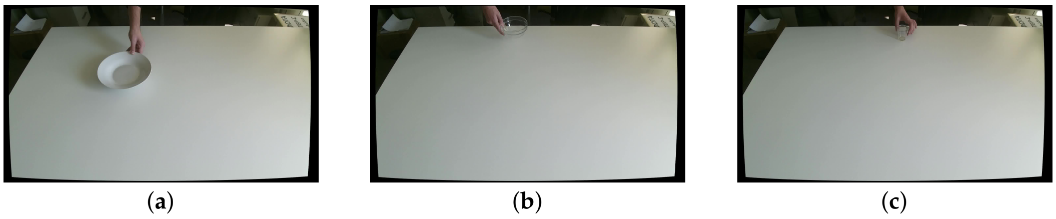


Figure 6. The objects that were utilized for the optical flow experiment: (**a**) plate, (**b**) glass bowl, (**c**) plastic cup.

The following methods were implemented in this research work:

1. Sparse Optical Flow with Lucas–Kanade algorithm [34–36]. Figure 7 shows the visualization of the movement of the objects.
2. Dense Optical Flow with interpolation of Sparse Optical Flow with Lucas–Kanade algorithm [34,35]. Figure 8 shows the visualization of the angle (direction) of flow by hue and the distance (magnitude) of flow by value of HSV color representation.
3. Dense Optical Flow with Farneback algorithm [34,35,37]. The visualization of the optical flow is shown in Figure 9.

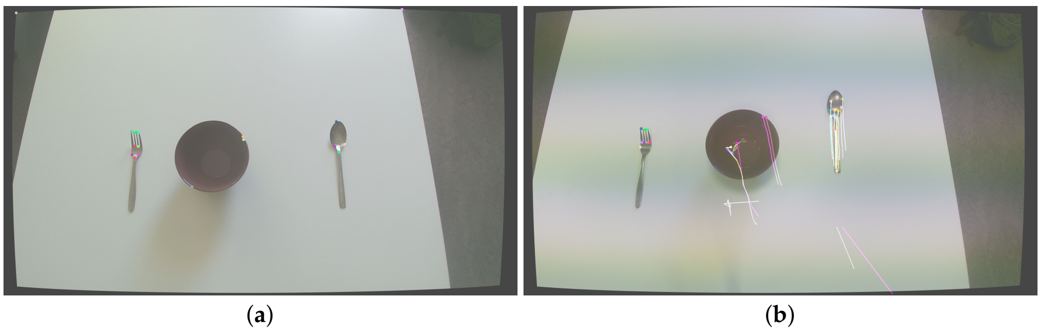


Figure 7. Sparse Optical Flow with Lucas–Kanade: (**a**) Shi–Tomasi features [34] to track, (**b**) Optical Flow with colored curves.

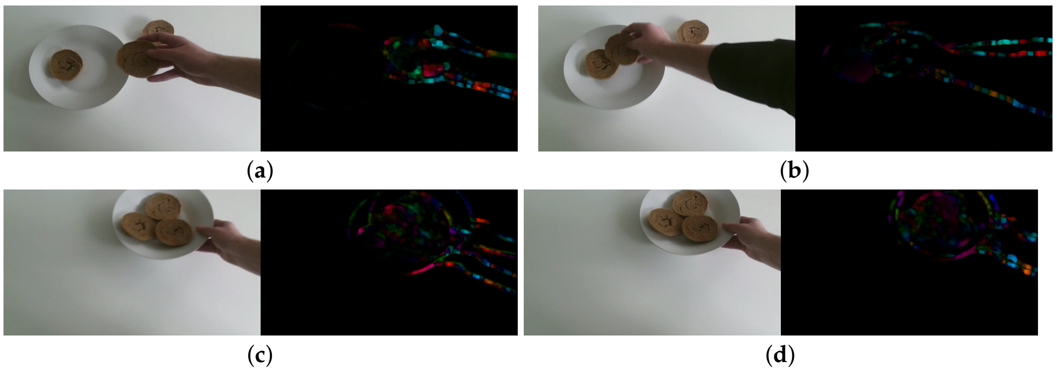


Figure 8. Visualization of Dense Optical Flow with Lucas–Kanade: (**a**) moving ntakos, (**b**) moving ntakos inside the plate, (**c**) moving plate with ntakos, (**d**) moving plate with ntakos.

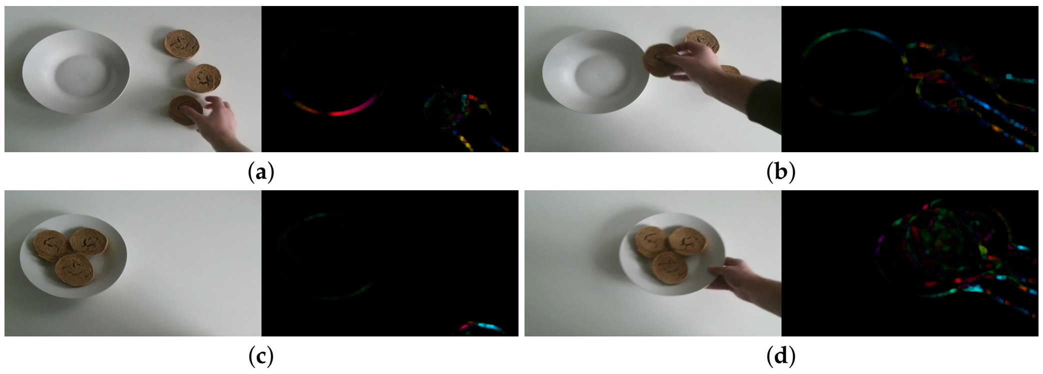


Figure 9. Visualization of Dense Optical Flow with Farneback: (**a**) moving ntakos, (**b**) moving ntakos inside the plate, (**c**) no move, (**d**) moving plate with ntakos.

3.4. Recipe Decision Logic

In the previous sections, we described the steps for acquiring all the essential sensory data that will be fed to the decision logic component. The workflow of this component is shown in Figure 10. Individual steps are discussed in subsequent sections.

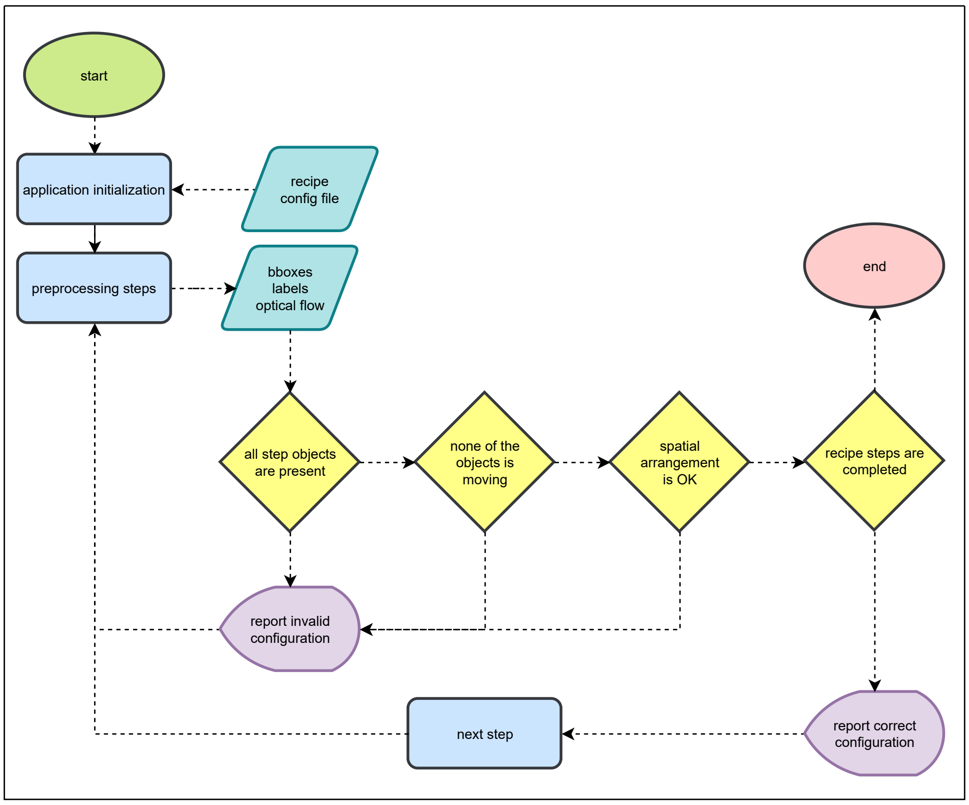

start
application initialization
recipe config file
preprocessing steps
bboxes labels optical flow
end
all step objects are present
none of the objects is moving
spatial arrangement is OK
recipe steps are completed
report invalid configuration
next step
report correct configuration

Figure 10. Workflow of our approach.

3.4.1. Recipe Specification

The recipe specification is included in a configuration file, which is a keystone regarding the execution of a cooking procedure. It contains every important information that the system needs in order to validate the facticity of this procedure. The configuration files corresponding to our two recipes are shown in Listings 1–2.

Listing 1. "Greek Ntakos" recipe specification.

```
{
 "ingredients": [ "plate", "oil", "ntakos", "ntakos", "ntakos","trimmed_tomatoes","
     feta" ],\
 "recipe": "Greek Ntakos",
 "spatial configurations": {
   "oil": "left_above_feta",
   "trimmed_tomatoes": "right_below_oil",
   "feta": "left_to_ntakos"
 },
 "cooking configurations": {
   "step 1": [ "plate_with_ntakos", "oil", "trimmed_tomatoes",  "feta" ],
   "step 2": [ "plate_with_ntakos", "oil", "glass_bowl",  "feta" ],
   "step 3": [ "plate_with_ntakos", "oil", "glass_bowl", "small_plate" ],
   "step 4": [ "plate_with_ntakos", "plastic_cup", "glass_bowl", "small_plate" ]
 },
  "cooking messages": {
    "step 1": "Add ntakos in plate",
    "step 2": "Add diced tomatoes on top of ntakos",
    "step 3": "Add feta on top of diced tomatoes",
    "step 4": "Add oil on top of everything",
    "step 5": "Recipe followed successfully! "
  }
}
```

Listing 2. "Fava" recipe specification.

```
{
 "ingredients": [ "fava", "oil", "salt", "lemon","onion" ],
 "recipe": "fava",
 "spatial configurations": {
   "oil": "left_above_fava",
   "spoon": "right_to_fava",
   "fava": "left_to_spoon"
 },
 "cooking configurations": {
   "step 1": [ "fava", "oil","salt","onion_bowl","lemon" ],
   "step 2": [ "fava", "plastic_cup","salt","onion_bowl", "lemon" ],
   "step 3": [ "fava", "plastic_cup","plastic_bowl","onion_bowl", "lemon" ],
   "step 4": [ "fava", "plastic_cup","plastic_bowl","onion_bowl", "squished_lemon" ]
 },
 "cooking messages": {
   "step 1": "Add onion in fava",
   "step 2": "Add oil on top",
   "step 3": "Add salt",
   "step 4": "Add lemon on top",
   "step 5": "Recipe followed successfully!"
 }
}
```

As mentioned in Secion 1, the cooking procedure that is examined in this research work is the correct execution of a recipe and the required spatial arrangement of the ingredients/utensils comprising it. At the same time, the system assists the user to execute the recipe by providing corresponding messages, which also exist in the configuration file. Collectively, the configuration file contains the features below:

- The name of the recipe in study;
- A list of the ingredients that are vital for completion of the recipe;
- The spatial configurations of the ingredients/utensils;
- The cooking configurations of each step of the recipe in term of ingredients;
- The necessary messages, pointed to the user to assist him in completing the recipe successfully.

3.4.2. Spatial Arrangement

Apart from existence of ingredients and utensils in the scene, knowing their relevant positions with respect to each other is also valuable information. This is suited for recipes which have such specific requirements, e.g., if serving instructions are included in the recipe. Serving instructions may come in the form of having a dip or a sauce that needs to be placed in a specific position next to a cooked meal. Instructions can also take into account particularities of the cook such as being left-handed or ambidextrous. The relative position of utensils is also useful in situations where there are safety measures that need to be applied in the cooking scene (see cooking ring). Utensils on high temperatures should be kept apart from flammable materials or cooking appliances.

In our implementation, spatial arrangement is defined as the position of an object, relative to another object. We approached this relative position as placement of an object in one of eight directions around the reference object; top, bottom, left, right and the intermediate ones. The method to determine whether the relative position of the object is right or left to another object, by calculating the angle between the respective vectors of these objects. The key idea behind this is that a coordinate system can be assumed in a frame even if it does not exist in mathematical terms. For example, the horizontal line of an object's centre to the end of the frame can easily be considered as an abscissa. In view of this fact, the angle can be calculated, and therefore, the relative position of two objects, by calculating the angle between the abscissa and the vector that is formed if the centres of the two objects are connected. The native coordinate system of OpenCV had to also be transformed counter-clockwise.

3.4.3. The Cooking Assistant

The role of the cooking assistant is to guide the user into carrying out a recipe and validate if each step of the recipe was executed correctly. Each recipe is modeled as intermediate steps, much like a cookbook that contains textual visualization for how to cook a recipe. The intermediate steps or cooking configurations as referred to the configuration file, is nothing more than lists with ingredients and utensils that comprise the scene at a certain point. Given the specification about the recipe, the cooking assistance process for a specific recipe step will be broken down into three parts.

The first part is about the correctness of the ingredients, meaning that all the ingredients that are related to the current step are present in the scene. This will be deduced by detecting the ingredients in the scene and comparing them with the ingredients that should exist for the execution of the specific recipe. The second part is about motion consideration of the individual detected objects. The information about the motion of an object functions as a safeguard for the substantiation of the sequence of steps. By obtaining definitive information on the ingredient that was meant to be added at a certain step of the recipe has indeed moved, the system reduces some uncertainty about the existing and future states. The third part is about the spatial configuration that the recipe ingredients or utensils should respect. If all of the above circumstances are fulfilled, the recipe step is considered as complete and the user is instructed to advance to the next recipe step. Otherwise, the user is informed with appropriate messages. The recipe is considered accomplished if all individual steps have been finished properly.

4. Experiments

The present section shows experimentation that was conducted in this research, and the corresponding results, both quantitative and qualitative. The qualitative results include demonstration of individual parts of our pipeline, e.g., object detection, spatial arrangement, etc., in the form of sequences of images originally extracted from videos that were recorded during execution. More specifically, Secions 4.1–4.2 present the behavior of the system while executing each of the two recipes. Preliminary experimentation and metrics corresponding to the individual components of the system are included in Secions 4.3–4.4.

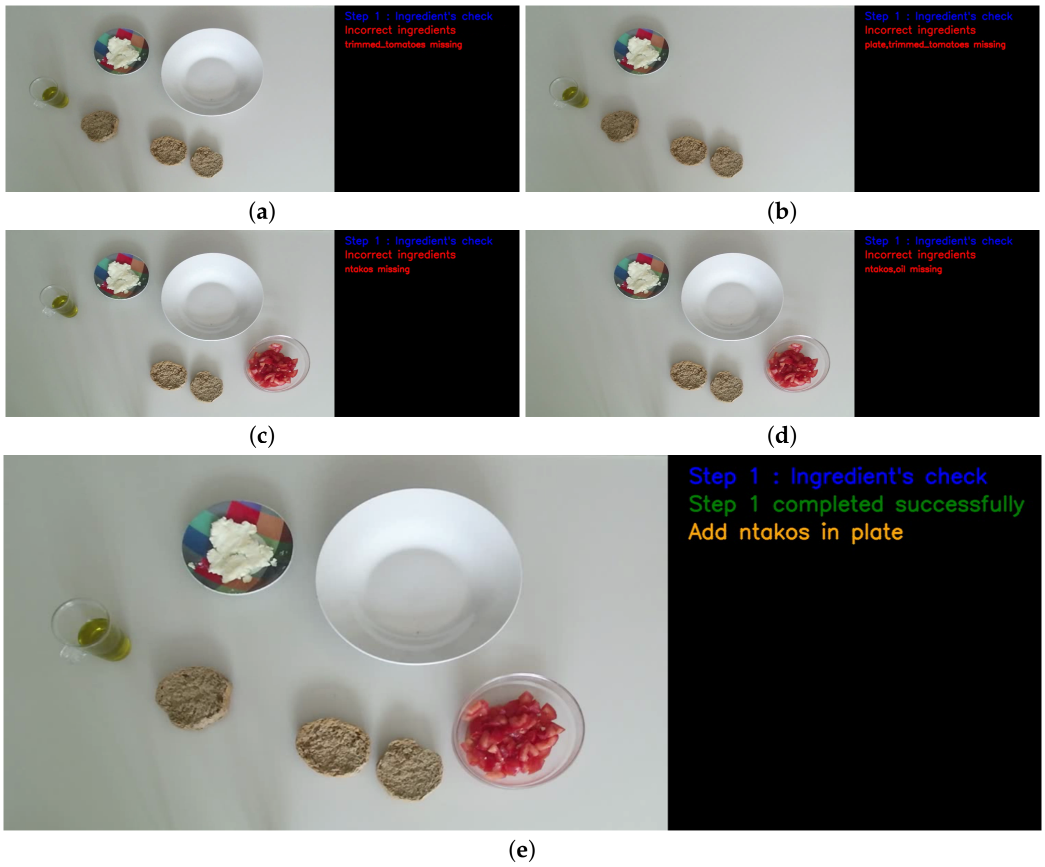
4.1. Checking for Correct Ingredients

Each recipe requires a list of utensils and ingredients, such that the user of the system/cook can carry out the recipe successfully. The system provides a preliminary utility, which informs the user whether or not all utensils/ingredients are present in the cooking scene. If something is missing, the system can understand and inform the user with an appropriate message.

The process is completed when:

1. Everything found in the scene matches the list of ingredients and utensils explained in Secion 3.4.1;
2. The first step of the recipe, as explained in Secion 4.2, is finished successfully.

An example of the above process is presented in Figure 11. Even if all the ingredients/utensils are contextually appropriate, the system will carry on with examining the aforementioned correctness until the first step of the recipe is validated. There is a two-fold explanation for this kind of behavior. First and foremost, the human error is taken into account: for example, if the user/cook mistakenly adds a different ingredient in the first step, the state (panel messages, etc.) will return to a state where the ingredients are incorrect once again, indicating that something went wrong. Furthermore, it takes into consideration a possible compromise or change of state of the food ingredients. This feature adds resilience and adaptability to the system, since every state is reversible, depending solely on the condition of the food ingredients and utensils.


(**a**)
Step 1 : Ingredient's check
Incorrect ingredients
trimmed_tomatoes missing

(**b**)
Step 1 : Ingredient's check
Incorrect ingredients
plate,trimmed_tomatoes missing

(**c**)
Step 1 : Ingredient's check
Incorrect ingredients
ntakos missing

(**d**)
Step 1 : Ingredient's check
Incorrect ingredients
ntakos,oil missing

(**e**)
Step 1 : Ingredient's check
Step 1 completed successfully
Add ntakos in plate

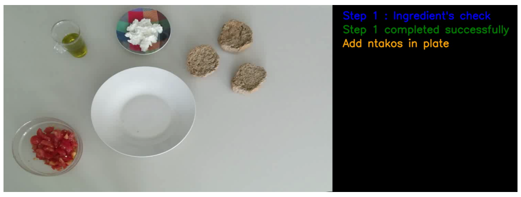
Figure 11. The process of checking for the correct ingredients: (**a**) tomatoes missing, (**b**) plate and tomatoes missing, (**c**) ntakos missing, (**d**) ntakos and oil missing, (**e**) all the ingredients are present.

4.2. Recipes

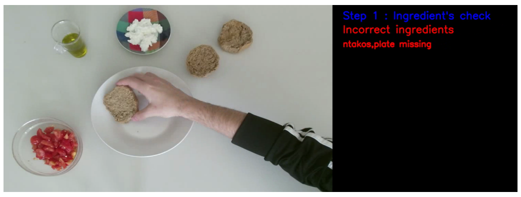
By acquiring the correct ingredients/utensils (see Figure 11), the user can proceed in the realization of the recipe. The ingredients are added one at a time, more or less like in real life. The first message that pops up on the screen is to add ntakos to the plate (see Figure 12a). The user moves the ntakos to the plate (see Figure 12b) and the cooking assistant calculates if the procedure is done accordingly to the plan. The message that pops up on the screen is "Ntakos added in plate" as well as the next message, which is to add the next ingredient. In this case, it is "Add diced tomatoes on top of plate" (see Figure 12c). After adding tomatoes on top of ntakos successfully, the messages inform the user about the outcome and instructs them to add the next ingredient, which is Feta (see

Figure 12e). In the same manner, Feta is added on top (see Figure 12g) and lastly oil on top of everything (see Figure 12i). Due to the fact that adding oil was the final step, a message pops up, informing the user that the recipe was executed successfully.

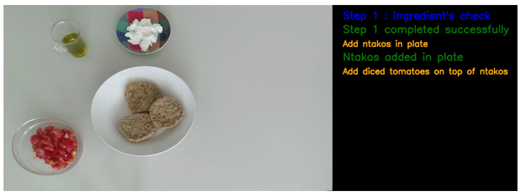
Similarly, the execution of Fava is shown in Figure 13.


Step 1 : Ingredient's check
Step 1 completed successfully
Add ntakos in plate

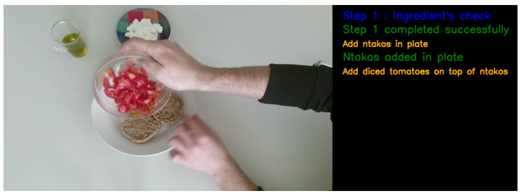
(**a**)


Step 1 : Ingredient's check
Incorrect ingredients
ntakos,plate missing

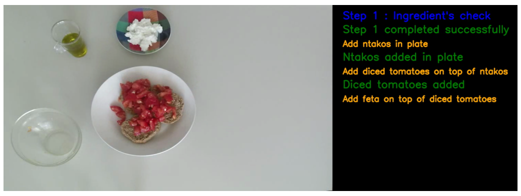
(**b**)


Step 1 : Ingredient's check
Step 1 completed successfully
Add ntakos in plate
Ntakos added in plate
Add diced tomatoes on top of ntakos

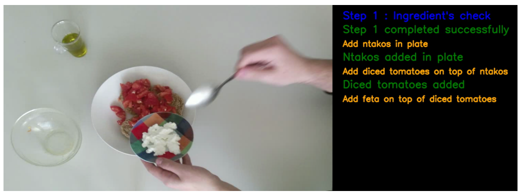
(**c**)


Step 1 : Ingredient's check
Step 1 completed successfully
Add ntakos in plate
Ntakos added in plate
Add diced tomatoes on top of ntakos

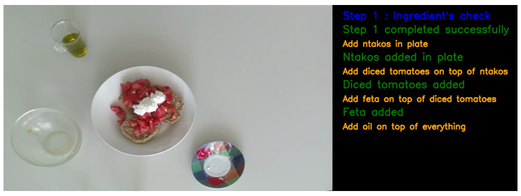
(**d**)


Step 1 : Ingredient's check
Step 1 completed successfully
Add ntakos in plate
Ntakos added in plate
Add diced tomatoes on top of ntakos
Diced tomatoes added
Add feta on top of diced tomatoes

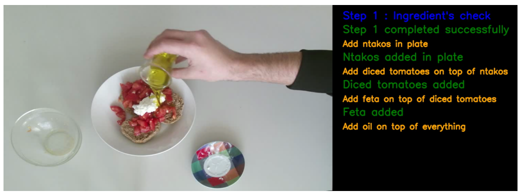
(**e**)


Step 1 : Ingredient's check
Step 1 completed successfully
Add ntakos in plate
Ntakos added in plate
Add diced tomatoes on top of ntakos
Diced tomatoes added
Add feta on top of diced tomatoes

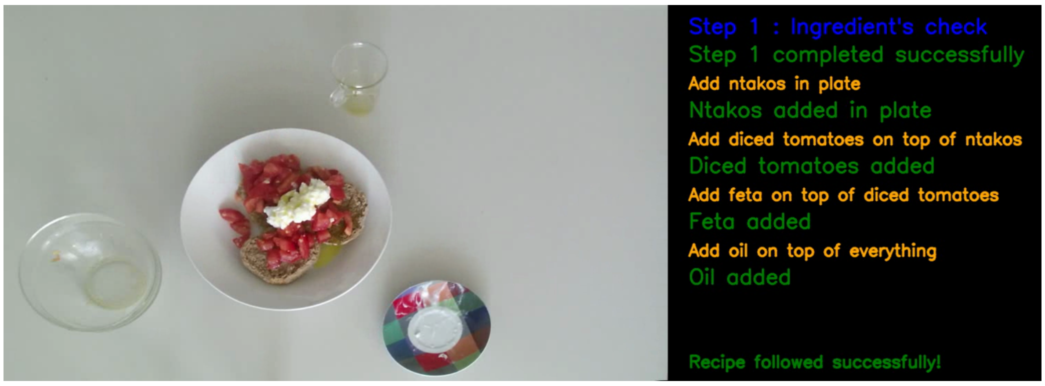
(**f**)


Step 1 : Ingredient's check
Step 1 completed successfully
Add ntakos in plate
Ntakos added in plate
Add diced tomatoes on top of ntakos
Diced tomatoes added
Add feta on top of diced tomatoes
Feta added
Add oil on top of everything

(**g**)


Step 1 : Ingredient's check
Step 1 completed successfully
Add ntakos in plate
Ntakos added in plate
Add diced tomatoes on top of ntakos
Diced tomatoes added
Add feta on top of diced tomatoes
Feta added
Add oil on top of everything

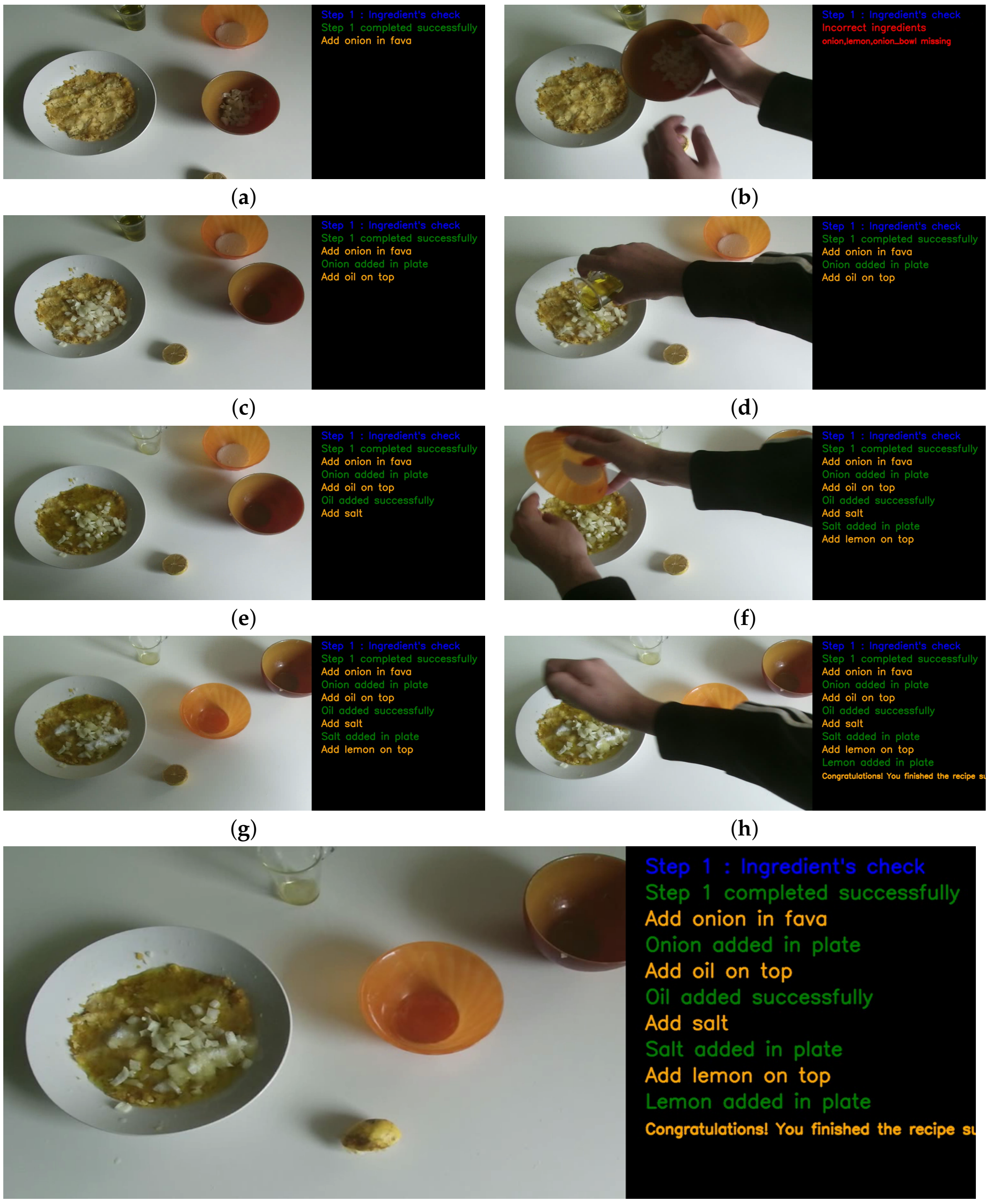
(**h**)


Step 1 : Ingredient's check
Step 1 completed successfully
Add ntakos in plate
Ntakos added in plate
Add diced tomatoes on top of ntakos
Diced tomatoes added
Add feta on top of diced tomatoes
Feta added
Add oil on top of everything
Oil added

Recipe followed successfully!

(**i**)

Figure 12. Cooking procedure of Greek Ntakos: (**a**) first step: add ntakos, (**b**) moving ntakos; (**c**) second step: add diced tomatoes, (**d**) moving tomatoes; (**e**) third step: add feta, (**f**) moving feta; (**g**) fourth step: add oil, (**h**) moving oil; (**i**) successful execution of the recipe.



(**a**)

```
Step 1 : Ingredient's check
Step 1 completed successfully
Add onion in fava
```

(**b**)

```
Step 1 : Ingredient's check
Incorrect ingredients
onion,lemon,onion_bowl missing
```

(**c**)

```
Step 1 : Ingredient's check
Step 1 completed successfully
Add onion in fava
Onion added in plate
Add oil on top
```

(**d**)

```
Step 1 : Ingredient's check
Step 1 completed successfully
Add onion in fava
Onion added in plate
Add oil on top
```

(**e**)

```
Step 1 : Ingredient's check
Step 1 completed successfully
Add onion in fava
Onion added in plate
Add oil on top
Oil added successfully
Add salt
```

(**f**)

```
Step 1 : Ingredient's check
Step 1 completed successfully
Add onion in fava
Onion added in plate
Add oil on top
Oil added successfully
Add salt
Salt added in plate
Add lemon on top
```

(**g**)

```
Step 1 : Ingredient's check
Step 1 completed successfully
Add onion in fava
Onion added in plate
Add oil on top
Oil added successfully
Add salt
Salt added in plate
Add lemon on top
```

(**h**)

```
Step 1 : Ingredient's check
Step 1 completed successfully
Add onion in fava
Onion added in plate
Add oil on top
Oil added successfully
Add salt
Salt added in plate
Add lemon on top
Lemon added in plate
Congratulations! You finished the recipe su
```

(**i**)

```
Step 1 : Ingredient's check
Step 1 completed successfully
Add onion in fava
Onion added in plate
Add oil on top
Oil added successfully
Add salt
Salt added in plate
Add lemon on top
Lemon added in plate
Congratulations! You finished the recipe su
```

Figure 13. Cooking procedure of fava: (**a**) first step: add onion, (**b**) moving onion; (**c**) second step: add oil, (**d**) moving oil; (**e**) third step: add salt, (**f**) moving salt; (**g**) fourth step: add lemon, (**h**) moving lemon; (**i**) successful execution of the recipe.

4.3. Spatial Arrangement of the Objects

Figure 14 depicts the relative position of each object to each other. The system generates each position accordingly in real time (see Secion 3.4.2). For example, in Figure 14a the plate with fava is right from the oil and left from the spoon. Similarly, the spoon is right from the oil and the plate with fava. The cup with the oil is left from spoon and left from fava. In addition to this, the configuration file, explained in Secion 3.4.1, contains the required positions of the ingredients/utensils used in the specific cooking procedure. By comparing the calculated relative position to that of the configuration file, the system determines if these match each other. The results are shown in Figure 15.

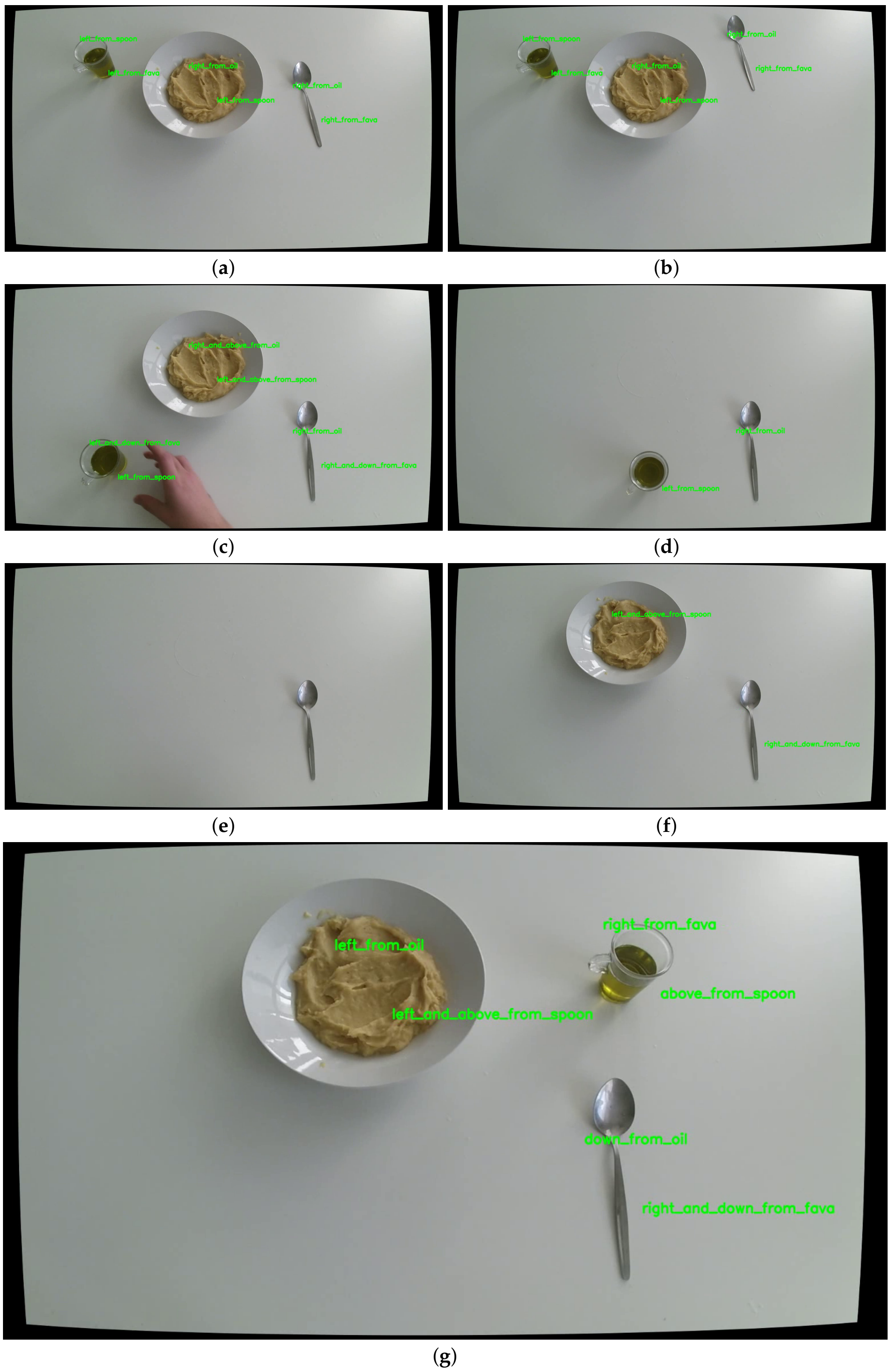


Figure 14. The spatial arrangement of objects at successive frames: (**a**) initial state, (**b**) spoon has been moved but is still in the right of the plate, (**c**) both oil and spoon have moved downwards, (**d**) plate has been removed from the scene and the labels have adapted, (**e**) only the spoon appears in the scene so no information is shown, (**f**) plate has reappeared, (**g**) all three objects are in the scene and appropriate relative positions are shown.

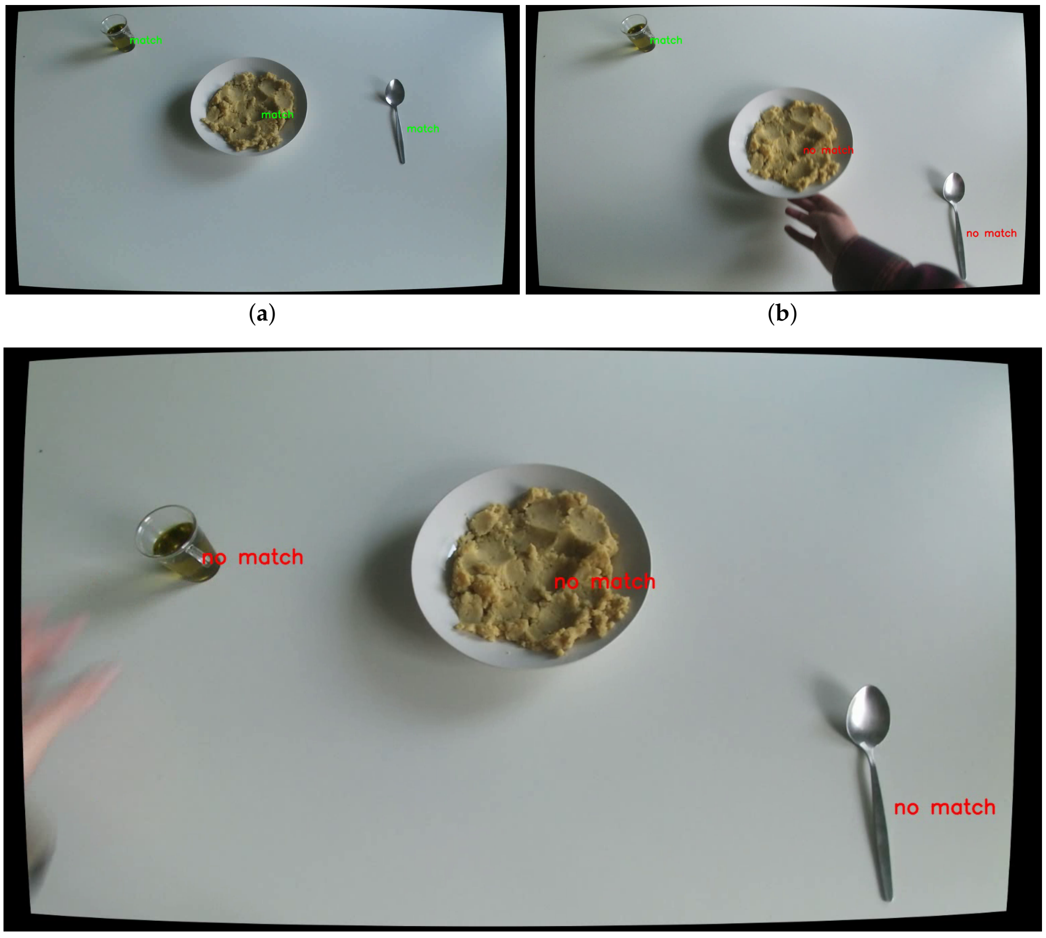


Figure 15. Assessment of the spatial configuration for "fava" recipe according to specification 2: (**a**) correct arrangement of objects, (**b**) some or all objects are not placed correctly, (**c**) objects are not placed correctly.

4.4. Metrics

In information retrieval, the instances are documents and the task is to return a set of relevant documents given a search term. Precision is the number of relevant documents retrieved by a search divided by the total number of documents retrieved by that search. In a classification task, the precision for a class is the number of true positives, otherwise the number of items correctly labeled as belonging to the positive class, divided by the total number of elements labeled as belonging to the positive class, i.e., the sum of true positives and false positives, which are items incorrectly labeled as belonging to the class (see Equation (1)).

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive} \tag{1}$$

4.4.1. Optical Flow Metrics

To assess the performance of the algorithms mentioned in Secion 3.3, an experiment was designed. The experiment included three objects, namely a plate, a glass bowl and a small plastic cup. Each object was resettled through eight various positions with a steady force and within an equal distance (see Figure 6).

Ground truth was established in order to cross-validate the true and false positives of each method. Precision was then calculated via the scikit-learn, an open-source Python ML tool for predictive data analysis [38]. The purpose of measuring the precision of each optical flow method was double-edged; on the one hand, it measured the performance of each method, establishing which is the best of these methods. On the other hand, it also serves as a conclusive way of establishing the best threshold for each method and later uses this threshold in the program that quantifies motion.

The thresholds tested were devised in a heuristic way by applying a 10^x, where x equals 0.5. Table 5 shows the results. As shown, Farneback Dense Optical Flow achieved perfect precision for medium to larger thresholds and, therefore, we utilized this method in the software.

Table 5. Precision of each method grouped by threshold.

Methods	**Thresholds**				
	3.16	**10**	**31.6**	**100**	**316.22**
Lucas–Kanade Sparse Optical Flow	0.72	0.67	0.77	0.75	0.83
Lucas–Kanade Dense Optical Flow	0.0	0.72	0.8	0.81	0.82
Farneback Dense Optical Flow	0.8	0.86	1	1	0.0

4.4.2. Object Detection Metrics

For the purpose of evaluation of the object detection module we conducted a short experiment using (a) a pre-trained detector and (b) our detector built on the "Greek ntakos" objects dataset. The motivation for this experiment was to evaluate what someone can expect from an off-the-shelf detector in a relatively easy task and a comparison against the detector, which we trained on the custom objects. The pre-trained model was trained on the LVIS [30] dataset; therefore, we chose only three objects, which appear in this dataset and our dataset. These objects were the glass bowl, the plastic cup and the plate. Each object was imaged at nine different positions on a table.

As shown in Figure 16, two of the objects were transparent, which added extra complexity but at the same time manifested the necessity of training a custom object detector. Figure 16 cements some of the notions discussed here. Overall, the precision of the pre-trained model was very poor. The average precision was 5.7% for the cup, and 0.0 % for the glass bowl and the plate. On the other hand, the performance of the custom object detector was outstanding in all three of the objects and in most of the cases. Even though detecting a transparent object or even a white plate on a white table remains an intractable task, the custom detector achieved high scores. The detector achieved 98.812% true positives for the plate, 95.198% for glass bowl and 83.647% for plastic cup, succeeding in all three of the criteria, which fulfills the "true positive" classification: confidence score > threshold. The predicted class matches the class of a ground truth; the predicted bounding box had an IoU greater than the threshold (e.g., 0.5) with the ground-truth. Figure 17 shows some of the successfully annotated instances of the custom detector. Although it is understandable that training a detector on these custom objects is beneficial over the off-the-shelf detector, at the same time, it is also experimentally confirmed that such detector performed poorly in a real-world situation. Therefore, we conclude that even if a detector was trained on a very large and diverse dataset, possibly including all categories of interest, there is not prospect for direct application and we suggest training a detector on a custom dataset.

The same custom detector and another one trained on the "Fava"' dataset were utilized in individual experiments including unseen images during a real recipe preparation. For these experiments, we did not conduct a formal evaluation, however we realized that the algorithm was able to detect almost all of the objects and at all video frames correctly. The few exceptions were when the algorithm was presented with transparent objects (class missmatch) and objects that appeared inside other objects (false positives).

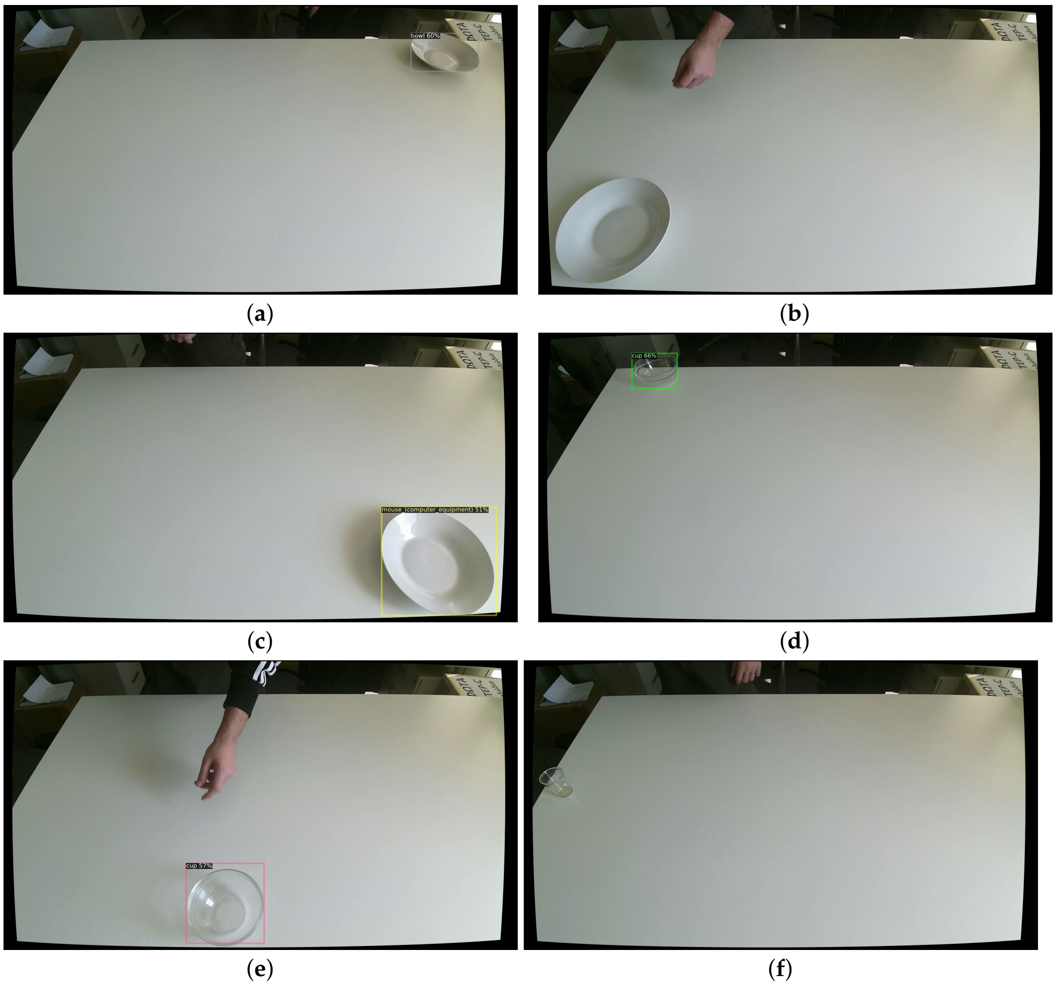


Figure 16. Performance of the off-the-self detector in three classes: (**a**,**d**,**e**) show that detector mismatched the classes, (**b**,**f**) show that the detector did not achieve any detection of object for the plate and the plastic cup. In (**c**) the detector misrecognized the object as a computer mouse rather than as a plate.

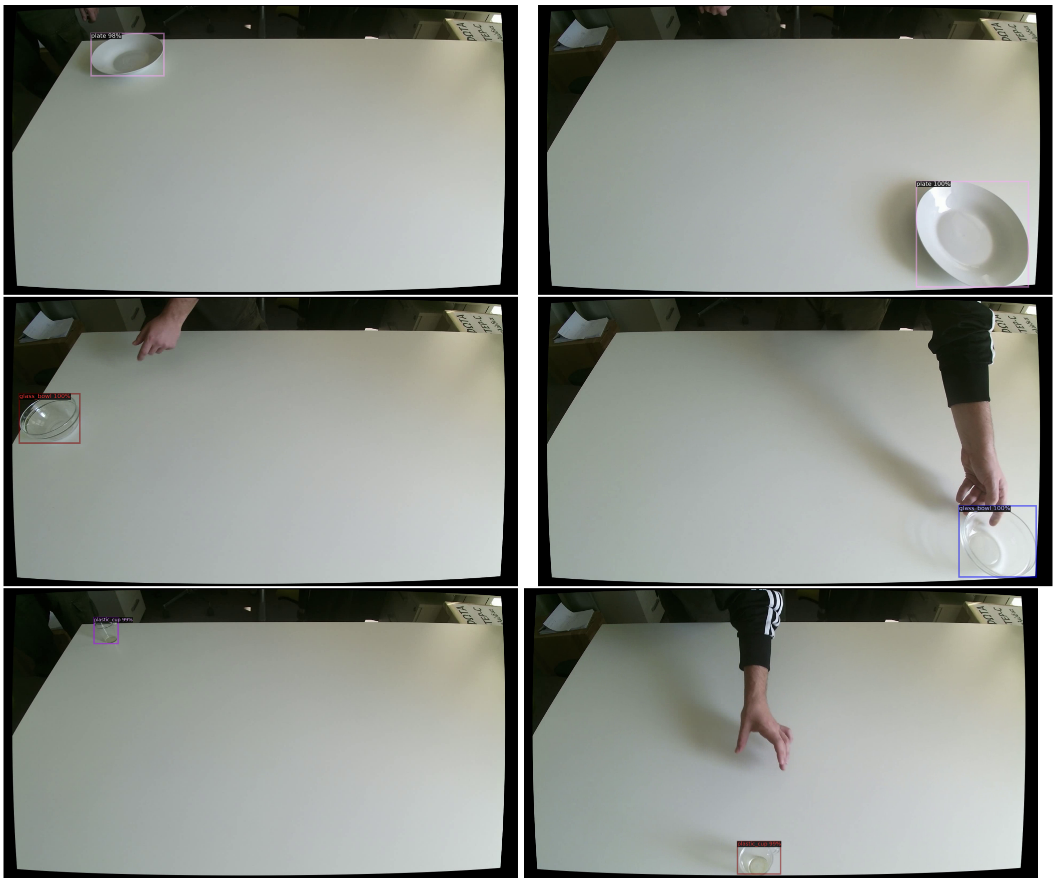


Figure 17. Performance of the custom detector in the same three classes: successfully annotated instances of the custom detector.

5. Conclusions

The current work introduces a cooking assistance system that provides the user with guidance in the accomplishment of a cooking activity in terms of a recipe and its correct execution. The potential of this work lies in the fact that each recipe is imported in the system in the form of a single configuration file. The system, by construing the configuration file, can provide the user with guidance for carrying out a recipe through appropriate messages that appear in a panel specifically designed for the user. The system can validate the correctness of each step by detecting used ingredients and the corresponding utensils and calculating each respective motion in each step.

The system also supports the spatial arrangement of objects such as utensils and food ingredients in terms of where each one is located to another. The configuration file contains, in the same fashion as the recipe, the correct position of an object relative to another object. The system can validate if the real position of an object is correct by calculating the respective geometric proportions.

The goal of this work is the integration in ICS-FORTH's Intelligent Home and aspires to be used in the ambient facilities of the Intelligent Kitchen. Alongside other pre-existing systems, it focuses on supporting and guiding the cooking process, decreasing the perceived complexity of unfamiliar recipes, increasing the user's confidence about the success of intermediate steps and finally reducing the user's inhibitions of using technology in a kitchen.

Based on the quantitative and qualitative evaluation of the methods used, this work seems to be a very promising cooking assistant. The results show that the system behaves robustly in low-confidence scenarios and that, in most of the cases, it succeeds both in identifying the correctness of a recipe but also establishing a satisfactory way of providing the necessary advice to the user. In contrast to other systems, we lay on the side of real-time feedback, which is missing [13] or partially missing [14] from recent approaches, but also on the side of easier deployment, since our approach does not depend on specialized hardware [11,14]. However, since all of these approaches concentrate on different aspects of recipe preparation, we conclude that future efforts should consider the outcomes from all of them.

It should be conclusive for the reader that, when it comes to the discussion of the future work, there is a vast amount of things that could have been done differently or different methods that could have been used in the given scenarios. Nonetheless, at this point, it should be obvious that none of these methods could guarantee a better result, due to the inherent difficulties of computer vision as a field and the uncertainty it entails. Despite the fact that some of the challenges faced still remain intractable till now, there are some improvements and additions that the system could benefit from. Specifically, handling occlusion, illumination variations, transparent ingredients such as water, corrupted pixels and complex background are important for better results, especially since this work was performed in a natural human setup. Extra sensors such as accelerometers, weight sensors or wearable cameras could be deployed to enhance the hypothesis that an object moves or that an ingredient empties. Activity recognition is a task that could further advance this work by recognizing specific cooking activities, such as cutting or peeling, or to interpret human gestures. The acceleration of some of the methods used could also signify better processing and better results. Specifically, deploying processing in CUDA or applying multi-threading could lead to better results in Optical Flow. Last but not least, it goes without saying that there is a necessity of a UI that engulfs the current work and provides the user with an interactive, context-aware, multi-modal, multi-sensory, user-adaptive and intuitive way of preparing their meals.

Author Contributions: Conceptualization, G.K., N.P. and X.Z.; methodology, G.K., N.P. and X.Z.; software, G.K. and G.G.; validation, G.K. and G.G.; formal analysis, G.K. and G.G.; investigation, G.K.; resources, N.P. and X.Z.; data curation, G.K. and G.G.; writing—original draft preparation, G.K.; writing—review and editing, G.K. and G.G.; visualization, G.K. and G.G.; supervision, N.P. and X.Z.; project administration, N.P. and X.Z.; funding acquisition, N.P. and X.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This work has been conducted in the context of the Mingei project that has received funding from the European Union’s Horizon 2020 research and innovation program under grant agreement No 822336.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: The facilities for the creation of the dataset and experimentation were provided by the FORTH-ICS internal RTD Programme ’Ambient Intelligence and Smart Environments’.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

AAL	Ambient Assisted Living
AR	Activity Recognition
CV	Computer Vision
CNN	Convolutional Neural Network
FAIR	Facebook AI Research
FPN	Feature Pyramid Network
ML	Machine Learning
ViT	Vision Transformer

References

1. Johnson, L.C. Browsing the modern kitchen—A feast of gender, place and culture (part 1). *Gender Place Cult.* **2006**, *13*, 123–132.
2. Byrd, M.; Dunn, J.P. *Cooking through History: A Worldwide Encyclopedia of Food with Menus and Recipes*; ABC-CLIO: Santa Barbara, CA, USA, 2020; 2 vols.
3. Berk, Z. *Food Process Engineering and Technology*; Academic Press: Cambridge, MA, USA, 2018.
4. Stein, S.; McKenna, S.J. Towards recognizing food preparation activities in situational support systems. In Proceedings of the Digital Futures 2012 (3rd Annual Digital Economy All Hands Conference), Aberdeen, UK, 23–25 October 2012.
5. Stein, S.; McKenna, S.J. Combining embedded accelerometers with computer vision for recognizing food preparation activities. In Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing, Zurich, Switzerland, 8–12, September 2013; pp. 729–738.
6. Urabe, S.; Inoue, K.; Yoshioka, M. Cooking activities recognition in egocentric videos using combining 2DCNN and 3DCNN. In Proceedings of the Joint Workshop on Multimedia for Cooking and Eating Activities and Multimedia Assisted Dietary Management, Mässvägen, Stockholm, Sweden, 15 July 2018; pp. 1–8.
7. Surie, D.; Partonia, S.; Lindgren, H. Human sensing using computer vision for personalized smart spaces. In Proceedings of the 2013 IEEE 10th International Conference on Ubiquitous Intelligence and Computing and 2013 IEEE 10th International Conference on Autonomic and Trusted Computing, Vietri sul Mare, Italy, 18–21 December 2013; IEEE: Piscataway, NJ, USA, 2013; pp. 487–494.
8. Zhang, Z. Microsoft kinect sensor and its effect. *IEEE Multimed.* **2012**, *19*, 4–10.
9. Hashimoto, A.; Mori, N.; Funatomi, T.; Yamakata, Y.; Kakusho, K.; Minoh, M. Smart kitchen: A user centric cooking support system. In Proceedings of the IPMU 2008, Malaga, Spain, 22–27 June 2008; Volume 8, pp. 848–854.
10. Tran, Q.; Mynatt, E. Cook's collage: Two exploratory designs. In Proceedings of the Position Paper for the Technologies for Families Workshop at CHI 2002, Mineapolis MN, USA, 22 April 2002.
11. Neumann, A.; Elbrechter, C.; Pfeiffer-Leßmann, N.; Kõiva, R.; Carlmeyer, B.; Rüther, S.; Schade, M.; Ückermann, A.; Wachsmuth, S.; Ritter, H.J. "Kognichef": A cognitive cooking assistant. *KI-Künstliche Intell.* **2017**, *31*, 273–281.
12. Thuy, N.T.T.; Diep, N.N. Recognizing food preparation activities using bag of features. *Southeast Asian J. Sci.* **2016**, *4*, 73–83.
13. An, Y.; Cao, Y.; Chen, J.; Ngo, C.W.; Jia, J.; Luan, H.; Chua, T.S. PIC2DISH: A customized cooking assistant system. In Proceedings of the 25th ACM international conference on Multimedia, Mountain View, CA, USA, 23–27 October 2017; pp. 1269–1273.
14. Neumann, N.; Wachsmuth, S. HanKA: Enriched Knowledge Used by an Adaptive Cooking Assistant. In P*roceedings of the German Conference on Artificial Intelligence (Künstliche Intelligenz)*; Springer: Cham, Switzerland, 2022; pp. 173–186.
15. Bosch. Cookit-Smart Food Processor. Available online: https://cookit.bosch-home.com (accessed on 28 August 2022).
16. Thermomix. Thermomix® TM6®. Available online: https://www.thermomix.com/tm6 (accessed on 28 August 2022).
17. Bianco, S.; Ciocca, G.; Napoletano, P.; Schettini, R.; Margherita, R.; Marini, G.; Pantaleo, G. Cooking action recognition with iVAT: An interactive video annotation tool. In *Proceedings of the International Conference on Image Analysis and Processing*; Springer: Berlin/Heidelberg, Germany, 2013; pp. 631–641.
18. Rohrbach, M.; Amin, S.; Andriluka, M.; Schiele, B. A database for fine grained activity detection of cooking activities. In Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, 16–21 June 2012; IEEE: Piscataway, NJ, USA, 2012; pp. 1194–1201.
19. Jiang, P.; Ergu, D.; Liu, F.; Cai, Y.; Ma, B. A Review of Yolo algorithm developments. *Procedia Comput. Sci.* **2022**, *199*, 1066–1073.
20. Wu, Y.; Kirillov, A.; Massa, F.; Lo, W.Y.; Girshick, R. Detectron2. 2019. Available online: https://github.com/facebookresearch/detectron2 (accessed on 28 August 2022).
21. Xiang, L.; Echtler, F.; Kerl, C.; Wiedemeyer, T.; Lars.; hanyazou.; Gordon, R.; Facioni, F.; laborer2008.; Wareham, R.; et al. libfreenect2: Release 0.2 **2016**. Available online: https://doi.org/10.5281/zenodo.50641 (accessed on 28 August 2022).
22. Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*; Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., Garnett, R., Eds.; Curran Associates, Inc.: Vancouver, Canada, 2019; pp. 8024–8035.
23. Abadi, M.; Agarwal, A.; Barham, P.; Brevdo, E.; Chen, Z.; Citro, C.; Corrado, G.S.; Davis, A.; Dean, J.; Devin, M.; et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems, 2015. Software. Available online: tensorflow.org (accessed on 28 August 2022).
24. Albawi, S.; Mohammed, T.A.; Al-Zawi, S. Understanding of a convolutional neural network. In Proceedings of the 2017 International Conference on Engineering and Technology (ICET), Antalya, Turkey, 21–23 August 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 1–6.
25. Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv* **2020**, arXiv:2010.11929.
26. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster R-CNN: Towards real-time object detection with region proposal networks. *Adv. Neural Inf. Process. Syst.* **2015**, 91–99. Available online: https://www.bibsonomy.org/bibtex/2ceee12b1f8d61eed786dcdc15ffeb99f/nosebrain (accessed on 28 August 2022).
27. He, K.; Gkioxari, G.; Dollár, P.; Girshick, R. Mask R-CNN. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 2961–2969.

28. Xie, S.; Girshick, R.; Dollár, P.; Tu, Z.; He, K. Aggregated residual transformations for deep neural networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 1492–1500.
29. Lin, T.Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; Zitnick, C.L. Microsoft coco: Common objects in context. In Proceedings of the European conference on computer vision. Springer, 2014, pp. 740–755.
30. Gupta, A.; Dollar, P.; Girshick, R. LVIS: A Dataset for Large Vocabulary Instance Segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA 15–20 June 2019.
31. Damen, D.; Doughty, H.; Farinella, G.M.; Fidler, S.; Furnari, A.; Kazakos, E.; Moltisanti, D.; Munro, J.; Perrett, T.; Price, W.; et al. The EPIC-KITCHENS Dataset: Collection, Challenges and Baselines. *IEEE Trans. Pattern Anal. Mach. Intell. (TPAMI)* **2021**, *43*, 4125–4141. https://doi.org/10.1109/TPAMI.2020.2991965.
32. Sekachev, B.; Manovich, N.; Zhiltsov, M.; Zhavoronkov, A.; Kalinin, D.; Hoff, B.; TOsmanov.; Kruchinin, D.; Zankevich, A.; DmitriySidnev.; et al. opencv/cvat: v1.1.0, 2020. https://doi.org/10.5281/zenodo.4009388. Available online: https://github.com/opencv/cvat (accessed on 28 August 2022).
33. Lukezic, A.; Vojir, T.; ˇCehovin Zajc, L.; Matas, J.; Kristan, M. Discriminative correlation filter with channel and spatial reliability. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 6309–6318.
34. Bradski, G. The OpenCV Library. *Dr. Dobb'S J. Softw. Tools* **2000**, *11*, 120–123.
35. Itseez. Open Source Computer Vision Library. 2015. Available online: https://github.com/itseez/opencv (accessed on 28 August 2022).
36. Bruce D. Lucas, T.K. An Iterative Image Registration Technique with an Application to Stereo Vision. In Proceedings of the Imaging Understanding Workshop, Vancouver, Canada, 24–28 August 1981; pp. 121–130.
37. Farnebäck, G. Two-frame motion estimation based on polynomial expansion. In Proceedings of the Scandinavian Conference on Image Analysis, Halmstad, Sweden, 29 June–2 July 2003; Springer: Berlin/Heidelberg, Germany, 2003; pp. 363–370.
38. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.