Semantically annotated cooking procedures for an Intelligent Kitchen Environment

George Kondylakis

Thesis submitted in partial fulfillment of the requirements for the

Masters' of Science degree in Computer Science and Engineering

University of Crete
School of Sciences and Engineering
Computer Science Department
Voutes University Campus, 700 13 Heraklion, Crete, Greece

Heraklion, November 2021

Thesis Advisor: Assistant Prof. *Constantinos Stephanidis*

This work has been performed at the University of Crete, School of Sciences and Engineering, Computer Science Department.

The work has been supported by the Foundation for Research and Technology - Hellas (FORTH), Institute of Computer Science (ICS).

University of Crete
Computer Science Department

TitleSemantically annotated cooking processes in Smart Kitchen Environment

Thesis submitted by
Student George Kondylakis
in partial fulfillment of the requirements for the
Masters' of Science degree in Computer Science

THESIS APPROVAL

Author:	Student George Kondylakis
Committee approvals:	Professor 1 Assistant Professor, Thesis Supervisor
	Professor 2 Professor, Committee Member
	Professor 3 Professor, Committee Member
Departmental approval:	Antonios Argyros Professor, Director of Graduate Studies

Heraklion, November,2021

Abstract

Food preparation is one of the most essential tasks in daily life and it involves a large number of physical interactions between hands, utensils, ingredients, etc. The fundamental unit in the food preparation activity is the concept of the recipe. The recipe describes the cooking process, the way how to make a dish as 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. The complexity of such a task 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. With the huge advance of technology and the increasing interest on assisting humans in their lives, various attempts have been made to enhance the cooking process by integrating multimedia and embedding technologies in various stages. Computer vision can be used to establish spatial relationships and enable reasoning about the interaction between visual entities.

The Thesis introduces a cooking assistance system that provides the user with guidance in the accomplishment of a recipe, while validating the correct execution of each step. All the necessary information regarding the recipe and the spatial arrangement of ingredients/utensils is imported in the form of a single configuration file, which is construed and analyzed by the system. Specifiacally, the system designed in the current research work can provide the user with guidance in carrying out a recipe through the appropriate messages which appear in a panel, as a form of an interface with the user. It can also validate the overall correctness of the cooking procedure by utilizing the following information: (a) the correct sequence of steps which comprise a specific recipe, obtainable from the configuration file; (b) the detection of used ingredients and the corresponding utensils ; (c) the overall motion estimation of the scene; (d) the calculation of relative positions of objects involved in the cooking procedure, with respect to other objects.

In order to explore the benefits and limitations of the developed system, two evaluations have been conducted. The first experiment included various motion detection algorithms. The best algorithm achieved 100 % precision even for small lighting variations. The second experiment included detection of objects in unseen images during a real recipe preparation. The algorithm was able to detect almost all of the objects and at all video frames correctly. An exception was when the algorithm was presented with transparent objects (class missmatch) and objects that appeared inside other objects (false positives).

Περίληψη

Η διαδικασία της προετοιμασίας του φαγητού αποτελεί μια από τις σημαντικότερες διεργασίες της καθημερινότητας και περιλαμβάνει μια πληθώρα από αλληλεπιδράσεις με διάφορα αντικείμενα όπως τα υλικά του φαγητού, τα χέρια ή σκεύη. Στη διαδικασία αυτή, πολύ σημαντικό ρόλο παίζει η έννοια της συνταγής. Η συνταγή είναι αυτή που περιγράφει τη διαδικασία του μαγειρέματος, σαν διαδοχικά βήματα που πρέπει να ακολουθήσει κάποιος. Συχνά το να ακολουθήσει κάποιος μια συνταγή μπορεί να είναι αρκετά περίπλοκη διαδικασία, η οποία απαιτεί συντονισμό κινήσεων, επιτήρηση και ταυτόχρονα εκτέλεση πολλαπλών εργασιών. Η πολυπλοκότητα αυτή εντείνεται από την ποικιλομορφία που μπορεί να έχουν κάποιες συνταγές και από το γεγονός ότι ένα άτομο μπορεί να εκτελέσει την ίδια συνταγή με πολλούς διαφορετικούς τρόπους. Τα τελευταία χρόνια, λόγω της συνεχούς ανάπτυξης της τεχνολογίας και του αυξανόμενου ενδιαφέροντος για τη βελτίωση της ζωής του ανθρώπου, το ερευνητικό ενδιαφέρον στράφηκε και προς την κατεύθυνση της βελτίωσης των διεργασιών που αφορούν το μαγείρεμα. Ο τομέας της υπολογιστικής όρασης αποτελεί ένα από τα πιο ελπιδοφόρα πεδία στα οποία επικεντρώνεται αυτό το ενδιαφέρον, καθώς μέσω της υπολογιστικής όρασης δίδεται η δυνατότητα για υπολογισμό χωρικών σχέσεων μεταξύ των αντικειμένων και ανίχνευση λογικών αλληλεπιδράσεων μεταξύ οντοτήτων που λαμβάνουν χώρα στη διαδικασία του μαγειρέματος.

Η παρούσα εργασία περιγράφει τη δημιουργία ενός συστήματος που δρα επικουρικά στο χρήστη που μαγειρεύει, προσφέροντας του καθοδήγηση στην εκτέλεση μιας συνταγής, ενώ ταυτόχρονα αναλύει την ορθότητα της εκτέλεσης αυτής της συνταγής μέσω συστημάτων υπολογιστικής όρασης. Οι απαραίτητες πληροφορίες που αφορούν τόσο την συνταγή όσο και την χωρική διάταξη των αντικειμένων, περιλαμβάνονται εξ ολοκλήρου σε ένα αρχείο το οποίο περιέχει τις παραμέτρους για όλα τα παραπάνω και αναλύεται από το σύστημα.

Συνοπτικά, το παρόν σύστημα υποστηρίζει την καθοδήγηση του χρήστη στην εκτέλεση μιας συνταγής μέσω της προβολής των κατάλληλων μηνυμάτων. Ταυτόχρονα ελέγχει την ορθότητα εκτέλεσης της μαγειρικής διαδικασίας αξιοποιώντας και αναλύοντας τις παρακάτω πληροφορίες: (α) την αλληλουχία των βημάτων εκτέλεσης της συνταγής, τα οποία περιέχονται στο αρχείο που περιέχει τις παραμέτρους μιας συνταγής, (β) την αναγνώριση των αντικειμένων που υπάρχουν στο χώρο, είτε αυτά είναι τα σκεύη ή υλικά που χρησιμοποιούνται για τη διεξαγωγή μιας συνταγής, (γ) την πληροφορία για το αν ένα αντικείμενο κινείται στο χώρο και (δ) την χωρική διάταξη των αντικειμένων και τη σχετική τους θέση ως προς ένα άλλο αντικείμενο ή σημείο αναφοράς. Για να αξιολογηθούν τα οφέλη και οι περιορισμοί του συστήματος, σχεδιάστηκαν δύο πειράματα. Το πρώτο πείραμα περιλάμβανε τρεις αλγορίθμους για την ανίχνευση και εκτίμηση της κίνησης. Ο καλύτερος από τους τρεις αλγορίθμους παρουσίασε 100 % ακρίβεια ακόμα και για μικρές αλλαγές στις συνθήκες φωτισμού. Το δεύτερο πείραμα περιλάμβανε ανίχνευση και αναγνώριση αντικειμένων σε άγνωστες εικόνες κατά τη διάρκεια της προετοιμασίας μιας συνταγής σε πραγματικές συνθήκες. Ο αλγόριθμος ήταν σε θέση να αναγνωρίσει και να ανιχνεύσει σωστά σχεδόν όλα τα

αντικείμενα και σε κάθε καρέ. Εξαίρεση αποτελεί η περίπτωση που χρησιμοποιήθηκαν διαφανή αντικείμενα, όπου παρατηρήθηκε η αναγνώριση διαφορετικής κλάσης για το αντικείμενο, καθώς και η περίπτωση που αντικείμενα περικλείονταν μέσα σε άλλα αντικείμενα, όπου παρατηρήθηκε αναγνώριση αντικειμένων τα οποία δεν υπήρχαν.

Acknowledgments

First of all, I would like to express my gratitude to my master thesis supervisor, Professor Constantine Stephanidis, who entrusted me and gave me the opportunity to be a member of the HCI laboratory. I would like to thank him for his support, assistance and guidance throughout my M.Sc. studies.

I am also grateful to the advisory committee members, Associate Professor Konstantinos Papagiannakis and Principal researcher Xenophon Zabulis, who helped me improve my work with their valuable feedback.

Dr. Zabulis, beside a committee member, was a close collaborator throughout the course of my studies and my partnership with ICS-FORTH. He introduced me to the world of computer vision and assisted me in my first steps. Through his guidance, I was able to carry through this research work and was equipped with a lot of knowledge about this particular discipline.

Furthermore, I would like to thank Dr. Nikos Partarakis and Dr. Asterios Leonidis for their help and guidance, whenever I needed them.

Next, I would like to thank my colleague and later friend, PhD student George Galanakis, who helped me a great deal during this research work. George was actively involved in many steps of this process and through his experience and expertise in computer vision and machine learning, he helped me carry on through tough times.

I need to thank the Department of Computer Science (CSD) of the University of Crete, and the Human Computer Interaction (HCI) Laboratory of the Institute of Computer Science (ICS) of the Foundation for Research and Technology - Hellas (FORTH) for equipping me with the means and providing a highly motivational working environment to achieve my research goals.

Finally, I would like to express my deepest gratitude to my parents, Vasilis and Maria, and my friends Iraklis, Maria, Anna, Stayros, Andriani who gave me their unconditional love and support and equipped me with the courage I needed to achieve my goals throughout my master studies.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Cooking is a very important aspect of the everyday life, task that usually requires a lot of practice and experience while at the same time is very time-consuming. Cooking takes place inside the kitchen, which according to L. C. Johnson [6], *"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 such a task 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 around the culinary art. A recipe is 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 BC and were recorded on cuneiform tablets found in Mesopotamia[18]. Other early written recipes date from approximately 1600 BC and come from an Arcadian tablet from southern Babylonia[18]. There are also works in ancient Egyptian hieroglyphs depicting the preparation of food.

With the huge advance of technology and the increasing interest on enhancing humans in their regular lives, cooking recipes started being digitalized 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 endeavour 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, Robotics and Machine Learning started venturing on the cooking assistance drastically. These fields provided insight in some of the most unexcogitable problems while at the same facilitated the development of more realistic systems which could nonetheless address more comprehensively the cooking activity process.

Computer vision for human sensing has traditionally focused on using RGB image. Theoretically, fully automatic tools that integrate computer vision 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.

This current study introduces an assistive cooking system that provides to the user guidance in the accomplishment of a cooking activity via Computer Vision and Machine Learning. The potential of this work lies in the fact that cooking activities can be monitored and thus modelled in a dynamical infrastructure which does not rest on the use of cumbersome sensors or hardware, and 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 mean by any means 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's confidence about the success of intermediate steps and finally reduce the user's inhibitions of 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 like utensils and food ingredients in terms of where each one is located 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 its correct execution. Each recipe is imported in 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 rime provide the user with advice on how to carry on with 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.

Chapter 2

Related Work

Many attempts have been made to provide systems which assist the user in the culinary process, both at 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 Computer Vision and machine learning have to offer.

2.1 Literature Review

Combining Embedded Accelerometers with Computer Vision for Recognizing Food Preparation Activities

The contribution of this work [13] is a novel method for merging accelerometers and computer vision for activity recognition. The multi-modal activity recognition data-set used in this work includes more than 4.5 hours of annotated accelerometer and RGB-D video data (see 2.1). The evaluation of the experiments of the method and the comparison of various fusion methods also include a protocol for benchmarking.

Cooking Activities Recognition in Egocentric Videos Using Combining 2DCNN and 3DCNN

This research work [17] provides an accurate cooking activities recognition as the first step to realize cooking assistance service.The main focus is on the problem of cooking activities recognition in egocentric videos. The proposed method is based on hand detection, where the region around hand is extracted based on 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 (see 2.2). Therefore, a new data-set which

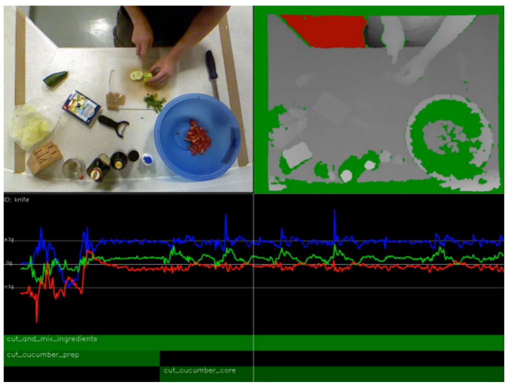


Figure 2.1: Data from an RGB-D camera and from accelerometers attached to kitchen objects

consists of 8 cooking-specific activities, like "Peeling", is proposed. In this activity hand motion could vary on whether a knife or a peeler is used.

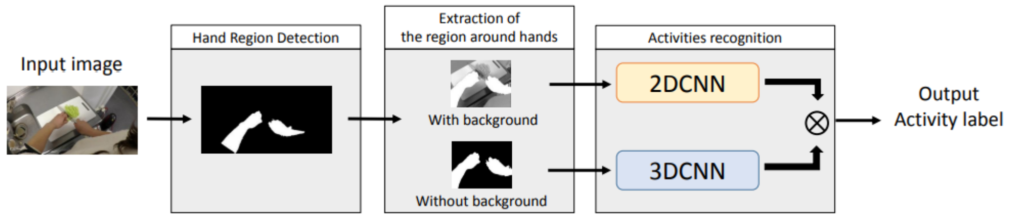


Figure 2.2: Cooking assistance service based on egocentric videos

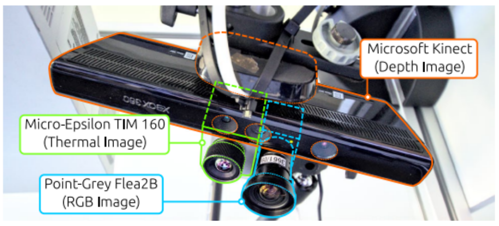
KogniChef: A Cognitive Cooking Assistant

KogniChef [7] is a smart kitchen environment and software framework that implements a cooking recipe assistant. The prototype (see 2.3a) 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.

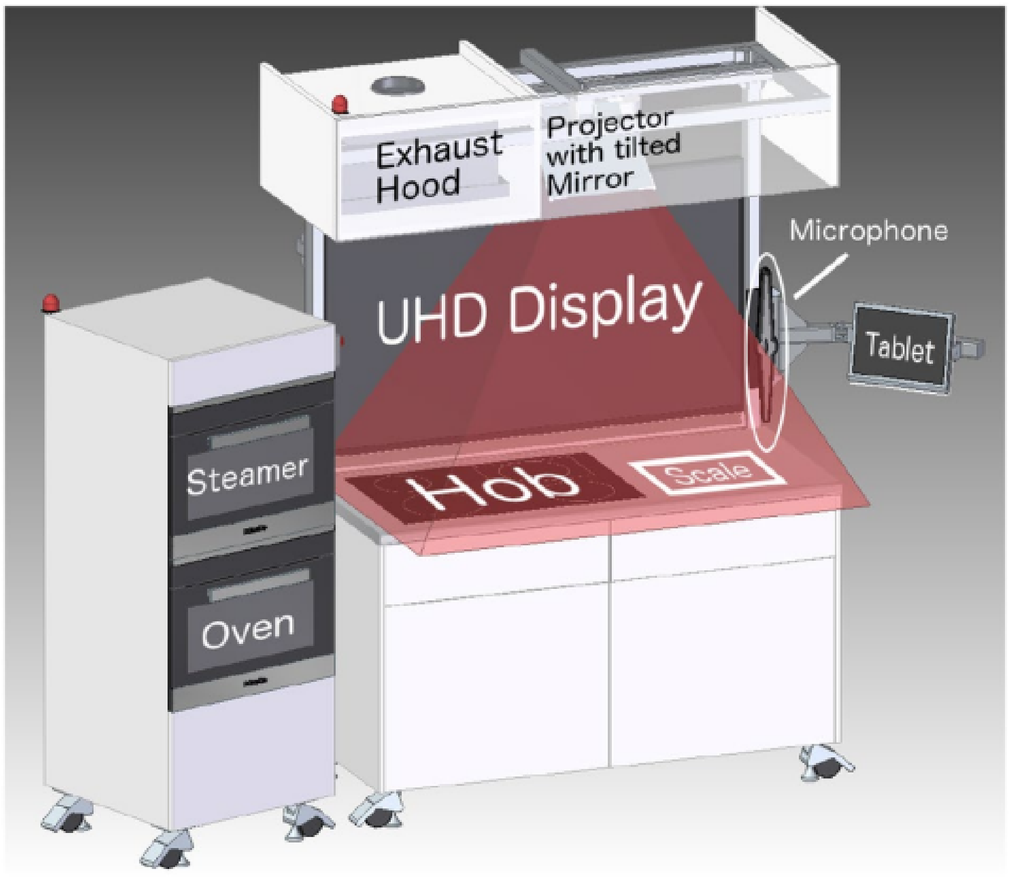
The system deploys object detection and tracking as one of the most central components because it defines a fundamental basis for the tracking of cooking actions carried out by the user. These modules provide a list of Object-Beliefs, each referring a detected and classified object in the scene. Based on an XYZ/RGB/temperature point-cloud (see 2.3b) identical objects can be distinguished.

The implementation also employs a three-layer unit consisting of a model-free segmentation unit (see 2.3c) and an spatio-temporal tracking layer.

(a) Prototype



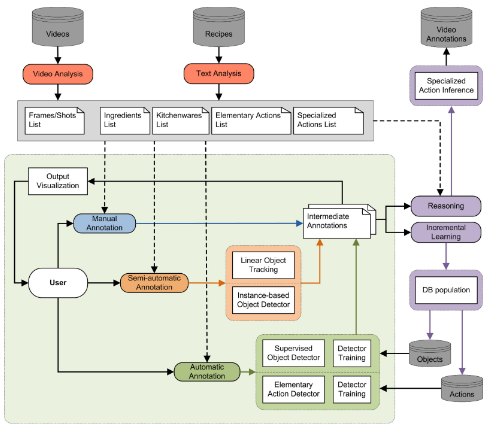
(b) XYZ/RGB/temperature point-cloud Kinect camera



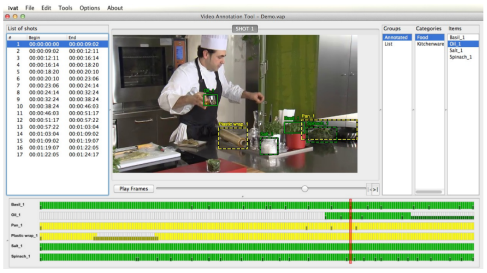
(c) Three-layer unit

Figure 2.3: Kognichef system

iVAT[1] 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 like food, kitchenware or action. iVAT overview is shown in 2.4a, while the GUI in 2.4b.



(a) iVAT overview



#	Begin	End
1	00:00:00:00	00:00:09:02
2	00:00:09:02	00:00:12:11
3	00:00:12:11	00:00:16:14
4	00:00:16:14	00:00:18:20
5	00:00:18:20	00:00:20:10
6	00:00:20:10	00:00:23:06
7	00:00:23:06	00:00:24:14
8	00:00:24:14	00:00:32:24
9	00:00:32:24	00:00:38:24
10	00:00:38:24	00:00:46:03
11	00:00:46:03	00:00:51:17
12	00:00:51:17	00:00:57:22
13	00:00:57:22	00:01:03:04
14	00:01:03:04	00:01:09:02
15	00:01:09:02	00:01:19:07
16	00:01:19:07	00:01:22:05
17	00:01:22:05	00:01:24:17

(b) iVAT GUI

In the main screen there are video related information like a list of various shots and items,while the most important feature is a video browser which 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 like click-able buttons, drag and drop operations, context menus and short-cuts. The lower part of the window (see 2.4b) 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 it shows if the item is present in that frame.

A Database for Fine Grained Activity Detection of Cooking Activities

The work of **Marcus Rohrbach, Sikandar Amin et.al.** [10] proposes a new activity dataset that contains 65 activities, which are recorded in a realistic setting. The activities are for the most part fine-grained and they display a low inter-class variability in order to be able to evaluate clssification and detection performance. Cooking activities are dealt with extreme caution and are dissected into small differences in activities as shown in Figure 2.5.

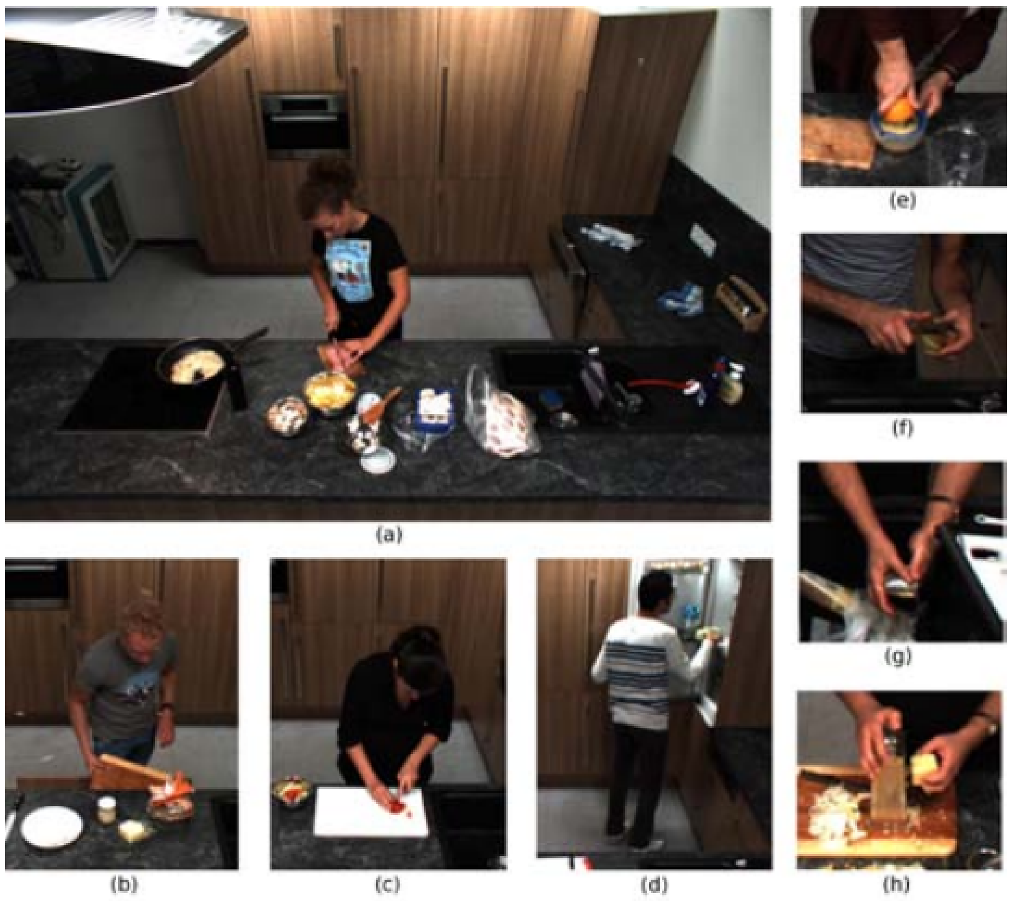


Figure 2.5: Figure representing cut (a,c) and peel (f), cut slices (a) and cut dice (c)

The contribution of this work relies on the novelty of the data-set which introduces a classification and detection challenge together with appropriate evaluation criteria.The form of the input data is 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 pre-computed pose.

Towards recognizing food preparation activities in situational support systems

This work [12] approaches food preparation activities with the use of machine learning algorithms combined with visual and accelerometer data (see 2.6). The goal of multi-modal activity recognition is achieved through an 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.

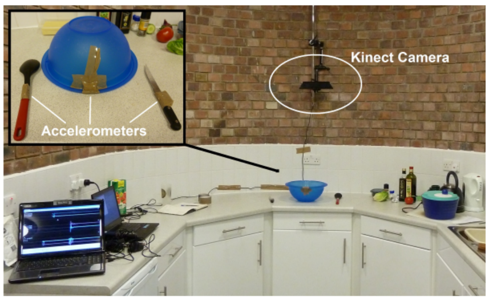


Figure 2.6: Experimental setup

Recognizing food preparation activities using bag of features

The work of Nguyen Thi Thanh Thuy and Nguyen Ngoc Diep [15] proposes a learning method for food preparation activity that is based on feature learning from histograms of motion primitives. The effectiveness is validated through recognizing ten activity classes. The segmentation of sensor data streams into frames was accomplished through a sliding windows, following the transformation of the sensor signals into a feature vector. The feature vector was classified classified using any classifier such as k-NN or Decision Tree (see 2.7) .

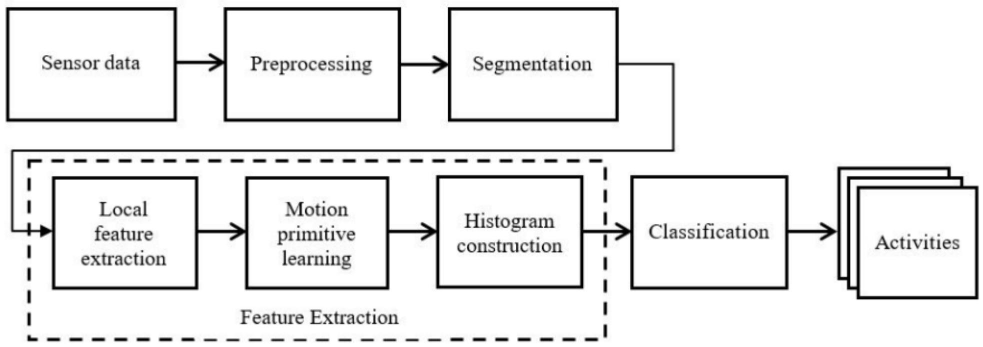
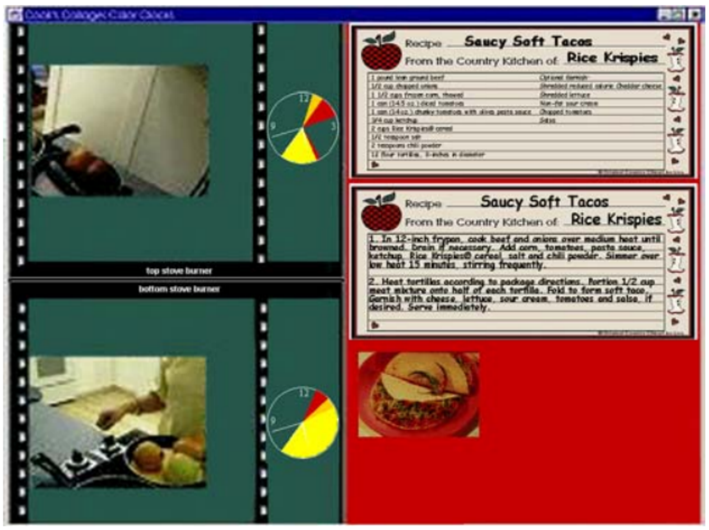


Figure 2.7: Schema for bag of features based food preparation activity recognition using accelerometers

Cook's Collage: Two Exploratory Designs

Cook's Collage [16] realizes a recipe tracking service through a video-based approach. If a cook interrupts his/her cooking, they will be provided with a memory aid in form of a video summary of what they have already done. More precisely, the user has access to images which show the last 6 actions and receives information about which ingredient 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 has proven helpful according to this work (Figure 2.8).



Recipe Saucy Soft Tacos
From the Country Kitchen of Rice Krispies

1 pound lean ground beef	Optional Garnish
1/2 cup chopped onions	Shredded reduced-fat Cheddar cheese
1 1/2 cups frozen corn, thawed	Shredded lettuce
1 can (14.5 oz.) diced tomatoes	Non-fat sour cream
1 can (14 oz.) chunky tomatoes with olive pasta sauce	Chopped tomatoes
3/4 cup ketchup	Salsa
2 cups Rice Krispies® cereal	
1/2 teaspoon salt	
2 teaspoons chili powder	
12 flour tortillas, 8-inches in diameter	

Recipe Saucy Soft Tacos
From the Country Kitchen of Rice Krispies

1. In 12-inch frypan, cook beef and onions over medium heat until browned. Drain if necessary. Add corn, tomatoes, pasta sauce, ketchup, Rice Krispies® cereal, salt and chili powder. Simmer over low heat 15 minutes, stirring frequently.

2. Heat tortillas according to package directions. Portion 1/2 cup meat mixture onto half of each tortilla. Fold to form soft taco. Garnish with cheese, lettuce, sour cream, tomatoes and salsa, if desired. Serve immediately.

Figure 2.8: Cook's collage

Smart Kitchen: A User Centric Cooking Support System

This particular work [4] proposes a "user 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. Important part in the aforementioned is that the user can change the cooking steps freely whenever he/she wants. Smart Kitchen supports three main protocols functionalities which are the following: tracking food, recognizing food material, and recognizing cooking actions (see 2.9).

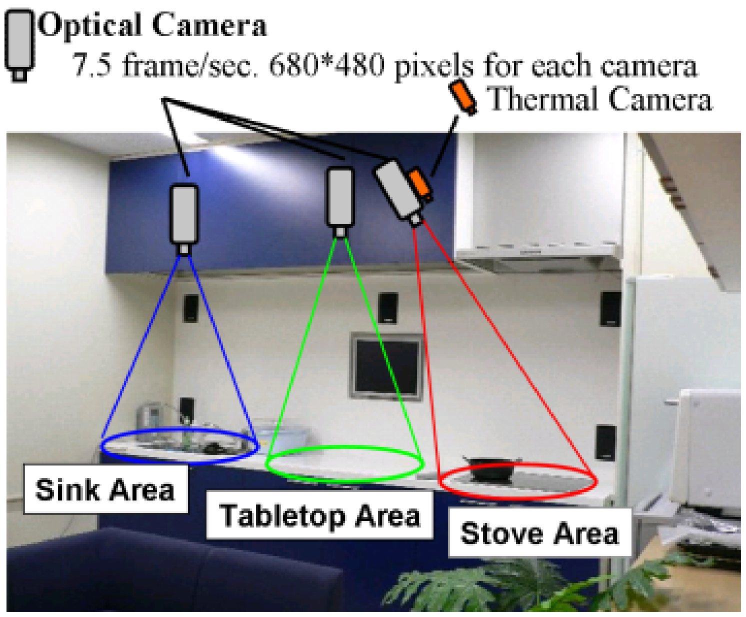
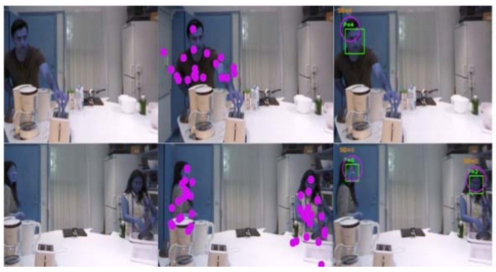


Figure 2.9: The Smart Kitchen environment

Human Sensing using Computer Vision for Personalized Smart Spaces

The work of Dipak Surie, Saeed Partonia, Helena Lindgren [14] or Kitchen As-A-Pal is described as a smart space with real-time human sensing capabilities.In particular, As-A-Pal is equipped with computer vision systems which are based on the fusion of fisher face recognition and skeletal tracking approaches. A Kinect 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.


(a) Skeletal tracking

(b) Skeletal tracking and face recognition

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 activity via the use of various hardware like sensors, cameras etc. [13, 17, 10]. At the same time there is an attempt to present more complete systems in a way that becomes apparent to the user that the future is right now and that ideas like assistance of the cooking process can be applied in the existing hardware [16]. 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 into real-case scenarios like a fully employable Smart Kitchen [7]. Next, there is going to be a typology of the discussed literature so as it is straight-forward even for the most benighted reader to follow the aforementioned literature review.

Most of the systems which utilize computer vision and machine learning methods, aim at the food preparation activity recognition or activity recognition in general. Systems like that [4], focus on solving intractable problems in computer vision like occlusion or food's change of form during cooking. In the same context, [14] move towards human sensing in terms of ubiquitous computing of human activity inside the smart house, ergo focusing on face recognition and skeletal tracking. A lot of research is also dedicated to the fusion of computer vision with data provided by sensors, a step to multi-modal activity recognition [13, 12], even though it is not successful in evaluating different fusion techniques as well as leaving a lot of unanswered regarding the credibility of such experimentations. Subsequently,a following research work also published by *Sebastian Stein and Stephen J. McKenna* [12], introduces data-sets of people performing food preparation activities as a means of evaluating different fusion techniques, additionally into the context of the recipe and the ingredients acted upon. On the same note, [1] focuses also 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 of evaluating different techniques like [4, 14, 13]. Nevertheless,it establishes an incremental learning framework that allows to increase the accuracy of the underlying algorithms over time which is compared to the works of *Thuy, Nguyen Thi Thanh et al.* [15] and *Urabe, Shuichi and Inoue et al.* [17]. The more "machine-learning based" approaches depend on neural networks to comprehensively respond to the challenge of cooking activities recognition. While both [17, 15] propose custom data-sets with food preparation activities, [15] only bases it's results in motion primitives and does not combine it with activity streams.On the other hand, [17] combines appearance-based approach using 2DCNN and motion-based approach using 3DCNN, which enables the improvement of the recognition performance of cooking activities on their data-set. Last but not least, *KogniChef* [7] is a commercial product which combines annotation modality like [12, 1, 15] reinforced with a XYZ/RGB/temperature point-cloud which serves as a sensor system for monitoring the food preparation activities, much like [13, 12, 1, 15, 17, 15]. 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.

There are many features presented in the above section some of which display great diversity to one another. Each system has its own weaknesses, as its own strengths, and deserves equal and impartial studying from the research world in order to gain and benefit both from the good points but also from the inadequacies.

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 like 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 data-sets**. The annotation of cooking videos requires a special care due to difficulties of gathering experimental data. It is an extremely laborious process which nonetheless provides with palpably significant records as to everything regarding a food preparation activity. It is to be noted that the process of annotating and documenting the

results in a data-set, is the utmost and most impartial way of contributing in the progress of the field.

3. **Activity recognition**. Activity recognition aims to recognize the actions and goals of the user in a food preparation activity like a cooking recipe.

4. **Sensor fusion**. There is a lot of research in the direction of using more than one sensors and combining the obtained data to advance the procedure of 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 has "infiltrated" horizontally various disciplines, and Computer Vision is no exception. Models like YOLO or Detectron developed by Facebook AI Research team, have improved tasks like object detection.

2.1 table demonstrates the comparative study of section 2.1 .As the reader will straightforwardly comment,there is not any system which specializes in every feature in total. A research work may serve well in the direction of the activity recognition analysis,while another may exemplify sensor fusion research. That is one of the main reasons that renders this research significant and and its results indispensable. This research work manages to consolidate a lot of research fields and define by far a new paradigm in food preparation activity and the recognition thus. It is a research work which consolidates "classic" computer vision techniques like object tracking with CSRT tracker Channel and Spatial Reliability of Discriminative Correlation Filter with powerful machine learning frameworks,based on Deep Learning and Faster R-CNNs. It is a research work which emphasizes in actuality by performing multiple undistortion and calibration actions, so as to reciprocate in the real-world scenarios. It is a research work which harnesses the modern powers of distributed and parallel computing in cases which is mostly essential like in the computation of optical flow which requires the full exploitation of the resources of a computer system. Present research work introduces new modalities which advance assistance in a cooking environment trough the medium of Linear Algebra and Calculus. Spatial Configuration modality solidifies assistance in a cooking recipe and opens up new horizons regarding cooking assistance systems for persons with disabilities.

Something							
Related work	ML	AR	SC	CV	RWS	CAD	SF
[13]	–	✓	–	–	–	–	✓
[14]	–	–	–	✓	–	–	–
[4]	–	✓	–	✓	–	–	–
[16]	–	–	–	–	–	–	–
[15]	–	✓	–	–	–	–	✓
[12]	–	✓	–	✓	–	–	✓
[10]	–	✓	–	✓	–	✓	–
[7]	–	–	–	✓	–	–	–
[17]	–	✓	–	✓	–	✓	–
Present thesis	✓	–	✓	✓	✓	✓	–

Table 2.1: Table of Comparative Study of Literature work

Chapter 3

Building Blocks

This chapter focuses on documenting and analyzing the building blocks of this work and the mathematical concepts comprising each. Accordingly, the hardware requirements are discussed first, whereas camera calibration and perspective transformation are analyzed after that. Object detection and the main concepts of it are discussed later, with object tracking following that. Finally, optical flow as an important aspect of the system regarding the monitoring of object's movement.

3.1 Getting Started: Hardware Requirements

The camera that was deployed for this research work was Kinect v2. Kinect v2 3.1 is a 3D sensor produced by Microsoft. It is composed of an RGB camera with resolution of 1920×1080 pixels, an infrared camera with resolution of 512×424 pixels and an infrared emitter.

Figure 3.1: Kinect V2 for Windows

After Kinect's demise as the main input device and controller for Xbox, it found its place in academia and other applications since around 2011. Specifically, the research world and hobbyists exploited the refined depth-sensing functionality as well as the low cost of Kinect compared to the high cost and general unreliability of other 3D camera options at the time. Especially Robotics and Computer Vision as fields were privileged over the use the depth-sensing of Kinect as it would enable robots to determine the shape and approximate distances to obstacles and maneuver around them.

3.2 Camera Calibration

Some pinhole cameras introduce significant distortion to images. Two major kinds of distortion are radial distortion and tangential distortion. Radial distortion causes straight lines to appear curved.Radial distortion becomes larger the farther points are from the center of the image.In the image that is shown below in which two edges of a chess board are marked with red lines it is visible that the border of the chess board is not a straight line and doesn't match with the red line. All the expected straight lines are bulged out [2].

Camera calibration is the process of estimating the coefficients of a camera which are necessary for obtaining a relationship between a 3D point in the real world and its corresponding 2D projection (pixel) in the image captured by that calibrated camera.

The parameters are mostly internal and external.Internal parameters pertain

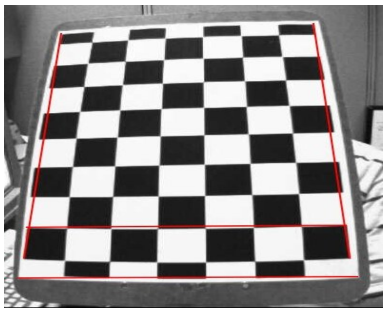


Figure 3.2: Chessboard edges marked with red line

to the lens system of the camera. The focal length, optical center, and radial distortion coefficients of the lens are some of them. Internal parameters are specified in what is called the intrinsic matrix **K**. The intrinsic matrix is an upper 3×3 triangular matrix.

$$K = \begin{bmatrix} f_x & \gamma & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

where,

f_x, f_y: x,y focal lengths

c_x, c_y: x and y coordinates of the optical center in the image plane

γ: skew between the axes,usually 0

External parameters refer to the orientation of the camera (i.e. rotation and translation) with respect to some world coordinate system. External parameters are specified in the extrinsic matrix. The extrinsic matrix is a 3×4 matrix, that is a combination of a 3×3 rotation matrix and a 3×1 translation vector.

3.3 Perspective Transformation

The nature of the human eye is to translate the closer objects as bigger and the farther as smaller. The term to describe this phenomenon in Computer Vision is called perspective. Meanwhile, transformation refers to the process of transferring an object from one state to another.

The combination of these two terms (i.e. perspective transformation) is used to describe the conversion of 3d world into 2d image. In other words, perspective transformation comprises a bilateral way of communicating the information the human eye receives, but in terms of camera and digital processing.

The mathematical notion that is used to reconcile these two worlds in this research is known Homography. The term is going to be analyzed in depth in the following examples.

3.3.1 What is Homography?

A Homography is a 3×3 transformation matrix that maps the points in one image to the corresponding points in the other image. The term refers to the transit of a specific plane of view to another using the aforementioned transformation matrix. Perspective transformation through homography matrix will transform the image in such a way that the resulting image will be "in front of us". The below example will illustrate the use of Homography.

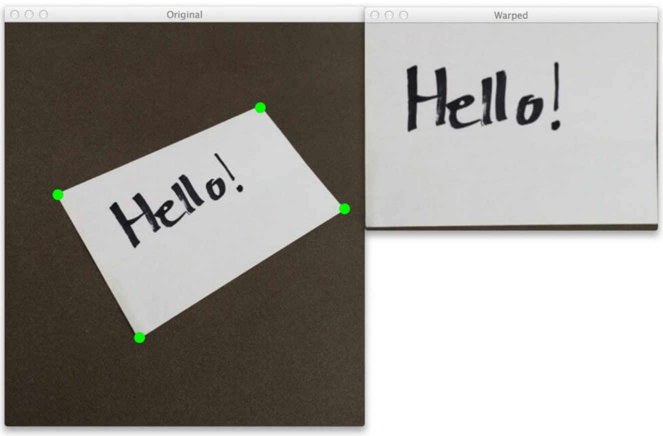


Figure 3.3: “top-down” view of an image

The relationship between these points or corresponding points as they are called in computer vision jargon is conveyed through Homography. Since a Homography is a matrix it can be written as:

$$H = \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix}$$

For each set of corresponding points $\{x_1, y_1\}$ in the first image and $\{x_2, y_2\}$ in the second image, the Homography H maps them in the following way:

$$\begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = H \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix}$$

3.4 Object Detection

In computer vision and machine learning, object detection refers to the two-step process of finding bounding boxes which 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 correctly the objects existing is crucial and it should be verified in every step of the procedure. The way of ”saying” if the object is detected well or if the object is detected poorly is by providing scores of detection.

3.4.1 Object Recognition vs. Object Detection

Object detection and object recognition are similar techniques for identifying objects, but they vary in their execution. Object detection is the process of finding instances of objects in images. In the case of deep learning, object detection is a subset of object recognition, where the object is not only identified but also located in an image. This allows for multiple objects to be identified and located within the same image.

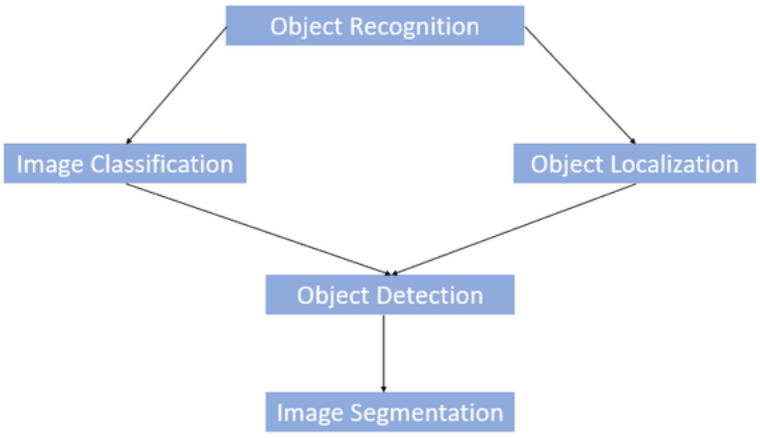


Figure 3.4: Object Detection and Object Recognition

3.4.1.1 Tools explained

Detectron2 The building block of Detectron2 is the Faster R-CNN, which befits the extensive family of Convolutional Neural Network (CNN) based image classifiers, that became popular after a CNN based method won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2012. In regards to Faster R-CNN, it is nothing but a single-stage model that is trained end-to-end. It uses a novel region proposal network (RPN) for generating region proposals, which saves time compared to traditional algorithms like Selective Search. It uses the ROI Pooling layer to extract a fixed-length feature vector from each region proposal.

For Faster/Mask R-CNN, the provided baseline is based on the FPN backbone combination which uses a ResNet+FPN backbone with standard conv and FC heads for mask and box prediction, respectively. It obtains the best speed/accuracy tradeoff.

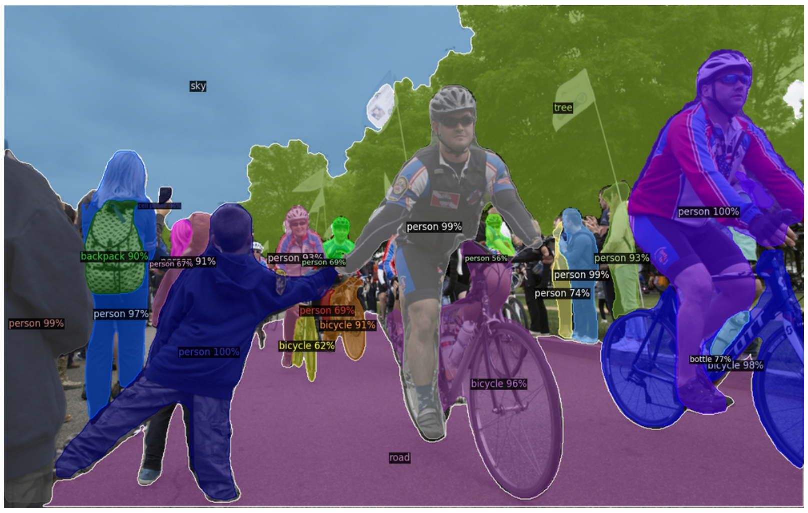


Figure 3.5: Image showing object detection task done with Detectron2

TorchServe TorchServe is an open-source model serving framework for PyTorch that makes it easy to deploy trained PyTorch models performantly at scale 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 necessar to write custom code to deploy the models. With powerful TorchServe features including multi-model serving, model versioning for A/B testing, metrics for monitoring, and RESTful endpoints for application integration, models can be taken from research to production quickly.

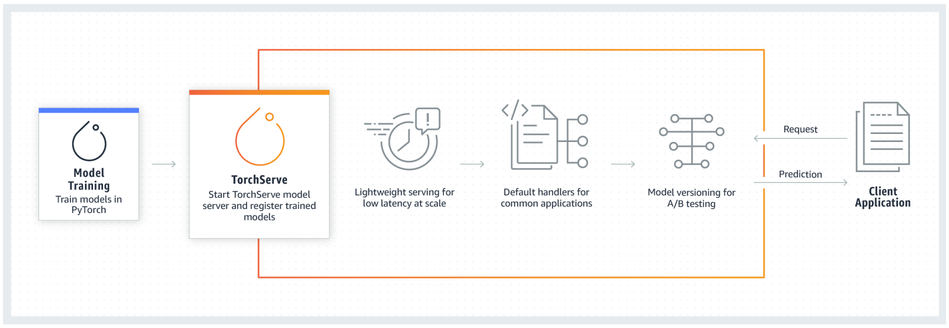

Model Training
Train models in PyTorch
TorchServe
Start TorchServe model server and register trained models
Lightweight serving for low latency at scale
Default handlers for common applications
Model versioning for A/B testing
Request
Prediction
Client Application

Figure 3.6: TorchServe features

CVAT In the real-world scenario, the object detection model has to be trained on a custom dataset. Building custom trained object detection model is not very straightforward with no respect to the framework used i.e. TensorFlow or PyTorch.

The tool used for labeling out images to train the model is CVAT [11]. CVAT is a free, online, interactive video and image annotation tool for computer vision. It is being used to annotate million of objects with different properties, whereas many UI and UX decisions are based on feedbacks from professional data annotation team.

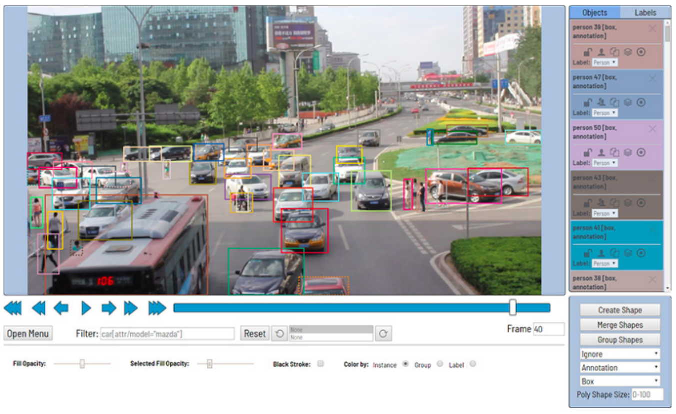


Figure 3.7: CVAT: an online computer vision annotation tool

3.5 Object Tracking

Locating an object in successive frames of a video is called tracking. However, tracking in computer vision is a very broad term that encloses conceptually similar but technically different ideas. For example, all the following different but related ideas are generally studied under Object Tracking:

1. **Dense Optical flow**: Algorithms that estimate the motion vector of every pixel in a video frame.

2. **Sparse optical flow**: Algorithms, like the Kanade-Lucas-Tomashi (KLT) feature tracker, that track the location of a few feature points in an image.

3. **Kalman Filtering**: Signal processing algorithm used to predict the location of a moving object based on prior motion information. One of the early applications of this algorithm was missile guidance.

4. **Meanshift and Camshift** : Algorithms for locating the maxima of a density function. They are also used for tracking.

5. **Single object trackers** : Trackers in which the first frame is marked using a rectangle to indicate the location of the object that needs to be tracked. The object is then tracked in subsequent frames using the tracking algorithm. In most real-life applications, these trackers are used in conjunction with an object detector.

6. **Multiple object track finding algorithms** : In the presence of an object detector, it makes sense to detect multiple objects in each frame and then run a track finding algorithm that identifies which rectangle in one frame corresponds to a rectangle in the next frame.

3.5.1 Why is Tracking essential

First, when there are multiple objects (kitchen objects in our case) detected in a video frame, tracking helps establish the identity of the objects across frames.

Second, in some cases, object detection may fail but it may still be possible to track the object because tracking takes into account the location and appearance of the object in the previous frame.

Third, some tracking algorithms are very fast because they do a local search instead of a global search. So a very high frame rate can be obtained for the system by performing object detection every n-th frame and tracking the object in intermediate frames.

3.6 Optical Flow

In this chapter, the algorithms used for calculating Optical Flow will be discussed. Namely, the method of Farneback and Lucas-Kanade with interpolation were implemented for Dense Optical Flow and the method of Lucas-Kanade for Sparse Optical Flow. Eventually, there will be measures and metrics presented as to which method performs better in an user-defined experiment.

3.6.1 What is Optical Flow?

Optical flow is the pattern of apparent motion of image objects between two consecutive frames caused by the movement of object or camera. It is a 2D vector field where each vector is a displacement vector showing the movement of points from first frame to second.

Optical Flow can be used in many areas where the object's motion information is crucial. Optical Flow is commonly found in video editors for compression, video stabilization, slow-motion, etc. Also, Optical Flow finds its application in Action Recognition tasks and real-time tracking systems.

3.6.2 Theoretical basics

Optical flow works on several assumptions:

- The pixel intensities of an object do not change between consecutive frames.
- Neighbouring pixels have similar motion.

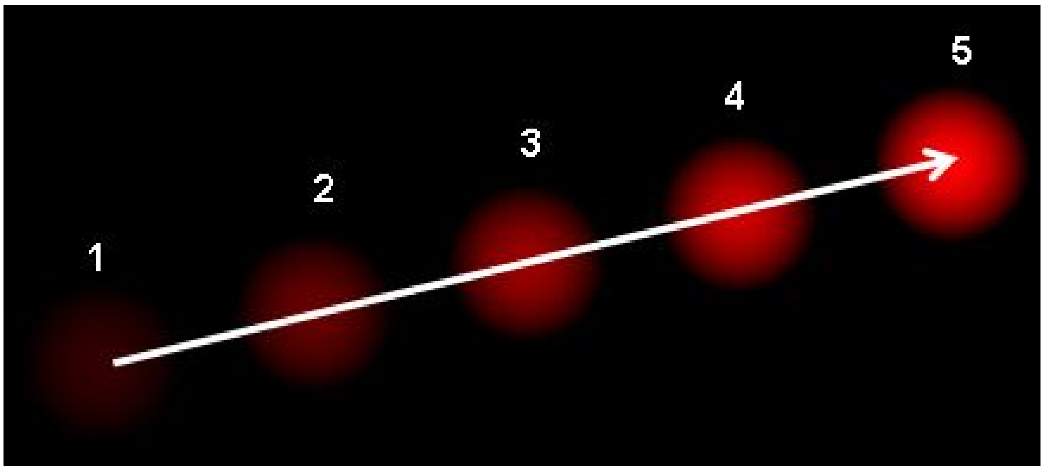


Figure 3.8: A ball moving in 5 consecutive frames. The arrow shows its displacement vector

Consider a pixel $I(x, y, t)$ in first frame. It moves by distance (dx, dy) in next frame taken after dt time. So since those pixels are the same and intensity does not change, we can say $I(x, y, t) = I(x+dx, y+dy, t+dt)$. Then by taking Taylor series approximation of right-hand side, removing common terms and dividing by dt we get the following equation:

$$\boldsymbol{f_x u + f_y v + f_t = 0}$$

where:

$$\boldsymbol{f_x = \frac{\partial f}{\partial x}; f_y = \frac{\partial f}{\partial y}}$$

$$\boldsymbol{u = \frac{du}{dt}; v = \frac{dy}{dt}}$$

Above equation is called Optical Flow equation. In it, we can find $\boldsymbol{fx}$ and $\boldsymbol{fy}$, they are image gradients. Similarly $\boldsymbol{ft}$ is the gradient along time. But $\boldsymbol{(u, v)}$ is unknown. We cannot solve this one equation with two unknown variables. So several methods are provided to solve this problem and one of them is Lucas-Kanade.

3.6.3 Sparse vs Dense Optical Flow

Sparse optical flow gives the flow vectors of some "interesting features" (say few pixels depicting the edges or corners of an object) within the frame while Dense optical flow, which gives the flow vectors of the entire frame (all pixels) - up to one flow vector per pixel. Dense optical flow has higher accuracy at the cost of being slow/computationally expensive.

3.7 Math Operations in 2D Image

The dot product or scalar product is an algebraic operation that takes two equal-length sequences of numbers (usually coordinate vectors), and returns a single number. In Euclidean geometry, the dot product of the Cartesian coordinates of two vectors is widely used. It is often called "the" inner product (or rarely projection product) of Euclidean space, even though it is not the only inner product that can be defined on Euclidean space or the 2D space of an image with the appropriate transformations. The determinant is a scalar value that is a function of the entries of a square matrix. It allows characterizing some properties of the matrix and the linear map represented by the matrix. In particular, the determinant is nonzero if and only if the matrix is invertible and the linear map represented by the matrix is an isomorphism. The angle between two vectors [x1, y1] and [x2, y2] is calculated through the following formula:

$$\boldsymbol{angle = atan2(det, dot)}$$

where:

$$\boldsymbol{dot = x1 * x2 + y1 * y2}$$

$$\boldsymbol{det = x1 * y2 - y1 * x2}$$

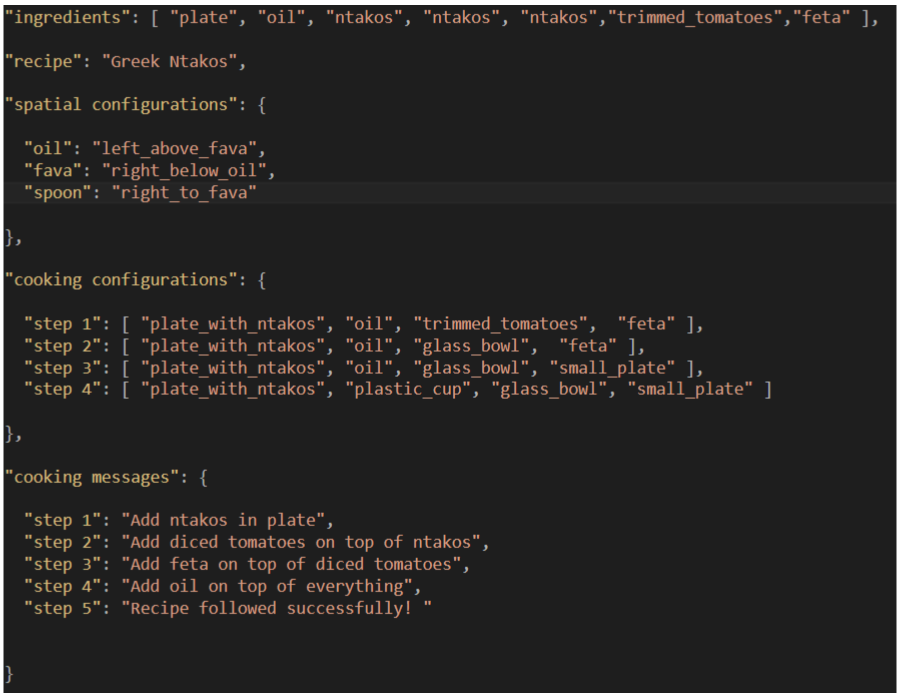
3.8 Configuration file explained

The configuration file 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.

As mentioned in chapter 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 a user to execute the recipe by providing him with the 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



```
"ingredients": [ "plate", "oil", "ntakos", "ntakos", "ntakos","trimmed_tomatoes","feta" ],

"recipe": "Greek Ntakos",

"spatial configurations": {

  "oil": "left_above_fava",
  "fava": "right_below_oil",
  "spoon": "right_to_fava"

},

"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! "


}
```

Figure 3.9: The configuration file used in this research work

Chapter 4

Implementation and Analysis

Chapter 3 includes the theoretical background of the methods which comprise the current research work. The current chapter extends the previous chapter and provides an insight of how these methods are implemented/adapted in order to serve the needs of the system.

4.1 Image Undistortion

In order to use Kinect with an operating system one should first determine what hardware and internal software is being used with the intention of finding how to connect and operate the Kinect with Microsoft Windows and OS X over USB. For this purpose, OpenKinect team created the libfreenect2 drivers [19].

For the purpose of calculating the camera intrinsic parameters and image undistortion, Python scripts were used [20].

Using video of a moving chessboard pattern or a sequence of images as an input, it finds following parameters:

- Focal length
- Principal point
- Radial distortion coefficients

4.1.1 Homography Calculation

In the current research, there are mainly two ways deployed for calculating Homography. Both of these ways share a lot in common and deploy already implemented methods which are provided in the OpenCV API.

The first one is predicated on the fact that at least four point correspondences between the two images are needed, even though having more that four corresponding points can affix the computation of Homography. OpenCV's function findHomograpy, which internally solves a linear system of equations to find the homography [2] will soundly estimate a homography that best fits all corresponding points. The points required for this operation are going to be clicked by hand through a GUI designed to collect hand-chosen points.

The second method of calculating Homography is hinged on the use of a chess board. The OpenCV function findChessboardCorners [2] is used for finding the corner points which will be placed in an order (from left-to-right, top-to-bottom). The kind of pattern that is being looked for, like 8x8 grid, 5x5 grid etc needs to be passed on. Normally a chess board has 8x8 squares and 7x7 internal corners. Then there is the option of rendering the corners with OpenCV function drawChessboardCorners [2]. The function draws individual chessboard corners detected either as red circles if the board was not found, or as colored corners connected with lines if the board was found. This procedure is going to bring out a set of four points, as it is clearly displayed in 4.1.

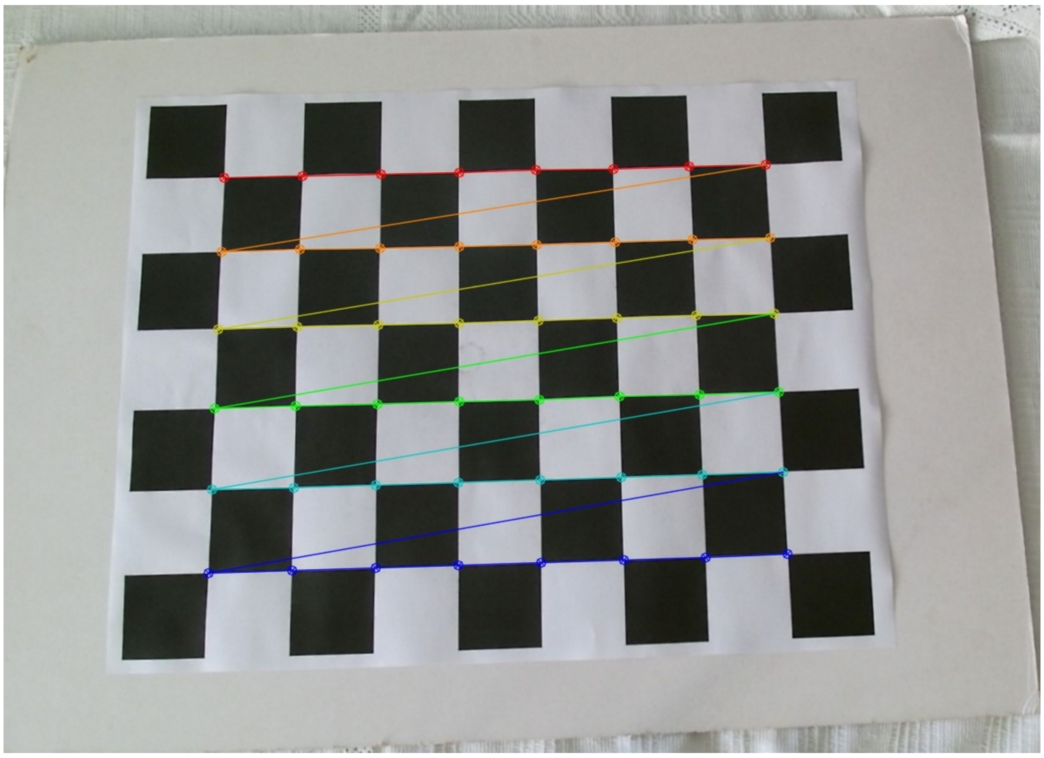


Figure 4.1: Found corners of a chessboard

4.2 Implementation of Object Detection

In the following section, the entire process of detecting the objects used in a cooking procedure will be explained in detail, starting from the use of a pre-trained model to classify images and ending up to the creation of custom data-sets that model a specific cooking procedure in order to be provided for the training a model that will be able to recognize and detect objects and tools used for this specific procedure (i.e. cooking).

The pre-trained model that was used for the task of detecting objects in our environment was Detectron2 which is Facebook AI Research's next generation library that provides state-of-the-art detection and segmentation algorithms. It is the successor of Detectron and maskrcnn-benchmark. It supports a number of computer vision research projects and production applications in Facebook.

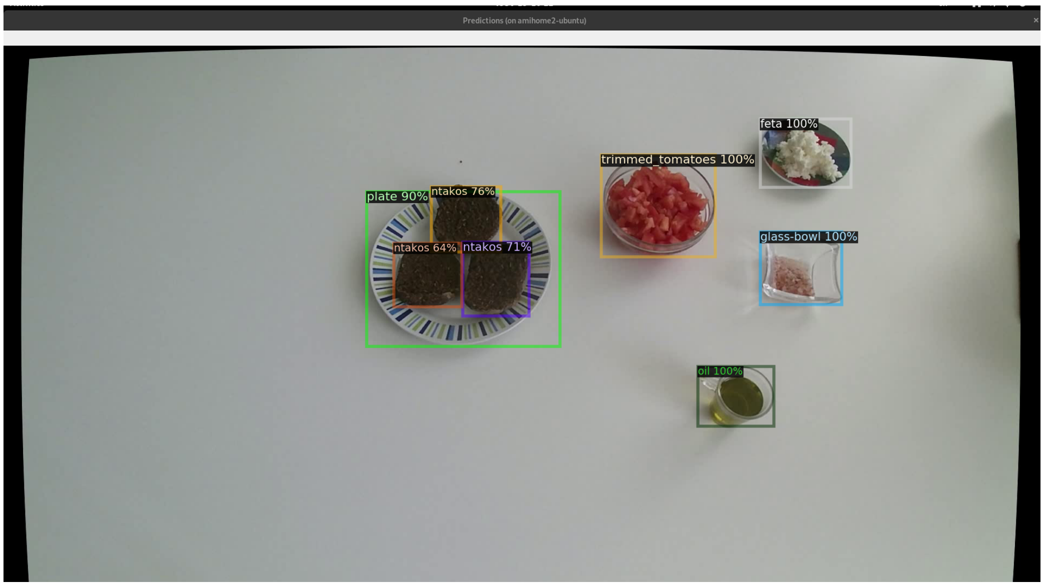


Figure 4.2: The output of the trained model: bounding boxes, scores and labels

4.2.1 Training with open datasets

The process of using the pre-trained models for predicting the classes (labels) of input is called Model Inference Process and it is going to be analyzed extensively in the following section.

Detectron2 uses COCO as base dataset for training regarding common objects, as kitchen objects. COCO, which stands for Common Objects in Context, is a large-scale object detection, segmentation, and captioning dataset. COCO has several features like: object segmentation, recognition in context, superpixel stuff segmentation. It consists of over 330K images (more than 200K labeled), 1.5 million object instances, 80 object categories, 91 stuff categories, 5 captions per image and 250,000 people with keypoints.

Let say that detector0 is the model, pre-trained on COCO classification tasks. The main steps of model inference process convert into the following:

1. Retrain detector0 with susbet of coco using classes of interest:
    - "name": "bottle",

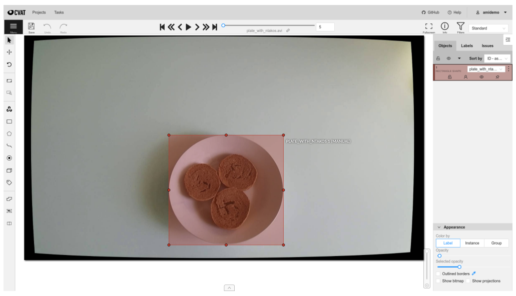
- "name": "wine glass",
- "name": "cup",
- "name": "fork",
- "name": "knife",
- "name": "spoon",
- "name": "bowl"

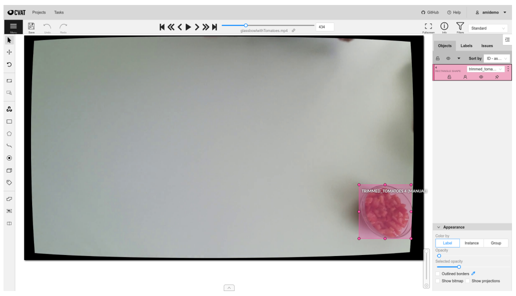
2. Obtain videos containing one object per video. Detector0 was run on each video with large confidence (0.7). Groundtruth data was extracted.
3. Retrain Detector0 with subset of coco using the above classes and 150 random images per class from step 1.
4. Serve Detector as a REST API using TorchServe [8].

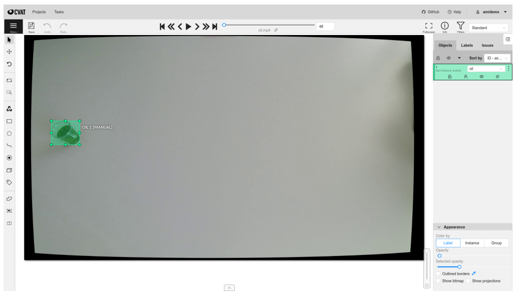
Step 3, despite it seemingly appears to be obsolete, is of key importance. The reason for that is that if the model is trained on only step 1 images, it has worse performance that if it is trained with a combination. An explanation is that the model overfitts to the custom objects. However, information about the general object class e.g. a knife from multiple knifes rather than a specific knife, is useful and has to be kept. The combined model achieves a balance between generalization and specialization.

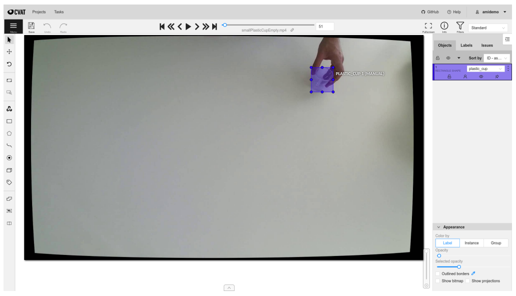
4.2.2 Training with custom datasets

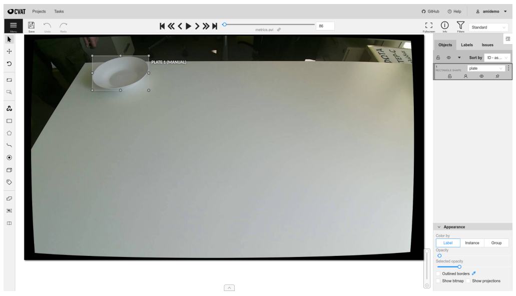
As mentioned in 3.4, when in the real-world, in most of the cases, the object detection model has need to be trained on a custom dataset. A custom dataset is a dataset that is created by the user, meaning that the user annotates the objects that the model will be trained to. The annotations as explained in 3.4 consist of the bounding boxes that include the concerning object and the label of the object. The annotations are created via the aid of CVAT.


(a)


(b)


(c)


(d)


(e)

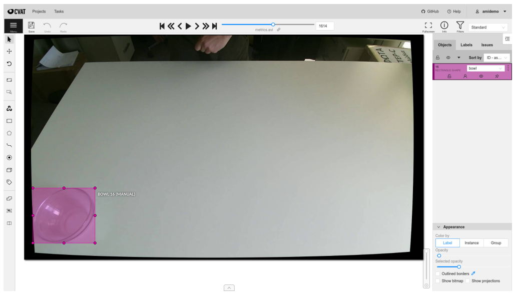

(f)

Figure 4.3: Custom annotations for the training of the model

4.3 Implementation of Object Tracking

Object tracking was implemented alongside object detection in order to enhance and improve it. In the context of this research work, tracking acts as supplementary method of identifying and annotating the objects in case of failure of object detection. The implementation of object tracking was facilitated through the use of the MultiTracker class [5], which exists in OpenCV's multi-object tracking API.The MultiTracker class is a naive implementation because it processes the tracked objects independently without any optimization across the tracked objects. The CSRT single object tracker is used in order to solve the problem of tracking. The CSRT tracker is not the fastest but it produces the best results in many cases examined.

Step 1: Attainment of first frame of a video

A multi-object tracker requires two inputs: a video frame and location (bounding boxes) of all objects that need to be tracked. Given this information, the tracker tracks the location of these specified objects in all subsequent frames. Video is loaded using the libfreenect2 which is the Callback interface of Driver for Kinect for Windows v2 (K4W2) devices to receive new frames. The MultiTracker is going to be initialized later from the first frame grabbed.

Step 2: Localization of objects in the first frame

The location of the objects to be tracked in the first frame is essential. The location is simply a bounding box. By deploying object detection, one of the first actions is identifying the position of the edges (i.e. top left corner and bottom right corner) of the objects in terms of pixels. OpenCV provides a function to form 2D rectangles of objects or forenamed bounding boxes.

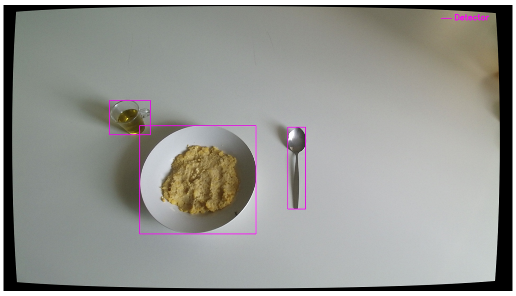
It is important to be clarified at this point that OpenCV provides other methods to acquire bounding boxes or Regions of Interests, as defined in the OpenCV jargon. However, it is seen fit that in this case to be counted solely on the detection data and thus acquire the needed bounding boxes in a more robust and computative way.

Step 3: Initialization of the MultiTracker

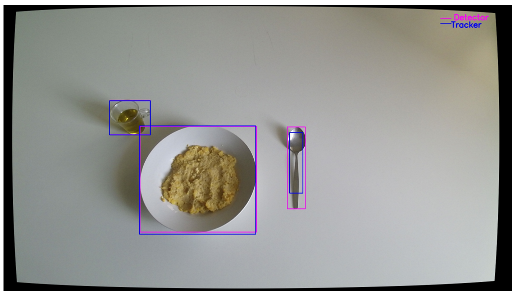
Until now, the first frame is read and the first bounding boxes are being formed by making good use of the detector. That is all the information that is being needed to initialize the multi-object tracker. First, a MultiTracker object is created and as many single object trackers to it as we have bounding boxes. The MultiTracker class is simply a wrapper for these single object trackers.The single object tracker is initialized using the first frame and the bounding box indicating the location of the object that needs to be tracked. The MultiTracker passes this information over to the single object trackers it is wrapping internally.

Step 4: **MultiTracker update and results' display**

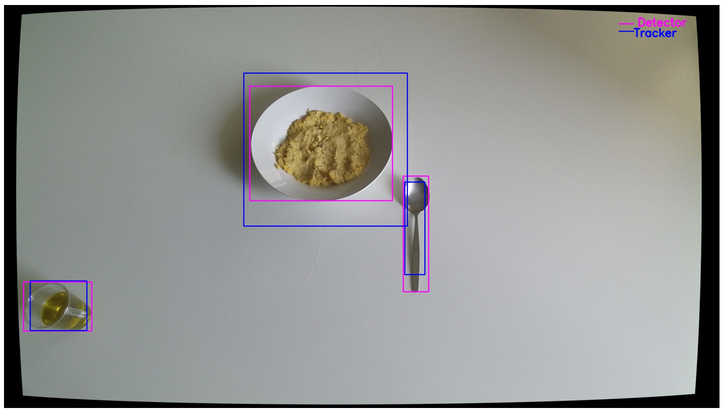
Finally, the MultiTracker is ready and multiple objects can be tracked in a new frame. It is very important to mention that the update procedure is implemented manually and not through the update methods provided by OpenCV. The procedure involves various deviations from any other method described in the OpenCV manual. For this reason, we make use of a special frame counter utility which helps us isolate the multi-tracker from possible mix-ups with the already implemented functionalities like Object Detection or Optical Flow.



(a) Detection



(b) Detection and tracking combined



(c) Updated tracking

Figure 4.4

4.4 Implementation of Optical Flow

In this section, the steps of the implementation will be presented in as much accuracy as well as simplicity possible. The methods analyzed are the following:

1. Sparse Optical Flow with Lucas-Kanade algorithm
2. Dense Optical Flow with interpolation of Sparse Optical Flow with Lucas-Kanade algorithm
3. Dense Optical Flow with Farneback algorithm

4.4.1 Sparse Optical Flow with Lucas-Kanade

OpenCV has the implementation of Pyramid Lucas and Kanade with Shi-Tomasi algorithm improvement to calculate sparse Optical Flow.

At first, the video is being read and the Shi-Tomasi algorithm's features from the first frame are extracted. Also, some preprocessing things for algorithms and visualization are required here. Preprocessing refers to :

- creating random colors for the purpose of vizualizing the dots, better described as the features that have to be tracked
- converting frames from RGB to gray-scale, as it is a prerequisite for Lucas-Kanade algorithm

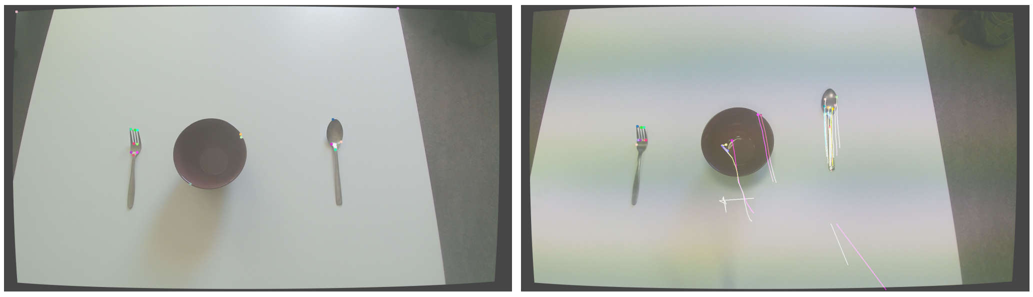

(a) Shi-Tomasi features to track (b) Optical flow with colored curves

Figure 4.5: Sparse Optical Flow with Lucas-Kanade

The procedure described above is a cycled process, where a new video frame is read and and the Shi-Tomasi features are calculated. Later-on, the Optical Flow is also calculated in a loop. In other words, the algorithm takes two consecutive frames and finds the corners on the first one. After that, the Optical Flow is computed with the Lucas-Kanade algorithm, using the information about the corner location and this is repeated for each pair of consecutive images.

To boost the results and optimize the position of pixels in the image, the locations of the corners are refined. The algorithm iterates to find the sub-pixel accurate location of corners or radial saddle points.

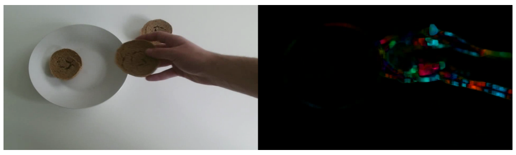
4.4.2 Dense Optical Flow with Lucas-Kanade

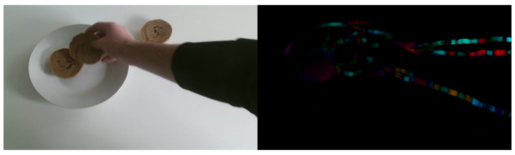
The calculation of Optical Flow is based on the same principles as the ones discussed in the previous section (see 4.4.1 for more details). The process of calculating Dense Optical Flow by providing Sparse Optical Flow attributs is achieved through interpolation of the flow matrix calculated in the method described in the previous section.

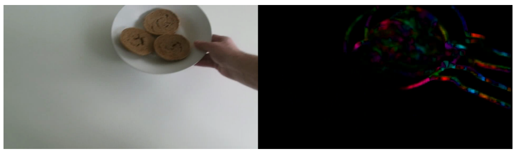
The following figure shows the vizualization (see 4.6) of the angle (direction) of flow by hue and the distance (magnitude) of flow by value of HSV color representation. The strength of HSV is always set to a maximum of 255 for optimal visibility.

4.4.3 Dense Optical Flow with Farneback

First, the method approximates the windows of image frames by quadratic polynomials through polynomial expansion transform. Second, by observing how the


(a) Moving ntakos


(b) Moving ntakos inside the plate


(c) Moving plate with ntakos

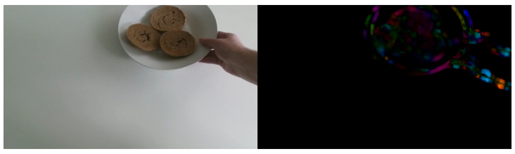
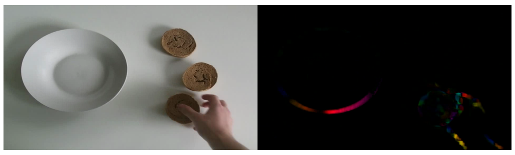
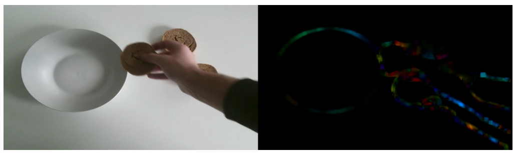
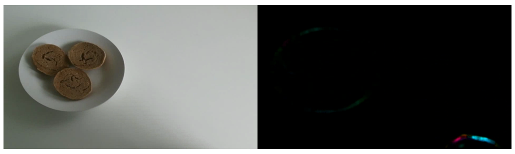

(d) Moving plate with ntakos

Figure 4.6: Visualization of Dense Optical Flow with Lucas-Kanade

polynomial transforms under translation (motion), a method to estimate displacement fields from polynomial expansion coefficients is defined. After a series of refinements, dense optical flow is computed. The visualization of the optical flow is shown below. It is identiacally (see 4.4.2) defined by hue and HSV color representation.


(a) Moving ntakos


(b) Moving ntakos inside the plate


(c) No move

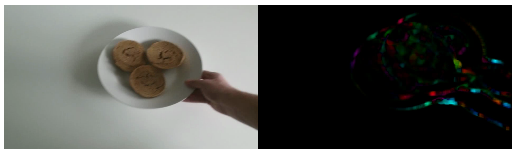
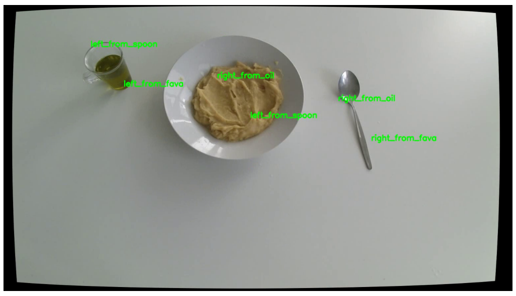

(d) Moving plate with ntakos

Figure 4.7: Visualization of Dense Optical Flow with Lucas-Kanade

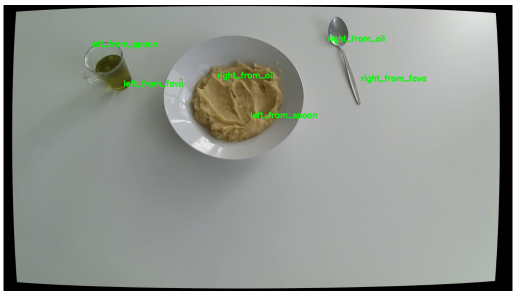
4.5 Calculation of angles between objects

The method to determine whether the object is right or left to another object is by calculating the angle between the respective vectors of these objects (see 3.7). 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.

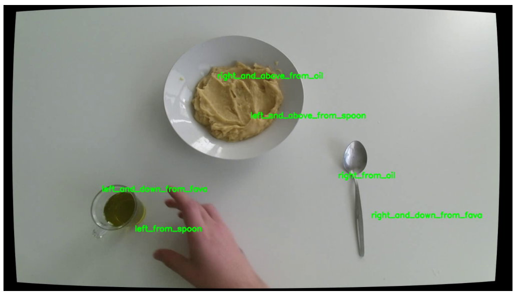
The angle does not represent the real angle of the objects in terms of 0-360 degrees because the principal value of the arc tangent of y/x is expressed in radians in C++. The native coordinate system of OpenCV had to be also transformed into counter-clockwise.



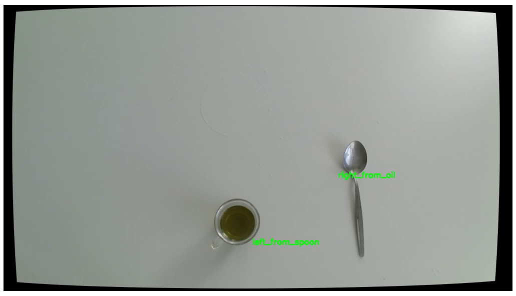
(a)



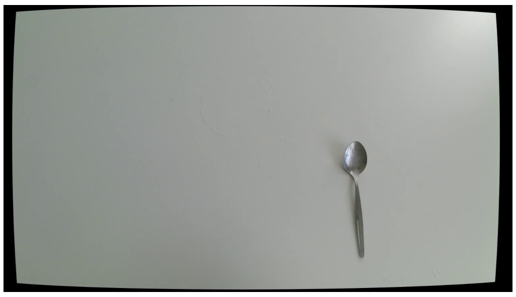
(b)



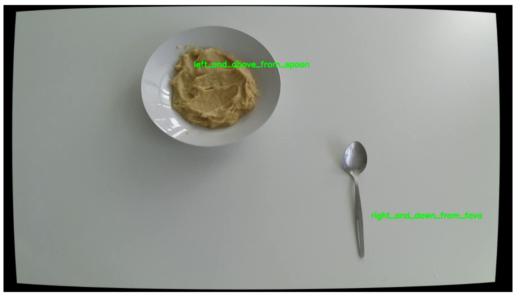
(c)



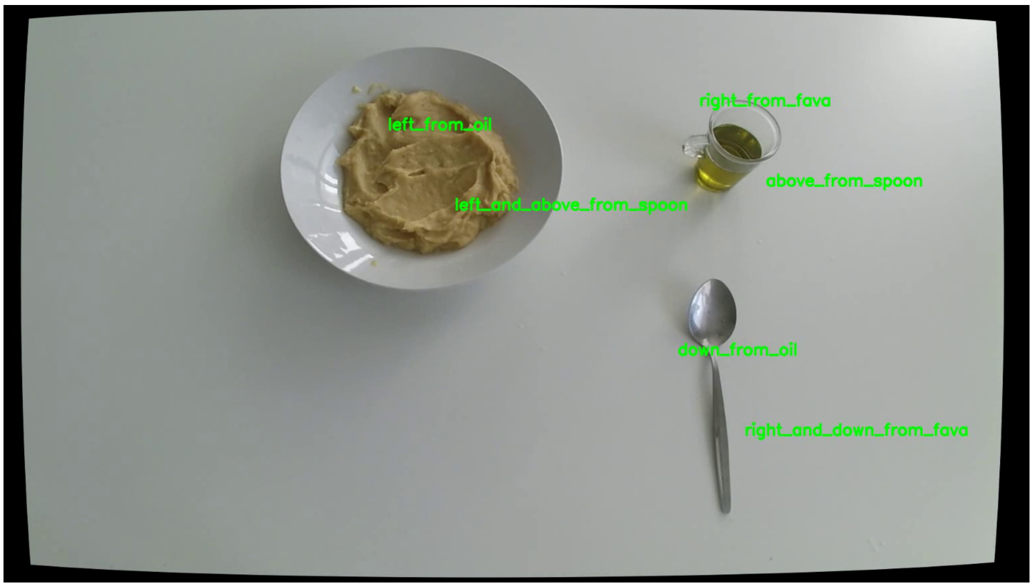
(d) Fourth step: add oil



(e)



(f)



(g)

Figure 4.8: Relative positions of objects

Chapter 5

Results and Evaluation

The following chapter shows the results produced in this research work in the form of sequences of images originally extracted from videos. The procedure is explained step by step and the results are tied in with the methods analyzed in chapter 4. The chapter is broken down into two parts which focus on the cooking assistance process and the spatial arrangement of the objects respectively. The part regarding the cooking assistance demonstrates the execution of two recipes, namely Greek Ntakos and Fava. The spatial arrangement modality demonstrates the capabilities of the system to locate an object in respect to another object. Both the assistance in the cooking process and the spatial arrangement modality are predicated upon the existence of a configuration file (see 3.8) which contains the wanted behaviors for each one of them.

5.1 Cooking Processes

The cooking assistance process will be broken down into two parts:

- ruling if the ingredients for a specific recipe are correct (see 5.1). This will be deduced by detecting what are the ingredients in the scene (see 3.4) and comparing them with the ingredients that should exist for the execution of this specific recipe. The sufficient ingredients for the execution of the recipe are listed also in the configuration file of the file (also described in 3.8).
- assisting the user into carrying out a recipe and validating if each step of the recipe was executed correctly. Each recipe is modelled as intermediate steps, much like a cookbook which 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 each step needs to suffice. The validation of the correct sequence of the steps will be explored in the next section.

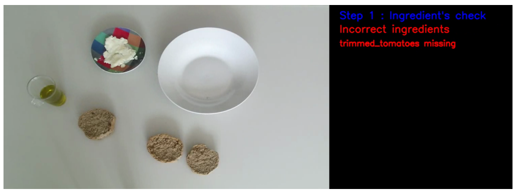
As explained in the previous section, the system is capable of validating the correctness of each step performed by the user by providing Computer Vision situated support. The solutions devised are predicated upon object detection and optical flow primarily, whilst several more techniques are used to boost them.

The correctness of the execution of a cooking recipe is validated through the following methods:

Sequence validation This method refers to the correct execution of the recipe which is established as the organizational unit of the cooking procedure. As mentioned before, a recipe is modelled as a sequence of steps that are coded in the configuration file (see 3.8). The step alone is nothing more than a list of correct ingredients/utensils that comprise the scene at a certain point. So, by employing object detection, the system is able to determine if the correct ingredients/utensils exist at that step and thus assert the precise realization of this specific step.

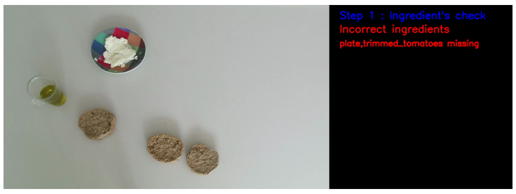
Motion Estimation Optical flow (see 3.6) is deployed to derive the motion information for a specific object. The information about the motion of an object functions as a safety net for the substantiation of the sequence of the steps. By obtaining definitive information that the ingredient that was meant to be added at a certain step of the recipe, has indeed moved, the system reduces some uncertainty about the existing and future states.

5.1.1 Checking for correct ingredients



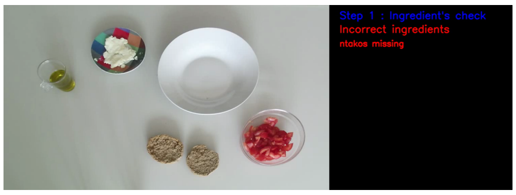
```
Step 1 : Ingredient's check
Incorrect ingredients
trimmed_tomatoes missing
```

(a) Tomatoes missing



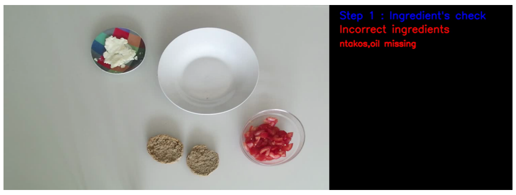
```
Step 1 : Ingredient's check
Incorrect ingredients
plate,trimmed_tomatoes missing
```

(b) Plate and tomatoes missing



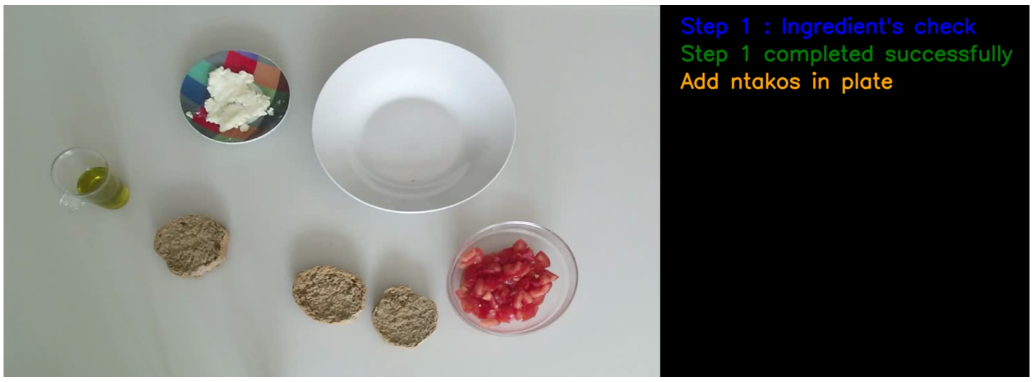
```
Step 1 : Ingredient's check
Incorrect ingredients
ntakos missing
```

(c) Ntakos missing



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

(d) Ntakos and oil missing

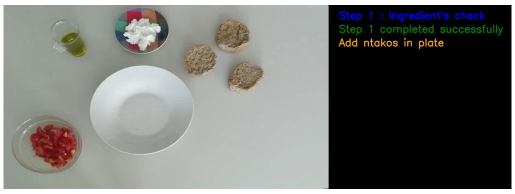


```
Step 1 : Ingredient's check
Step 1 completed successfully
Add ntakos in plate
```

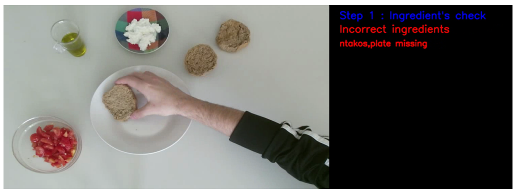
(e) All the ingredients are present

Figure 5.1: The process of checking for the correct ingredients

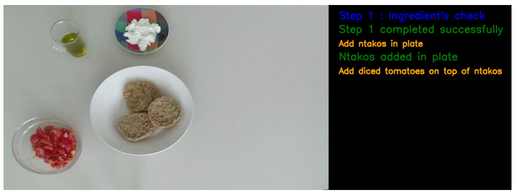
5.1.2 Recipes


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

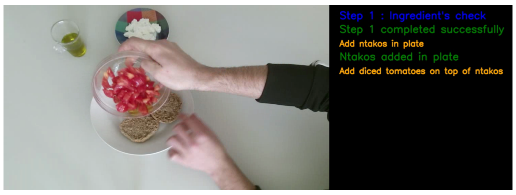
(a) First step: add ntakos


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

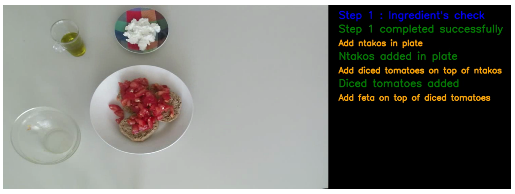
(b) Moving ntakos


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

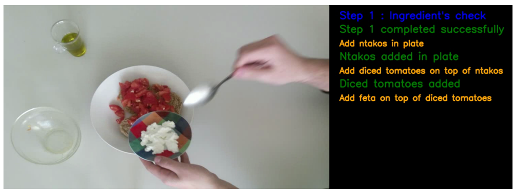
(c) Second step: add diced tomatoes


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

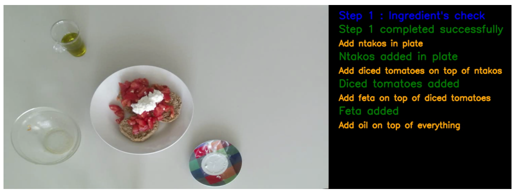
(d) Moving tomatoes


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

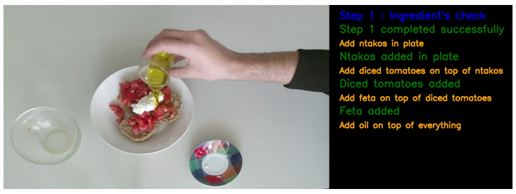
(e) Third step: add feta


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

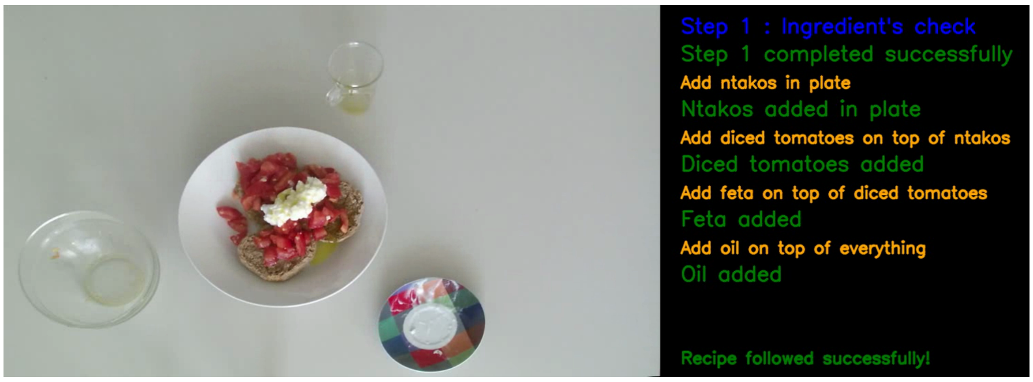
(f) Moving feta


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) Fourth step: add oil


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

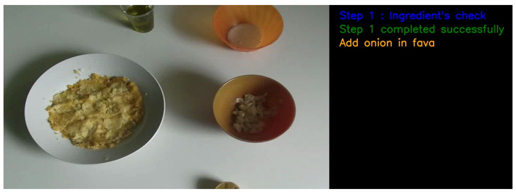
(h) Moving oil


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!

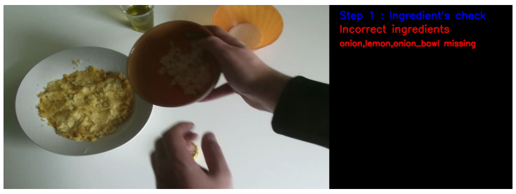
(i) Successful execution of the recipe

Figure 5.2: Cooking procedure: Greek Ntakos



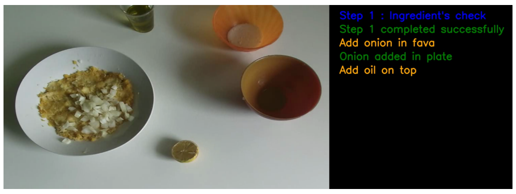
```
Step 1 : Ingredient's check
Step 1 completed successfully
Add onion in fava
```

(a) First step: add onion



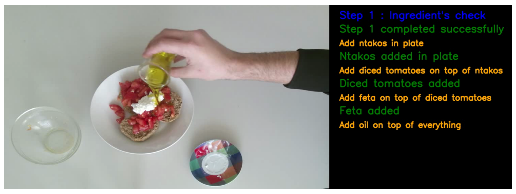
```
Step 1 : Ingredient's check
Incorrect ingredients
onion,lemon,onion_bowl missing
```

(b) Move onion



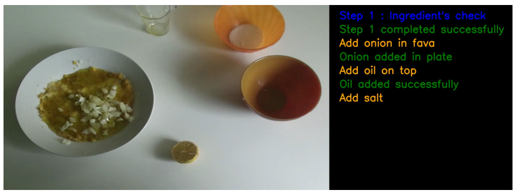
```
Step 1 : Ingredient's check
Step 1 completed successfully
Add onion in fava
Onion added in plate
Add oil on top
```

(c) Second step: add oil



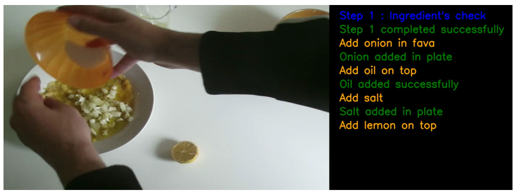
```
Step 1 : Ingredient's check
Step 1 completed successfully
Add ntakos in plate
Ntakos added in plate
Add diced tomatoes on top of ntakos
Diced tomatoes added
Add feta on top of diced tomatoes
Feta added
Add oil on top of everything
```

(d) Move oil



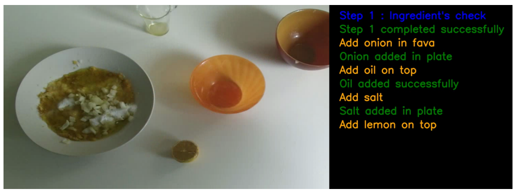
```
Step 1 : Ingredient's check
Step 1 completed successfully
Add onion in fava
Onion added in plate
Add oil on top
Oil added successfully
Add salt
```

(e) Third step: add salt



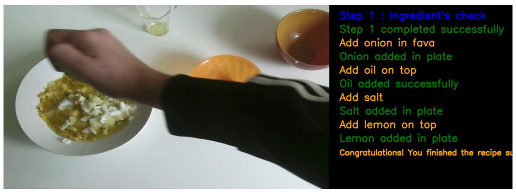
```
Step 1 : Ingredient's check
Step 1 completed successfully
Add onion in fava
Onion added in plate
Add oil on top
Oil added successfully
Add salt
Salt added in plate
Add lemon on top
```

(f) Move salt



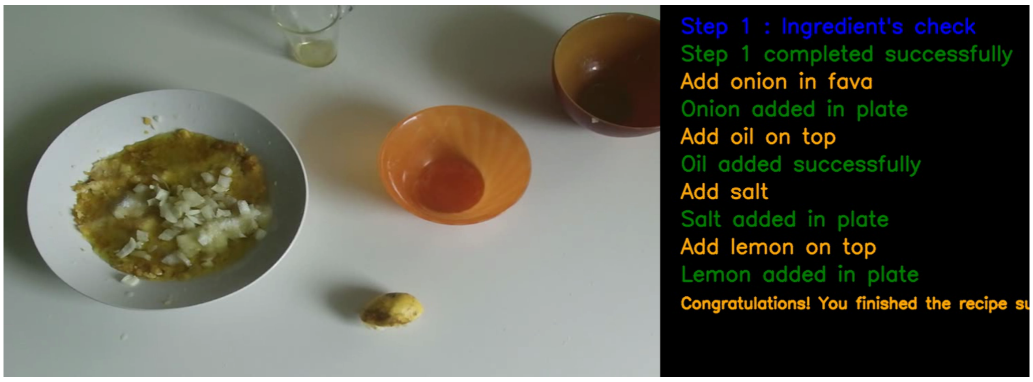
```
Step 1 : Ingredient's check
Step 1 completed successfully
Add onion in fava
Onion added in plate
Add oil on top
Oil added successfully
Add salt
Salt added in plate
Add lemon on top
```

(g) Fourth step: add lemon



```
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
```

(h) Move lemon



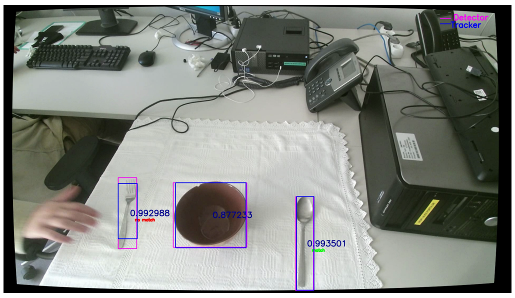
```
Step 1 : Ingredient's check
Step 1 completed successfully
Add onion in fava
Onion added in plate
Add oil on top
Oil added successfully
Add salt
Salt added in plate
Add lemon on top
Lemon added in plate
Congratulations! You finished the recipe su
```

(i) Successful execution of the recipe

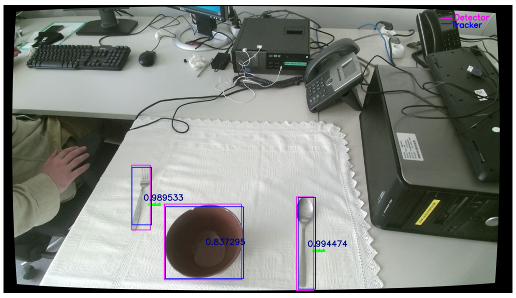
Figure 5.3: Cooking procedure: Fava

5.2 Spatial Arrangement of the objects

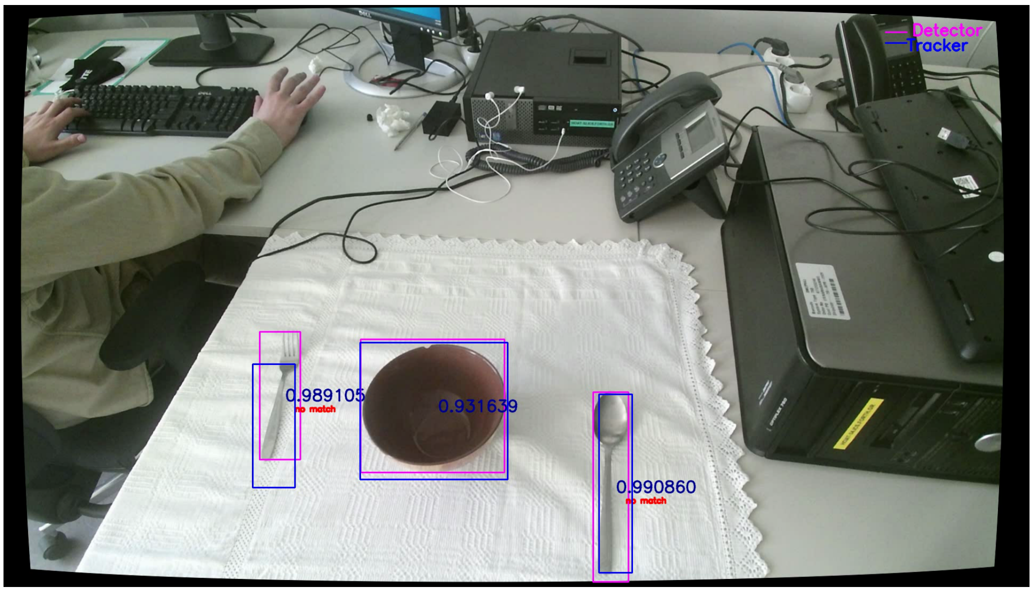
Clear from the plan mentioned in chapter 3 and explained explicitly in section 3.8, each and every object has a relative position to each other (see figure 3.9). The specified position of the fork is left above bowl and left above spoon. That is the reason why the message that pops in the first seconds is ”no match” meaning that fork’s position does not match with the specified position in the plan. However, if the fork is relocated a bit to the above, the message immediately updates. To this effect, it is also apparent that the message near the spoon is ”match” because its position matches with the specified position in the plan (i.e. right to bowl).



(a) ”Match”



(b) Move lemon



(c) ”No Match”

Figure 5.4: Spatial arrangement of the ingredients

5.3 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 (i.e. the number of items correctly labelled as belonging to the positive class) divided by the total number of elements labelled as belonging to the positive class (i.e. the sum of true positives and false positives, which are items incorrectly labelled as belonging to the class).

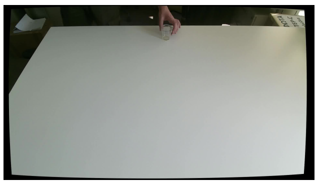
$$\boldsymbol{Precision = \frac{True\ Positive}{True\ Positive + False\ Positive}}$$

5.3.1 Optical Flow Metrics

To assess the performance of the algorithms mentioned in 3.6, 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 8 various positions with a steady force and within an equal distance.

(a) A plate

(b) glass bowl


(c) plastic cup

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 Machine Learning tool for predictive data analysis [9]. The purpose of measuring the precision of each optical flow method was double-edged; on the one hand it measured performance of each method, establishing which is the best of these methods. On the other hand, it also serves as conclusive way of establishing the best threshold for each method and later use this threshold in the program that quantified motion.

The thresholds tested were devised in a heuristic way by applying a $\mathbf{10^{\boldsymbol{x}}}$, where x equals to 0.5.

Table 5.1: Precision of each method grouped by threshold

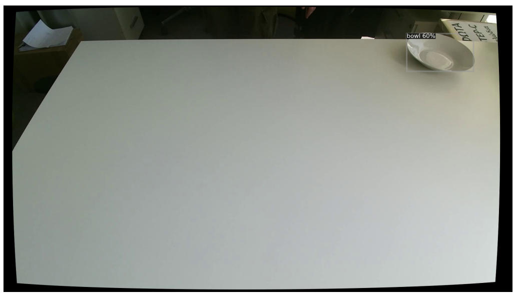
Methods	**Thresholds**				
	3.16	**10**	**31.6**	**100**	**316.22**
Lucas-Kanade Sparse Optical Flow	0.72	0.67	0.77	0.75	0.83
Lucas-Kanade Dense Optical Flow	0.0	0.72	0.8	0.81	0.82
Farneback Dense Optical Flow	0.8	0.86	1	1	0.0

5.3.2 Object Detection Metrics

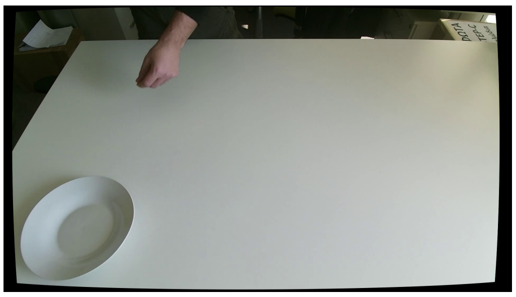
The experiment included detection of objects in unseen images during a real recipe preparation. The algorithm was able to detect almost all of the objects and at all video frames correctly. An exception was when the algorithm was presented with transparent objects (class missmatch) and objects that appeared inside other objects (false positives).

The pretrained model(off-the-shelf detector) was trained in LVIS [3], a dataset for large vocabulary image segmentation. LVIS uses the COCO 2017 train, validation, and test image sets. It included three objects that were common in the recipe of Greek Ntakos. These objects were the glass bowl, the plastic cup and the plate. Two of the objects (see 5.6) were transparent, which added extra complexity but at the same time manifested the necessity of training a custom object detector. Figure 5.6 cements some of the notions discussed here. In particular, the images (b) and (f) show that the detector did not achieve any detection of object for the plate and the plastic cup. In (a),(d) and (e), the detector mismatched the classes. In overall, the precision of the model was very poor. The average precision was 5.7 % for the cup was, and 0.0 % for the glass bowl and the plate.

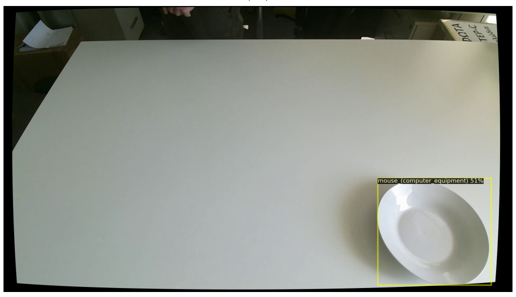
On the other hand, the performance of the custom object detector was really outstanding in all three of the objects and in most of the cases. Even though detecting a transparent object or even a white plate in white table, remains an intractable task, the custom detector achieved high scores. The detector achieved 98.812 % true positives in plate, 95.198 % in glass bowl and 83.647 % in plastic cup, succeeding in all three of the criteria which fulfill the ”true positive” classification: confidence score ¿ threshold; the predicted class matches the class of a ground truth; the predicted bounding box has an IoU greater than a threshold (e.g., 0.5) with the ground-truth. Figure 5.7 shows some of the successfully annotated instances of the custom detector.



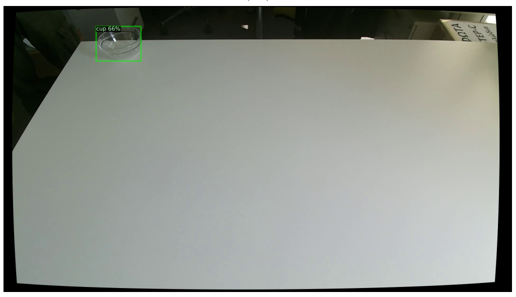
(a)



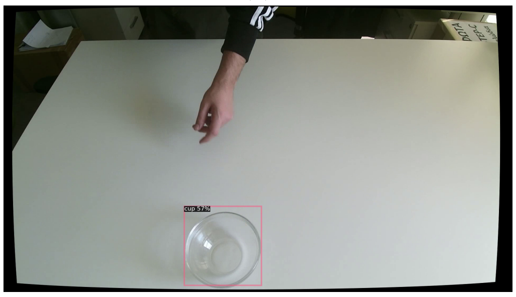
(b)



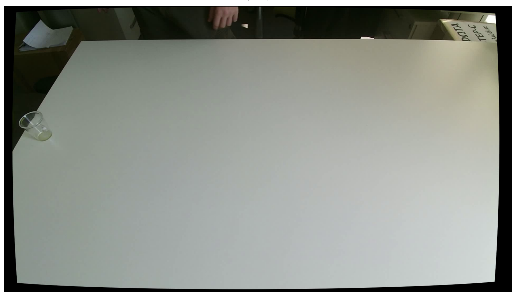
(c)



(d)

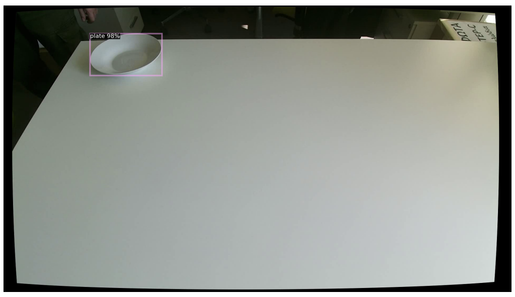


(e)

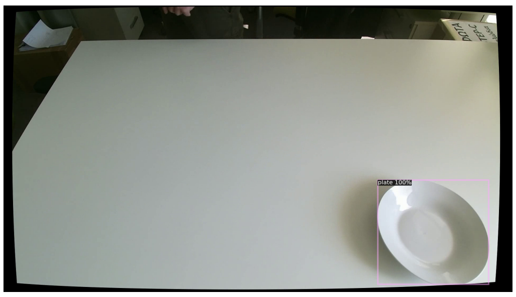


(f)

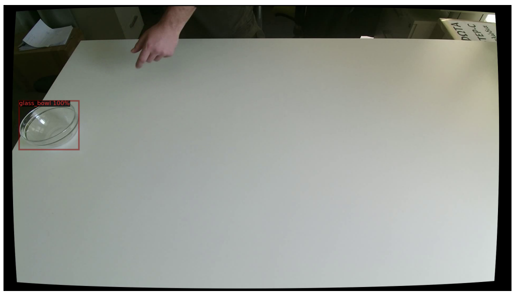
Figure 5.6: Performance of the off-the-self detector in three classes



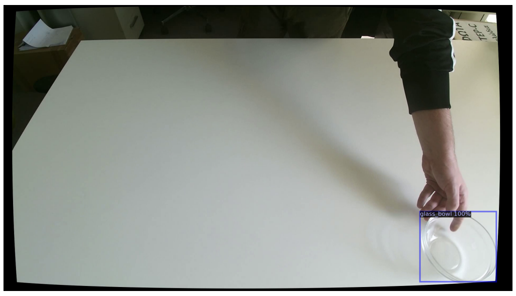
(a)



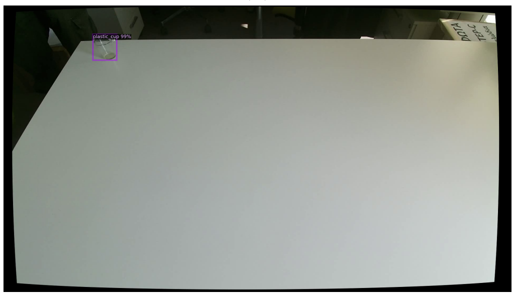
(b)



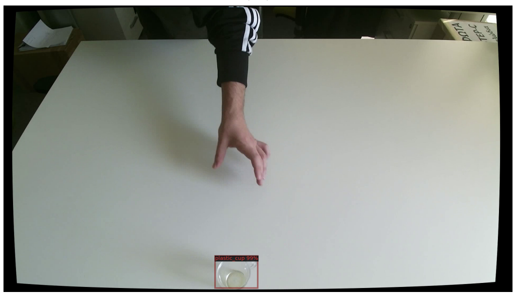
(c)



(d)



(e)



(f)

Figure 5.7: Performance of the custom detector in the same three classes

Chapter 6

Conclusion and Future Work

The current thesis 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 the appropriate messages which 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 by calculating each respective motion in each step.

The system also supports the spatial arrangement of objects like 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 relatively 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 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.

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 like water, corrupted pixels and complex background are important for better results, especially since this work was performed in a natural human setup. Extra sensors like accelerometers, weight sensors or wearable cameras could be deployed in order to enhance the hypothesis that an object moves or that an ingredient like oil empties. Activity recognition is a task that could further advance this work, by recognising specific cooking activities like cutting or peeling, or to interpret human gestures. The acceleration of some of the methods used could also signify a better processing and better results. Specifically, deploying processing in CUDA or applying multi-threading could lead in better results in Optical Flow. Last but not least, it is 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.

Bibliography

[1] Simone Bianco, Gianluigi Ciocca, Paolo Napoletano, Raimondo Schettini, Roberto Margherita, Gianluca Marini, and Giuseppe Pantaleo. Cooking action recognition with ivat: an interactive video annotation tool. In *International Conference on Image Analysis and Processing*, pages 631–641. Springer, 2013.

[2] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[3] Agrim Gupta, Piotr Dollar, and Ross Girshick. LVIS: A dataset for large vocabulary instance segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[4] Atsushi Hashimoto, Naoyuki Mori, Takuya Funatomi, Yoko Yamakata, Koh Kakusho, and Michihiko Minoh. Smart kitchen: A user centric cooking support system. In *Proceedings of IPMU*, volume 8, pages 848–854, 2008.

[5] Itseez. Open source computer vision library. `https://github.com/itseez/opencv`, 2015.

[6] Louise C Johnson. Browsing the modern kitchen—a feast of gender, place and culture (part 1). *Gender, Place & Culture*, 13(2):123–132, 2006.

[7] Alexander Neumann, Christof Elbrechter, Nadine Pfeiffer-Leßmann, Risto Kõiva, Birte Carlmeyer, Stefan Rüther, Michael Schade, André Ückermann, Sven Wachsmuth, and Helge J Ritter. "kognichef": A cognitive cooking assistant. *KI-Künstliche Intelligenz*, 31(3):273–281, 2017.

[8] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[9] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[10] Marcus Rohrbach, Sikandar Amin, Mykhaylo Andriluka, and Bernt Schiele. A database for fine grained activity detection of cooking activities. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1194–1201. IEEE, 2012.

[11] Boris Sekachev, Nikita Manovich, Maxim Zhiltsov, Andrey Zhavoronkov, Dmitry Kalinin, Ben Hoff, TOsmanov, Dmitry Kruchinin, Artyom Zankevich, DmitriySidnev, Maksim Markelov, Johannes222, Mathis Chenuet, a andre, telenachos, Aleksandr Melnikov, Jijoong Kim, Liron Ilouz, Nikita Glazov, Priya4607, Rush Tehrani, Seungwon Jeong, Vladimir Skubriev, Sebastian Yonekura, vugia truong, zliang7, lizhming, and Tritin Truong. opencv/cvat: v1.1.0, August 2020.

[12] Sebastian Stein and Stephen J McKenna. Towards recognizing food preparation activities in situational support systems. *Digital Futures 2012*, 2012.

[13] Sebastian Stein and Stephen J McKenna. 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*, pages 729–738, 2013.

[14] Dipak Surie, Saeed Partonia, and Helena Lindgren. Human sensing using computer vision for personalized smart spaces. In *2013 IEEE 10th international conference on ubiquitous intelligence and computing and 2013 IEEE 10th international conference on autonomic and trusted computing*, pages 487–494. IEEE, 2013.

[15] Nguyen Thi Thanh Thuy and Nguyen Ngoc Diep. Recognizing food preparation activities using bag of features. *Southeast Asian Journal of Sciences*, 4(2):73–83, 2016.

[16] Q Tran and E Mynatt. Cook's collage: Two exploratory designs. In *Position paper for the Technologies for Families workshop at CHI*, 2002.

[17] Shuichi Urabe, Katsufumi Inoue, and Michifumi Yoshioka. 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*, pages 1–8, 2018.

[18] Wikipedia contributors. Plagiarism — Wikipedia, the free encyclopedia, 2004. [Online; accessed 22-July-2004].

[19] Lingzhu Xiang, Florian Echtler, Christian Kerl, Thiemo Wiedemeyer, Lars , Hanyazou, Ryan Gordon, Francisco Facioni, Laborer2008, Rich Wareham, and et al. Libfreenect2: Release 0.2. *GitHub repository*, Apr 2016.

[20] Matěj Šmíd. Camera calibration using opencv. `https://github.com/smidm/video2calibration.git`, Sep 2, 2015.