UNIVERSITÀ DEGLI STUDI DI PARMA

Dottorato di Ricerca in Tecnologie dell'Informazione

Ventinovesimo Ciclo

Image Analysis-Based Food Recognition and Volume Estimation for Diet Monitoring

Coordinatore:

Prof.Marco Locatelli

Tutor:

Prof. Guido Matrella

Supervisor:

Prof. Stefano Cagnoni

PhD Student: *Hamid Hassannejad*

Decembre 2016

To my parents
and
To Pouya and Tara

Contents

List of Figures

List of Tables

Introduction

According to the World Health Organization, in 2014, the global prevalence of diabetes was estimated to be 9% among adults aged 18+ years, while at least 2.8 million people died as a result of being overweight or obese [1]. There are many other food-related problems and, among them, obesity is probably the most common one. Obesity decreases the quality and duration of life and increases individual, national, and global healthcare costs [2].

People's eating behavior has been shown to affect health issues. For example, a higher meal frequency has classically been associated with a lower incidence of obesity while eating outside the home, consuming fast food, and eating takeaway food tend to be associated with obesity [3].

These facts have fostered the emergence of many approaches to monitoring diet, whose target includes the food items of each meal, as well as the general eating habits of a person. However, most currently used self-reporting techniques demonstrate widespread underestimation of food intake, related to diverse psychological implications [4]. Under-reporting of dietary energy intake is common even in non-obese adults [5]. Other factors like person's awareness of food intake (particularly with snacks), literacy level, and memory and perception capabilities can also affect the reporting [6].

Thus, results of self-reported food intake should be interpreted with caution [5]. Because of this, many researchers have started developing affordable and non-invasive solutions that help users to measure and analyze their food intake. This idea has also inspired the Food Scanner prize, a 1 million Euro competition organized by the Eu-

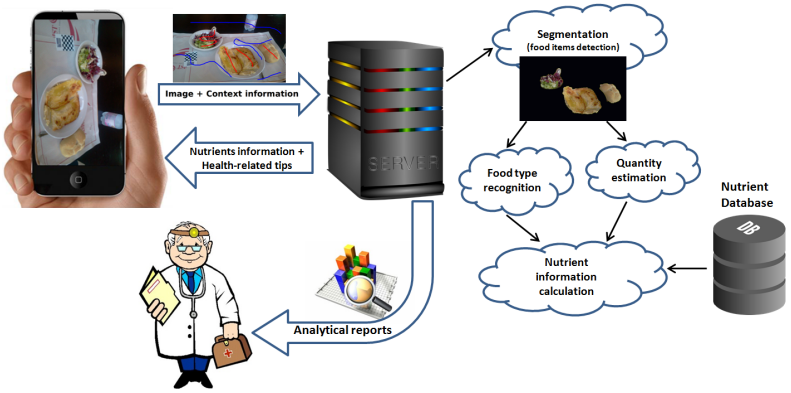


Figure 1: A typical system architecture for diet monitoring. The user takes a picture of the food using a smartphone and, after marking the food items, sends it to the server. The server analyzes the image and reports the nutrition properties to the user, while generating analytical reports for the health professional.

ropean Commission [7].

Research efforts have produced several methods for automatic diet monitoring, based on different principles and technologies.

In particular, two types of technologies have been utilized widely to tackle this issue: computer and wearable sensors, both relying on popularity of mobile devices.

Many apps have been introduced to record and analyze food intake. However, they are often based on procedures that are mostly manual and compel users to record meals and input food name, as well as personally estimate the food amount. In particular, this last task can be troublesome while being, at the same time, the most critical for meeting the apps' goals. Therefore, many efforts have attempted to make it faster, easier and more precise by automating it [8].

The applications aimed at analyzing food images to extract information about the food automatically typically ask the user to take one (or more) image(s) of his/her meal using a mobile device. These images, along with some possible annotations, are sent to a server that analyzes them and provides feedback to the user. The system may also provide the user with practical advice or generate analytical reports for the

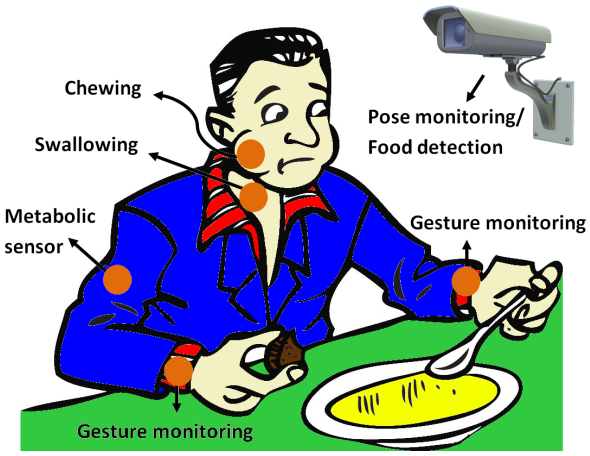


Figure 2: Multisensory monitoring for obesity research[2].

user's doctor (Figure 1).

One can describe image analysis as a chain of three main tasks, usually performed by the server: food recognition, image segmentation, and quantity estimation.

On the other hand, studies have shown that one's eating behavior has obvious effects on his/her energy intake, with all the corresponding health-related consequences. The same studies have also proven that modifying one's eating behavior can improve the health status of people with eating disorders. [9, 10, 11, 12, 13, 14].

The main purpose of developing wearable sensors is to monitor one's eating behavior and, later, to provide guidance on how to improve it. They could potentially capture timing, duration, and microstructure of food intake episodes, characterize rate of ingestion, ingested mass and nutritional and energy contents of food without imposing too heavy reporting burden to the user [2, 15, 16].

There are different choices for monitoring free-living events. Wireless Body Area Networks (WBANs) are widely used in medical and non-medical applications [17]. They usually monitor physiological parameters like blood pressure, body temperature, ECG, and blood flow using a set of sensors [18]. Moreover, the present generation of smartphones is already equipped with high-quality inertial sensors [2] which makes them a proper choice for monitoring physical activity. Moreover, smartphones

can also be used as a hub to gather information from different sensors. Other studies describe specific systems for detecting eating-related events. These sensors are usually designed to monitor events like:

- **Arm gestures**: Arm gestures have been one of the first activities to be monitored to detect eating behaviors in a non-invasive way. Amft et al. [19] tried to detect eating and drinking by proximity sensors placed on the arms. The idea is that, while eating, the hand approximates mouth, so two proximity sensors placed on the lower part and the upper part of the arm could detect the eating action. Y. Dong et al. [20, 21] also used a wrist-worn accelerometer to detect the hand's eating gestures. One of the main flaws of this approach in free-living cases is the similarity of eating-related hand gestures with other activities, like, for example, smoking.

- **Chewing**: Eating solid food can be detected by monitoring chewing events. Different sensors are available for this purpose. Amft introduced an acoustic ear-pad sensor device that captures air-conducted vibrations deriving from chewing [22]. In [23, 24, 25, 26] a sensor system is introduced containing an in-ear microphone to record chewing and swallowing sounds in the ear, while [27] use a piezoelectric film strain sensor to sense jaw motions during chewing. However, in the real world, environmental noises may introduce relevant errors in most of these systems.

- **Swallowing**: The basic idea is that eating episodes can be detected based on the swallowing pattern. Using Electromyography has been one of the earliest approaches to monitor swallowing events [28]. Amft and Troster enhanced the system by fusing Electromyography data with sounds coming from a microphone. [29]. Obviously, such an approach is too invasive. In [30], a miniature throat microphone placed over the laryngopharynx is used to detect the event. Later, [31] showed that using Electroglottograph signals instead of acoustic data yielded better results. Although this latter approach is not affected by environmental noise, it is sensitive to motion and difficult to use in free-living

scenarios.

There have also been some attempts to embed different types of sensors into a single system to improve performance [32, 33]; however, the main challenges faced by these systems are real-world issues. Most of these systems have been tested on a small number of subjects and even fewer on free-living cases to assess their actual performance. Some free-living facts, like unpredicted events or environmental changes, could affect the performance of the system dramatically.
Besides this, practicality is an open question that requires a careful evaluation of the user's comfort level and the burden that wearable sensors impose on him/her. To measure comfort level, different physical, chemical, and psychological factors may be taken into account [34]. Wearable sensors cannot be successful without optimizing this parameter.
Finally, battery supply duration can also be a major limitation.
In this thesis, a new approach to automatic diet monitoring based on image processing is introduced. The solution has aimed to detect food type and estimate its size. The problem is divided to 3 sub-problems: food recognition, image segmentation, and portion estimation. Each sub-problem is addressed in a separated part which we discuss in this thesis. First, in section 1, a new method for food recognition is suggested. Then, a semi-automatic solution is introduced in section 2. Next, in section 3, a method for 3D reconstruction of the food item and estimating its volume is presented. Moreover, a mobile app has been developed which is included in the solution. Last but not least, we discuss about the method and offer some suggestions for future developments.

Chapter 1

Food Recognition

Food recognition is probably the most popular topic in the literature about automatic diet monitoring. Typically, as in most object recognition tasks, some features are extracted from the image, represented according to a suitable model, and finally fed into a classifier to recognize food. One can categorize features into two main groups: global and local features. Global features, such as color histogram or circular shapes, can be of limited use in the presence of intra-class variability, occlusions, variable poses, and lighting. Instead, local features like pixel color or SIFT features, describe the local visual properties of the image in the neighbourhood of salient or invariant points.

As can be easily imagined, the choice of such features plays a key role in determining the final recognition accuracy. The same choice should be made for the classifier.

Many studies have tried to combine different features to achieve better results. [35] uses a combination of color and texture features. In [36], various image features such as intensity, color, texture, contour continuity, motion are integrated within one uniform framework, while [37] operates on different color spaces (RGB, HSV, LAB), and on texture properties.

Also, several works have used Bag of Features (BoF) [38] algorithms for food recog-

nition. In [39], one of the baseline algorithms applied to the Pittsburgh Fast-Food Image Database (PFID) [39] uses a bag of SIFT features. Other studies have employed different kind of BoF algorithms to recognize food [40, 41, 42]. SIFT is a popular feature with BoF approaches; nevertheless, standard SIFT features do not consider color information. To address this shortage, Matsuda et al. [43] proposed a new method that use the SIFT features of different color channels together, while Kawano et al. [44] suggest using Fisher Vectors instead of conventional BoF to reduce quantization error and improve recognition accuracy.
Also, color features are widely used for food recognition. However, most studies do not perform a color calibration to enhance the features' impact. Fang et al. [45] suggest a method for color calibration and show its effectiveness when applied to color features.
Once a suitable feature set has been found, an algorithm is needed to recognize food. Support Vector Machine (SVM) classifiers have been one of the favorite choices [36, 40, 41, 42, 46, 47, 48, 49, 50, 51]. The SVM algorithm maximizes the margin between the feature space regions which represent different classes according to a distance measure, so the component of the feature vector should be normalized to avoid overestimating the relevance of features having larger values. Additionally, the Multi-Kernel Learning algorithms [42, 51] can optimize the process of assigning weights to the different components of the input feature vector.

Regardless of selected method, the accuracy of classifier is directly influenced by the chosen features. Moreover, it can also be affected by implementation details such as data structures and quantization methods.
Therefore, choosing the right features and finding the best way to represent them have been major challenges. This problem could be solved if good features were learned automatically using a general-purpose procedure. This is the key advantage offered by deep learning methods [52].
The great success of AlexNet [53] in the ImageNet's Large Scale Visual Recognition Challenge 2012 (ILSVRC2012) attracted attention of many researchers onto the effectiveness of deep learning methods in image processing. Consequently, other stud-

ies have evaluated the effectiveness of CNNs for food recognition [54, 55]. Although the results have been promising, they still need to be improved to be considered as a practical solution. Moreover, the large number of parameters and the high computation cost could be a barrier for mobile solutions.

In ILSVRC 2014, a new CNN model called GoogLeNet [56] reached unprecedented level of performances in general object recognition, thanks to the introduction of Inception, a multi-branch convolution module which could be adjusted to meet different design objectives. Compared to AlexNet's 8 layers, one might call it a very deep network. However, GoogLeNet actually used 12 times fewer parameters than AlexNet's architecture, while being significantly more accurate [56].

Although the results have been promising, further investigation is needed about the choice of the optimal CNN network architecture and training parameters, especially in terms of implementation on mobile devices.

Moreover, personalizing the recognition algorithm has been shown to be an effective way to improve recognition. In [57, 58], food recognition is based also on the user's eating habits. For example, the fact that the user drinks milk quite frequently may be used to assign high priors to milk during recognition. Wang et al. [59] incorporate temporal information to learn a person's dietary pattern, based on a recursive Bayesian model, to improve food classification accuracy. The temporal context in this paper refers to a relationship between specific food and the days when a subject is most likely to eat it. An example might be that person A eats pasta every other day while person B never has pasta but eats bread every day. In other studies, that assume that more than one kind of food may appear in the image, the recognition method involves food co-occurrence patterns [43, 57].

In spite of importance of food recognition for diet monitoring system, this step does not need to be fully automated. The diet monitoring process requires that the user cooperates in any case since he/she must, at least, take a picture of the food according to some specifications. Therefore, one can involve the user more effectively in lighter tasks which do not impose any unacceptable burden on him/her.

Both [60] and [61] describe semi-automatic approaches in which, at the end of the recognition process, the application shows a list of possible food categories to the

user, who manually indicates the correct one. In [35], the user is asked to list food items orally. Considering the open problems that are still to be solved in automatic food recognition, particularly when the number of food items is unknown "a priori", user-aided recognition could be a viable solution. However, the deep learning shows that it is possible to benefit from an automatic approach with a high level of accuracy. In this section, a food recognition method, based on one of the models introduced by Google. It is evaluated by classifying images belonging three well-known datasets of food images.

1.1 Image Datasets

If one wants to evaluate the results of an algorithm, a proper reference dataset including representative examples of all the issues the method is designed to tackle must be available. If one wants to compare the results of different algorithms, such a benchmark must also meet the requirements of all methods.

In fact, if such a database is available, researchers can define a new method's requirements based on it. Unfortunately, as long as this is not the case, as happens with relatively young research fields, each research group will tend to use a dataset tailored to the method they are developing. This problem, on the one hand, imposes a further burden on scientists, who need to collect a new specific dataset based on the method's specifications. On the other hand, it affects evaluation by favorably biasing the algorithm's results. Obviously, this also holds for food image analysis. Table 1.1 presents a few studies on food recognition. As shown, the number of food categories, diversity on the categories and source of the test sets vary significantly. In addition, in several studies the test sets are not publicly available, so there is no way to wave a fair comparison.

There have been a few attempts to build a general reference dataset, but none of them have fully reached the goal. For example, the Pittsburgh Fast-Food Image Database (PFID) [39] has been one of the first food image databases to be made publicly available: it contains 4,545 still images and 606 stereo pairs of images of

101 different food types. It can certainly be a good reference because of the number of images it contains and because it allows one to develop and test stereo vision algorithms. Nevertheless, it includes only fast-food images taken in a way that does not match real-life scenarios. Therefore, building algorithm-customized databases is still a common practice when developing new methods.

In other fields, the datasets used in widely-attended competitions have most often become a standard because the competitors' number has made it possible for researchers to compare their algorithms with a large set of other methods. The same might also happen with food image analysis if the Food Scanner contest [7] is successful.

Gathering food images from the Internet has become popular thanks to the diffusion of social networks and to the users' habit of posting pictures of their meals [62, 63, 64]. In fact, the Internet hosts millions of food images that, if properly extracted and organized, could allow one to produce large databases with many instances for every food category. For example, UEC FOOD 265 [64] is a food image database comprising 265 popular food classes from different countries. Every class comprises more than 100 images, annotated with a bounding box that provides the ground truth.

Most food images from the Internet have been acquired in environments that differ significantly from those that the user of a food monitoring app would find in real life. Thus, [65] describes ETH Food-101, a data set that comprises more realistic images selected from the foodspotting.com website. The database takes into consideration 101 food classes, each of which is represented by 750 original images for training and 250 manually cleaned for testing.

To increase generality, [66] introduces a database that includes 3,000 images of 20 different food courses, acquired using different cameras, in different illumination conditions, and from different shooting angles.

The diversity of food types in different regions and countries is another issue that makes it difficult to build a database that can work well everywhere. [67] describes the UNICT-FD889 dataset. It comprises 3,583 images of 889 different food courses of different nationalities (e.g., Italian, English, Thai, Indian, Japanese, etc.) acquired in unconstrained settings (e.g., no specification on background, lighting conditions,

etc.). The database comprises, on average, four pictures of each dish.

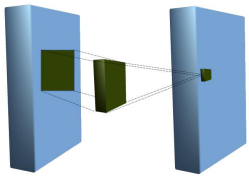
In summary, even if many food image databases have recently been made available, some basic requirements are still unaddressed, nor does any well-defined standard guideline for building them up exist yet. Databases including images taken from the Internet comprise a large number of images for each food type, but they are usually unable to provide different images of the same course or camera-calibration information. Additionally, in different databases, food classes are defined with different granularity.

However, the most relevant problem is perhaps the lack of reliable and complete ground-truth information. Food image databases usually report the name of the food but lack a size reference as well as information about the actual food amount, food location (segmentation data), etc.

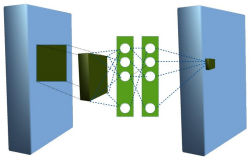
The algorithm we have developed has been run on three open datasets. Although they lack a ground-truth for size or other contextual data and so it is not possible to use them for evaluating other parts of this thesis, this high number of images and categories make them a proper tool to evaluate recognition methods.

Table 1.1: A few examples of different approaches to the food recognition problem. The test sets vary significantly among the studies. That makes comparing the approaches very difficult.

Title	Year	Feature	Classifier	Accuracy	Categories	Type of cat.	Test set
Wearable Context-Aware Food Recognition for Calorie Monitoring [68]	2008	Color, Size, Shape and Texture	Two-layer feed forward back propagation NN classifier	62.5% in average	4	Hamburger,Fries, Chicken nugget, Apple pie	Private - collected by cell phone
A Food Image Recognition System With Multiple Kernel Learning [51]	2009	Bag-of-SIFT	Multiple Kernel Learning (MKL)	61%	50	Japanese food	Private - Web images
Fast Food Recognition From Videos Of Eeting For Calorie Estimation [69]	2009	Different features including SIFT and geometric data	Customized feature matching method	73%	10 to 13	Fast food	Private - videos recorded by webcam
DietCam: Automatic dietary assessment with mobile camera phones [70]	2012	SIFT	Nearest neighbor	92%	50	Fast food	Private
Context Based Food Image Analysis [57]	2013	Color, Texture, Contextual dietary, Food co-occurrence	Customized classifier including SVM and nearest neighbors	44%	56	Commonly eaten food items in the United States	Private
Food Image Recognition with Deep Convolutional Features [71]	2014	HoG and Color patches	DCNN	72.26%	100	Japanese food	Public - UECFOOD100
A mobile, lightweight, poll-based food identification system	2014	Texture and Color	SVM	88%	9	Rice, Potatoes Meat, ...	Private
Food Recognition Using Consensus Vocabularies [50]	2015	SIFT and SPIN	SVM	79% and 30%	7 and 61	Fast Food	Public - PFID dataset
Food Recognition for Dietary Assessment Using Deep Convolutional Neural Networks [54]	2015	The whole image	6-layer deep convolutional neural network	84.90%	7	Pasta, Potatoes, Meat, ...	Private- 573 food items
Multiple Hypotheses Image Segmentation and Classification With Application to Dietary Assessment [72]	2015	Global and local features	KNN and SVM	70% 57%	83	Different food categories	Private
Food Image Recognition Using Deep Convolutional Network With Pre-Training And Fine-Tuning [55]	2015	The whole image	DCNN	78.77% and 67.57%	100 and 256	International and local food	Public - UECFOOD100 and UECFOOD256
Food Image Recognition Using Very Deep Convolutional Networks [73]	2016	The whole image	DCNN	88.28%, 81.45% and 76.17%	101, 100 and 256	International and local food	Public - Food-101, UECFOOD-100 & 256
Food Recognition and Detection with Minimum Supervision[74]	2016	Extracted with CNN	SVM	precision: 83.78%	10	Apple, Pizza, Coffee, ...	Private
DietCam: Multiview Food Recognition Using a Multikernel SVM [46]	2016	Part-based models and texture models	SVM	90%	55	popular American food categories	Private



(a) Linear convolution layer



(b) Multilayer perceptron layer

Figure 1.1: Comparison of a linear convolution layer and a multilayer perceptron layer. The linear convolution layer includes a linear filter while the multilayer perceptron layer includes a non-linear micro-network (adapted from [75]).

1.2 Deep Learning

Representation learning is a set of methods that allows a machine to be fed with raw data and to automatically discover the representations needed for detection or classification. Deep-learning methods are representation-learning methods with multiple levels of representation, obtained by composing simple non-linear modules each of which transform the representation at one level (starting with the raw input) into a representation at a higher, slightly more abstract level. By composing of enough such transformations, very complex functions can be learned. For classification tasks, higher layers of representation amplify aspects of the input that are important for discrimination and suppress irrelevant variations. An image, for example, comes in the form of an array of pixel values, and the learned features in the first layer of the representation typically correspond to the presence or absence of edges at particular orientations and locations in the image. The second layer typically detects motifs by spotting particular arrangements of edges, regardless of small variations in the edge positions. The third layer may assemble motifs into larger combinations that correspond to parts of familiar objects, and subsequent layers would detect objects as combinations of these parts. The key aspect of deep learning is that these layers of features are not designed by human engineers: they are learned from data using a general-purpose learning procedure [52].

Since the earliest days of pattern recognition, the aim of researchers has been to

replace hand-engineered features with trainable multilayer networks but, despite its simplicity, the solution was not widely understood until the mid 1980s. As it turns out, multilayer architectures can be trained by a simple stochastic gradient descent algorithm. As long as the modules are relatively smooth functions of their inputs and of their internal weights, one can compute gradients using the backpropagation procedure. The idea that this could be done, and that it worked, was discovered independently by several different groups during the 1970s and 1980s [52].

However, the design of the layers plays a main role to shape the networks capabilities. For example, solving the general problem of object recognition requires that the system which performs such a task (in nature, the human brain) be able to tackle the virtually infinite variability which characterizes its surrounding environment. Even large datasets with millions of images would represent just a fraction of the possible cases an individual is exposed to in her/his lifetime. Therefore, the recognition system cannot learn everything from scratch and needs some sort of prior knowledge to be directly "wired" into its structure. This is what seems to have happened to the human brain in millions of years of natural evolution. According to most theories about natural vision [76] our brain has a hierarchical structure, able to synthesize information at different abstraction levels. A newborn child is therefore immediately able to start learning about the infinitesimal fraction of the world represented by the surrounding environment because of the a priori knowledge/processing ability that its brain provides.

In summary, to tackle the problem of recognizing objects one needs a properly structured engine with a strong capability of further learning and adapting to the specific stimuli with which it is expected to deal properly. Convolutional neural networks (CNNs) [77] constitute one such class of models. The CNN layers are directly inspired by the classic notions of simple cells and complex cells in visual neuroscience [52] which is the prior knowledge injected in the network's structure. Their capacity of dealing with a large number of image categories and making strong and mostly correct assumptions about the nature of images [53] can be controlled by varying their depth and width.

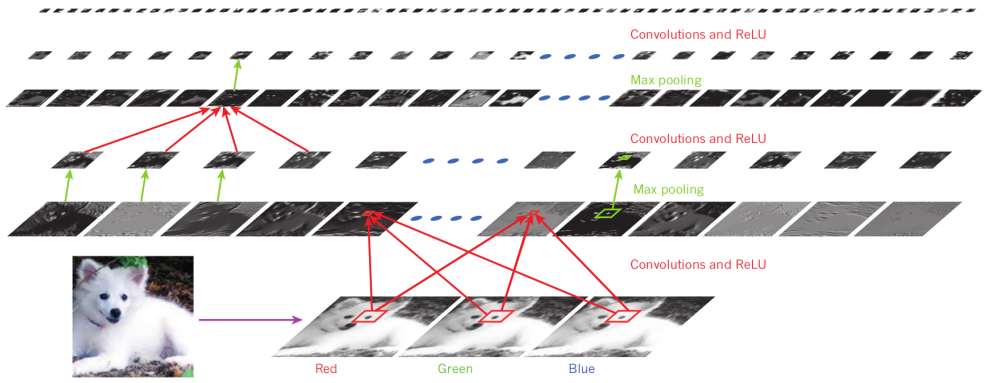


Figure 1.2: Inside a convolutional network. The outputs (not the filters) of each layer (horizontally) of a typical convolutional network architecture applied to the image of a Samoyed dog (bottom left; and RGB (red, green, blue) inputs, bottom right). Each rectangular image is a feature map corresponding to the output for one of the learned features, detected at each of the image positions. Information flows bottom-up, with lower-level features acting as oriented edge detectors, and a score is computed for each image class in output [52].

1.3 Convolutional Neural Networks (CNNs)

Convolutional neural networks(CNNs) are designed to process data that come in the form of multiple arrays, for example a colour image composed of three 2D arrays containing pixel intensities in the three colour channels. Many data modalities are in the form of multiple arrays: 1D for signals and sequences, including language; 2D for images or audio spectrograms; and 3D for video or volumetric images. There are four key ideas behind CNNs that take advantage of the properties of natural signals: local connections, shared weights, pooling and the use of many layers. [52]
The architecture of a typical ConvNet (Figure 1.2) is structured as a series of stages. The first few stages are composed of two types of layers: convolutional layers and pooling layers. Units in a convolutional layer are organized in feature maps, within which each unit is connected to local patches in the feature maps of the previous layer through a set of weights called a filter bank. The result of this local weighted sum is then passed through a non-linearity such as a Rectified Linear Unit (ReLU) [52]. All units in a feature map share the same filter bank. Different feature maps in a layer use different filter banks. The reason for this architecture is twofold. First, in array data such as images, local groups of values are often highly correlated, forming distinctive local motifs that are easily detected. Second, the local statistics of images and other signals are invariant to location. In other words, if a motif can appear in one part of the image, it could appear anywhere, hence the idea of units at different locations sharing the same weights and detecting the same pattern in different parts of the array. Mathematically, the filtering operation performed by a feature map is a discrete convolution, hence the name. [52]
Another main building block of CNNs is pooling layer. Although the role of the convolutional layer is to detect local conjunctions of features from the previous layer, the role of the pooling layer is to merge semantically similar features into one. Because the relative positions of the features forming a motif can vary somewhat, reliably detecting the motif can be done by coarse-graining the position of each feature. A typical pooling unit computes the maximum of a local patch of units in one feature map (or in a few feature maps). Neighbouring pooling units take input from patches

that are shifted by more than one row or column, thereby reducing the dimension of the representation and creating an invariance to small shifts and distortions. Two or three stages of convolution, non-linearity and pooling are stacked, followed by more convolutional and fully-connected layers. Backpropagating gradients through a CNN is as simple as through a regular deep network, allowing all the weights in all the filter banks to be trained. [52]

Deep neural networks exploit the property that many natural signals are compositional hierarchies, in which higher-level features are obtained by composing lower-level ones. In images, local combinations of edges form motifs, motifs assemble into parts, and parts form objects. Similar hierarchies exist in speech and text from sounds to phones, phonemes, syllables, words and sentences. The pooling allows representations to vary very little when elements in the previous layer vary in position and appearance. [52]

The convolutional and pooling layers in CNNs are directly inspired by the classic notions of simple cells and complex cells in visual neuroscience , and the overall architecture is reminiscent of the hierarchy in the visual cortex ventral pathway . When CNNs models and monkeys are shown the same picture, the activations of high-level units in the CNN explains half of the variance of random sets of 160 neurons in the monkey's inferotemporal cortex. CNNs have their roots in the neocognitron, the architecture which was somewhat similar, but did not have an end-to-end supervised-learning algorithm such as backpropagation. A primitive 1D CNN called a time-delay neural net was used for the recognition of phonemes and simple words [52].

1.4 Method

A deep-learning architecture is a large multilayer stack of simple modules. For example, convolutional and pooling layers in CNNs are the main building blocks of the architecture. Such building blocks (small trainable multilayer networks) can be trained by the backpropagation algorithm [79] to extract specific features derived from the processing of training datasets.

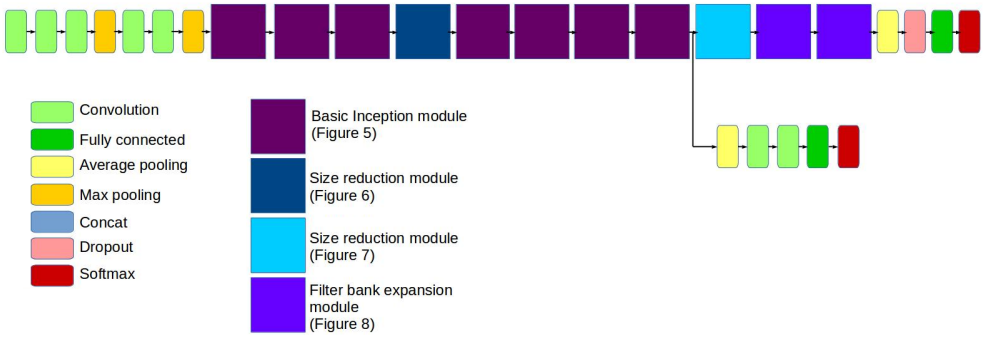


Figure 1.3: Inception v3 architecture [78].

The most straightforward way of improving the performance of (deep) neural networks is by increasing their size. This includes both increasing its depth, i.e., the number of network levels, as well as its width, i.e. the number of units at each level. However, if this principle is applied in the most straightforward way, e.g., by adding new large layers of neurons, fully connetced to the neighboring ones as is usual, this simple solution comes with two major drawbacks. Bigger size typically means a larger number of parameters, which makes the enlarged network more prone to overfitting training data, especially if the number of examples in the training set is limited. The other drawback is the dramatic increase of the computational resources required for the task.

In [56], Google introduced an approach to manage these issues. The architecture of the deep network they propose (see figure 1.3) is based on ”Inception modules” and is inspired by two main concepts: Network-in-Network and sparse networks.

In conventional convolutional layers the input is scanned and processed using linear filters followed by a nonlinear activation function. Instead, Network-in-Network, an approach proposed by Lin et al. [75], builds smaller “micro” neural networks with more complex structures to abstract the data falling within the receptive field (the data neighborhood used as input). As shown in figure 1.1, the feature maps are obtained by convolving the micro networks with the input in a similar manner as with CNNs; such maps then become the input for the next layer. This approach is heav-

ily used in the Inception architectures. Figures 1.6, 1.7, 1.8, and 1.9 show different schemata used in Inception modules.
The next idea is to replace the fully-connected layers by sparsely-connected ones, not only between the main modules, but even within micro networks. Besides mimicking biological systems, this also has the advantage of a firmer theoretical underpinnings according to the groundbreaking work of Arora et al. [80].
Unfortunately, today's computing infrastructures are very inefficient when it comes to numerical calculation on non-uniform sparse data structures. A main idea of the Inception architecture is to consider that the optimal locally sparse structure of a convolutional vision network can be approximated by readily available dense components. The basic Inception modules (figure 1.6) are designed to meet this goal.
Because of its extreme modularization, networks built based on Inception modules are usually very deep, which makes them difficult to analyse or modify. However, they are constructed based on a few design principles, some of which are listed below [81]:

1. Avoid representational bottlenecks, especially early in the network. In other words, the representation size should gently decrease from the inputs to the outputs before reaching the final representation used for the task at hand.
2. Higher-dimensional representations are easier to process locally within a network.
3. Spatial aggregation can be done over lower-dimensional embeddings without much or any loss in representational power.
4. Optimal performance of the network can be reached by properly balancing the number of filters per stage and the depth of the network.

1.4.1 Model

This approach is based on version 3 of the Inception network [78]. It processes RGB images with resolution of 299×299 through a deep path of 54 layers, counting only

layers with parameters to be learned (i.e., not including pooling layers).
One of the most relevant properties of this architecture, if not the most relevant at all, is that an increase in the number of units at each stage does not cause an uncontrolled blow-up in computational complexity in the later stages. This design follows the practical intuition that visual information should be processed at various scales and then aggregated so that the next stage can abstract features at different scales simultaneously [56].
Also, the model uses Softmax for classification and cross-entropy to calculate the loss. This loss was used as the objective function to be minimized by the backpropagation algorithm.
In this model, a single frame evaluation costs 5 billion multiply-adds, using "only" less than 25 million parameters, which is much less than the 60 million parameters of AlexNet.
Different techniques are invented or exploited to build up the model. They are applied in order to follow the design principles mentioned in previous section or to overcome the computational barriers of the model [81].

1×1 convolution

Suppose that we have a convolutional with an output of size (W, H, F), where

- F is the number of convolutional filters.
- W, H data dimensions.

Suppose this output is fed into a convolutional layer with 1×1 filters, zero padding and convolution stride 1. Then the output of this 1×1 convolution layer will have shape (W, H, F1).
So 1×1 convolutional filters can be used to change the dimensionality in the filter space. If $F1 > F$ then we are increasing dimensionality, if $F1 < F$ we are decreasing filter dimension. For example, an image of 100×100 with 50 features on convolution with 20 filters of size 1×1 would result in size of $100 \times 100 \times 20$.
In all the Inception modules, a 1×1 convolution layer is introduced before other

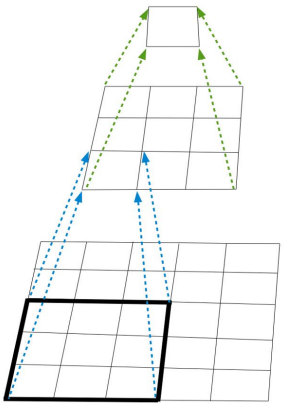


Figure 1.4: Mini-network replacing the 5×5 convolutions [56] (processing flows upwards).

more expensive convolutions. It is used mainly as a dimension reduction module to remove computational bottlenecks.

Factorizing

A main feature of this architecture is the use of different factorizing techniques to implement the convolutions with larger filter support (i.e. the size of the input region). Two main techniques are applied in order to increase computational efficiency:

1. Factorization into smaller convolutions: Convolutions with larger spatial filters (e.g. 5×5 or 7×7) tend to be disproportionally expensive in terms of computation. For example, a 5×5 convolution with n filters over a grid with m filters is $25/9 = 2.78$ times more computationally expensive than a 3×3 convolution with the same number of filters. Of course, a 5×5 filter can capture dependencies between signals between activations of units further away in the earlier layers, so a reduction of the geometric size of the filters comes at a large cost of expressiveness. However, we can ask whether a 5×5 convolution could be replaced by a multi-layer network with less parameters with the same input

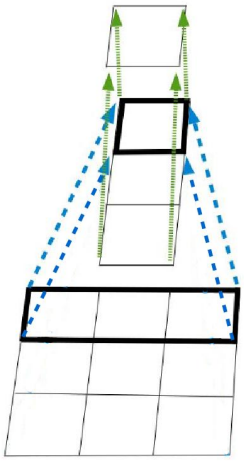


Figure 1.5: Mini-network replacing the 3×3 convolutions. The lower layer of this network consists of a 3×1 convolution with 3 output units [56] (processing flows upwards).

size and output depth. If we zoom onto the computation graph of the 5×5 convolution, we see that each output looks like a small fully-connected network sliding over 5×5 tiles on its input (Figure 1.4).
Since we are constructing a vision network, it seems natural to exploit translation invariance again and replace the fully connected component by a two-layer convolutional architecture: the first layer is a 3×3 convolution, the second is a fully connected layer on top of the 3×3 output grid of the first layer (see Figure 1.4. Sliding this small network over the input activation grid boils down to replacing the 5×5 convolution with two layers of 3×3 convolution .

2. Spatial factorization into asymmetric convolutions: The above results suggest that convolutions with filters larger than 3×3 might not be generally useful as they can always be reduced into a sequence of 3×3 convolutional layers. Still we can ask the question whether one should factorize them into smaller units, for example 2×2 convolutions. However, it turns out that one can do even better than 2×2 by using asymmetric convolutions, e.g. $n \times 1$. For example, using

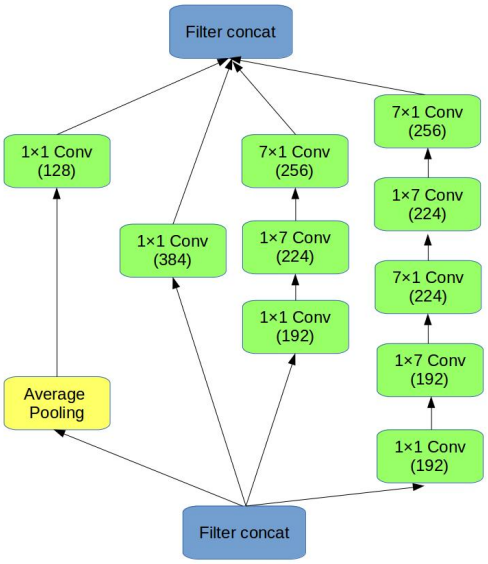


Figure 1.6: Basic Inception module. It is applied on 17×17 grids [82].

a 3×1 convolution followed by a 1×3 convolution is equivalent to sliding a two-layer network with the same receptive field as in a 3×3 convolution (Figure 1.5). Still the two-layer solution is 33% cheaper for the same number of output filters, if the number of input and output filters is equal. By comparison, factorizing a 3×3 convolution into a two 2×2 convolution represents only a 11% saving of computation resources.

Inception modules

In total, four different kinds of Inception modules are used in the structure [81]:

- *Basic modules*: There are seven basic modules (figure 1.6) in the model which are designed to approximate optimal local sparse structure, inspired by the theoretical work of Arora et al [80]. They factorize a bigger 17×17 grid as two consecutive 7×7 convolutions. The filter sizes are set according to principle 3 of section 1.4.

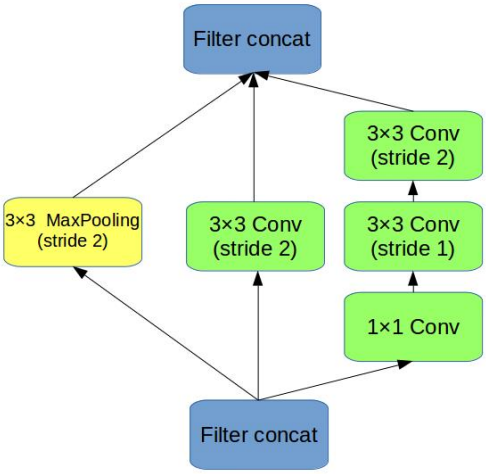


Figure 1.7: Size reduction module: schema for reducing size from 35×35 to 17×17 [81].

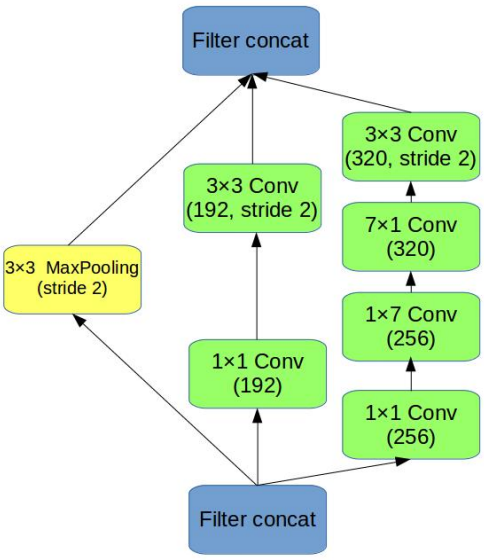


Figure 1.8: Size reduction module: schema for reducing size from 17×17 to 8×8 [82].

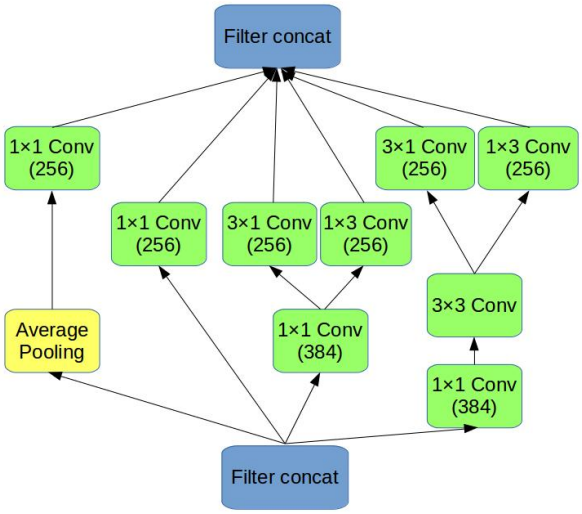


Figure 1.9: Filter bank expansion module: the schema for 8×8 grid modules.

- *Size reduction modules (two types):* Two Inception modules (figures 1.7 and 1.8) with different depths, that reduce the grid size while expanding the number of filter banks. They are used to reduce model dimension wherever the computational requirements would be too heavy otherwise. This solution is both cheap and avoids the representational bottleneck, as suggested by principle 1. The first one is a reduction module from 35×35 to 17×17 and the second one from 17×17 to 8×8.

- *Filter bank expansion module*: Two Inception modules (figure 1.9) are used on the coarsest (8×8) grids to promote high-dimensional representations by expanding the filter bank outputs, as suggested by principle 2.

At the end of each module, the output of all those layers are combined by concatenating them into a single output vector, which forms the input of the next stage.

(a) (b)

Figure 1.10: A few examples of the test dataset's images: (a) UEC FOOD 100 and UEC FOOD 256 and (b) ETH Food-101.

Table 1.2: The best published results without using deep learning.

Method	ETH Food-101		UEC FOOD 100		UEC FOOD 256	
	Top-1	Top-5	Top-1	Top-5	Top-1	Top-5
Random Forest [84]	50.76					
Fisher Vector / AROW [55]			65.32	86.70	52.85	75.51

Auxiliary classifier

Considering the large depth of the network, the ability to propagate error gradients back through all the layers effectively is a concern. The better performance of the backpropagation algorithm with shallower networks on this task suggests that the features produced by the layers in the middle of the network should be very discriminative. By adding auxiliary classifiers connected to these intermediate layers, it is expected that good discrimination properties can be kept up to the final stages of the network, solving the vanishing gradient problem [83] while providing regularization. In the Inception V3, these auxiliary classifiers take the form of smaller convolutional network on top of the output of one of the Inception modules (figure 1.3). During training, its loss is added to the total loss of the network with a discount weight. This auxiliary network affects only training and is ignored in the evaluation phase.

1.4.2 Fine-Tuning

Inception V3 is designed to tackle ImageNet's ILSVRC2012. The training data of ILSVRC2012 contains 1000 categories and 1.2 million images. This network achieved 21.2% top-1 and 5.6% top-5 error for single frame evaluation [78] beating the best published results at the time.
A pre-trained Inception V3 network is used as the starting point to implement the basic a priori knowledge about object recognition, as discussed before. To adapt it to processing food image datasets, the architecture needed to be changed in the classification layers (the last softmax layer and the auxiliary classifier) according to the classes taken into consideration in such sets. Afterwards, starting from the status of the pre-trained network, new training sessions were run using the food image datasets to achieve a fine-tuned version of the model for the problem.
According to the specifications of the following three food image datasets, the classifiers of the model are adapted for dealing with 101, 100, and 256 categories:

- *ETH Food-101*: A dataset of 101 food and dessert categories, with $101,000$ images. The dataset does not include bounding boxes to indicate where the food is located within each image.

- *UEC FOOD 100*:The dataset contains more than $14,000$ images, representing 100 food categories. Each food photo has a bounding box indicating the location of the food item within it. Most food categories in this dataset are popular in Japan. Therefore, some categories might not be familiar to other people than Japanese [85].

- *UEC FOOD 256*: The same as UEC FOOD 100, but considering 256 food categories and including about $32,000$ images [60].

Figure 1.10 shows a few examples of the test datasets' images.

1.5 Experiments

The experiments were run using TensorFlow™, Google's open source software library for numerical computation using data-flow graphs. The system is designed to facilitate research in machine learning, and a quick and easy transition from research prototypes to production [86].
To train such a deep model successfully from scratch, one would have needed millions of images. However, the available food image datasets provide a small fraction of such requirements. In addition, during the fine-tuning phase, a very low value for the learning rate should have been used, which would have made training absolutely infeasible for average computing resources. The latter problem was tackled by starting from the pre-trained model. As for the former, the datasets were artificially extended by applying a series of random distortions to the training images, as follows:

- Randomly cropping the images.
- Resizing the cropped piece to 299×299. This resizing operation may distort the images because the aspect ratio is not respected in the first step.
- Distorting the image brightness.
- Distorting the image contrast.
- Distorting the image saturation.
- Distorting the image hue.

Figure 1.11 shows a few examples of distorted images.
In evaluating the results and comparing them to the best ones obtained on the same data sets, it must be noticed that the ETH Food-101 dataset has been originally divided by its compilers into a training and a test set: 75% of the images for the training set, 25% for the test set. On the contrary, there is no separation or guideline for UEC FOOD 100 and UEC FOOD 256. Therefore, each dataset has been randomly divided into two subsets using 80% of the images for training and the rest for testing.
Moreover, different sets of parameters were used for training each dataset, due to the

different number of categories and of images in each category. In particular, the *initial learning rate* of the ETH Food-101, UEC FOOD 100 and UEC FOOD 256 datasets were set to 5×10^{-3}, 5×10^{-4}, and 5×10^{-4}, respectively. Also, the *number of images per decay* was set, respectively, to $500,000$, $300,000$, and $500,000$. In addition, *batch size* values of 16, 8, and 8 were used, respectively, during training.

The stochasticity introduced by the random initialization of weights in the learning phase has extremely limited effects on this networks, since the method starts from a pre-trained network which requires custom initialization only of the few weights of the specific classification layers we added. In any case, the sensitivity of the results to such variables was assessed by repeating some smaller experiments with the same settings, verifying that the obtained results in the different runs were virtually the same.

The tests were run on a PC equipped with an Intel® Core™ i7 2.8GHz CPU, 24GB RAM, and a Nvidia® GeForce® GTX 960 graphics card. On that PC, the training proceeds at an average speed of $15.6 images/second$. Classifying one image after training takes on average $0.23 s$ (excluding loading of the model and network initialization).

Figures 1.12, 1.13, and 1.14 plot the total loss against time during the training phase. Training is stopped when no further significant improvement had occurred for a specified time interval.

Table 1.2 and Table 1.3 summarize the best published results obtained on the test datasets, to the best of author's knowledge, without and with using deep learning, respectively. As one can see, using CNN has produced much better results in general. In both tables, the first row shows the results published in [84]. The authors tested different approaches and achieved their best results using Random Forest, when not using a CNN-based approach. When using CNNs, they did not pre-train the CNN, but trained it from scratch. The second row of each table shows the results published in [55]. Without using CNNs, they achieved the best results by extracting RootHoG patches and color patches, and coding them into Fisher Vectors. Their best deep-learning-based results were achieved by pre-training an AlexNet on food-related images before the fine-tuning phase.

Figure 1.11: Artificially distorted images for the training.

Table 1.3: The best published results previously obtained using deep learning compared to the results of fine-tuned Inception V3.

Method	ETH Food-101 (%)		UEC FOOD 100 (%)		UEC FOOD 256 (%)	
	Top-1	Top-5	Top-1	Top-5	Top-1	Top-5
CNN [84]	56.40					
DCNN-FOOD [55]	70.41		78.77	95.15	67.57	88.97
Inception V3	**88.28**	**96.88**	**81.45**	**97.27**	**76.17**	**92.58**

As shown in table 1.3, our method's results beat all the other approaches with significant gaps. Accuracy increase of 2.68% and 8.6% were achieved on UEC FOOD 100 and UEC FOOD 256. However, the best results were obtained on the ETH Food-101 dataset, on which a dramatic increase of 17.87% could be obtained. Besides the other possible causes, related with the fact that the best results on the three datasets were obtained in independent experiments, the reason for the larger increase in accuracy obtained on the ETH Food-101 dataset could be due to the higher number of images per category available in ETH Food-101. It includes, on average, 1000 images per category, which permits a better fine-tuning with respect to the other databases that, on average, include less than 150 images per category.

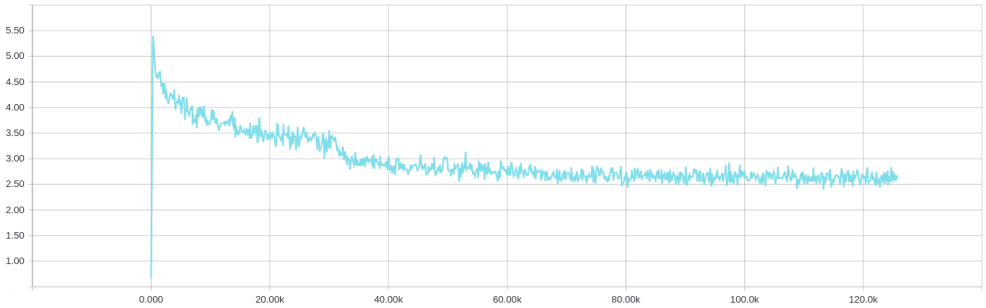


Figure 1.12: Total loss of the model for ETH Food-101 dataset. The horizontal axis represents the number of batches (batch size: 16).

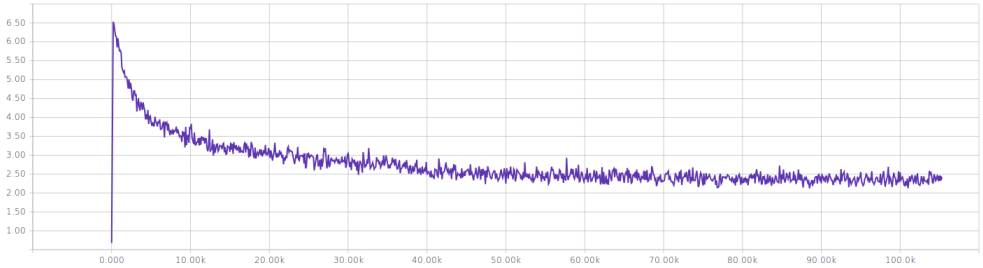


Figure 1.13: Total loss of the model for UEC FOOD 100 dataset. The horizontal axis represents the number of batches (batch size: 8).

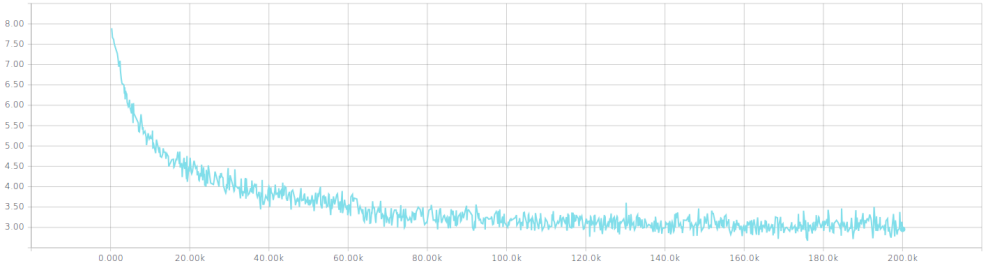


Figure 1.14: Total loss of the model for UEC FOOD 256 dataset. The horizontal axis represents the number of batches (batch size: 8).

1.6 Discussion

The method proposed in this thesis uses a new deep convolutional neural network model recently introduced by Google, Inception V3. The model benefits from new building blocks, called "Inception modules", which are used to produce very deep networks. Inception modules are forged based on a few guidelines for designing DCNNs.
In these experiments, the resulting models have beaten the best competitors in terms of increased accuracy, decreased computational cost, and number of parameters.
It is shown that the fine-tuned versions of Inception V3 can achieve very good results on three main food image datasets improving the best published results by a relevant amount. It achieved 88.28%, 81.45%, and 76.17% accuracy, respectively, on ETH Food-101, UEC FOOD 100, and UEC FOOD 256 which were about 17.87%, 2.68%, and 8.6% better than the best published results.
These results may have a relevant impact on mobile computing. Up to now, automatic food recognition on mobile devices has not appeared to be a viable solution. In fact, the methods proposed thus far had not reached an accuracy level which allowed them to be used in practical systems. Therefore, that the best option for automatic diet monitoring appears to be a semi-automatic or manual approach to the food recognition step of the process. However, the effectiveness of Inception in classifying food images suggested opposite consideration. In fact, considering the number of parameters and the consequent computational cost, the new approach could actually be a viable choice for mobile applications, which are the primary environment for automatic diet monitoring systems.
These results, despite being dramatically better than those previously published, are still preliminary, deriving from a rather straightforward adaptation of Google's pretrained network to the task. More studies on this new approach to deep learning could lead to even better results. Adjusting the method for delivering optimal performance on mobile devices is also a future priority.
A problem that is probably worth discussing is related with the huge computational resources required to train the whole network from scratch. If the network had been

trained with the computational resources used to fine-tune the pre-trained network, each training session would have taken more than one month. Considering that a number of training session is necessarily required to tune some of the network parameters for reaching the best possible accuracy, it is clear that organizing and carrying out such experiments is not trivial at all and, apparently, restricted to research or commercial groups which can rely on huge computer clusters. However, one important result of this study is showing that fine-tuning a pre-trained network can achieve good results in a reasonable time.

Because of this, considering also the need for further investigation on the settings of the training/fine-tuning parameters and of the architecture for the classification layer added to the pre-trained network, deep learning and its applications, food recognition in particular, appear to offer many opportunities for research at all levels.

Chapter 2

Segmentation

Detection of the food item in the image or *segmentation* is a basic requirement for food amount estimation. It is the first and the most challenging part of an automatic image analysis process.

Segmentation is a pre-process which partitions an image into unique multiple regions, where a region is a set of pixels [87]. Conventionally, segmentation can be grouped into five categories [88]. The first one includes threshold based segmentation methods [89]. These methods usually divides the image into two sets, namely foreground and background. When the intensity of the pixels is larger/smaller than a predefined threshold, those pixels are classified as foreground. Otherwise, they will be considered background. Threshold-based approaches are the simplest, easiest and fast ones among all of the existing segmentation methods. The drawback is that it is not easy to find an appropriate threshold which can separate the image into two groups directly. This method also requires that foreground and background in the image have obviously different intensity values. [88]

The second category is edge-based segmentation [89]. This method is by far the most common approach for detecting meaningful discontinuities in gray level[2]. First and second order derivatives like gradient and laplacian are used for detection of edges in

an image. In practice, Edge based techniques generate many noise which are not easy to distinguish from the real edges. Hence edge detection algorithms need additional post processing by using linking procedures to assemble edge pixels into meaningful edges. [87]
The third category is region-based segmentation [89]. It includes region growing and region splitting/merging procedures. Region growing procedure groups pixels or subregions into large regions based on predefined criteria. Initial set of 'seed' points are created and from these points other regions grows up if neighbouring pixels have similar properties as the 'seed' point. Selection of seed points is a critical procedure for color images if a priori information is not available. In region splitting/merging, an image is subdivided into arbitrary, disjoint regions that are either merged and/or split to satisfy prerequisite constraints. [87]
The forth category is watershed-based segmentation [89]. The Watershed transformation considers the gradient magnitude of an image as a topographic surface. Pixels having the highest gradient magnitude intensities (GMIs) correspond to watershed lines, which represent the region boundaries. Water placed on any pixel enclosed by a common watershed line flows downhill to a common local intensity minima (LMI). Pixels draining to a common minimum form catchments basins, which represent the regions. [87]
The fifth category is energy-based segmentation. This method need to establish an objective (energy) function which will reach a minimal value when the image is segmented as expected. Graph cut can be categorized to belong to this class. The energy function of graph cut segmentation is based on regional and boundary information and can achieve globally optimal result. [88]

In the context of food image analysis, segmentation implies an exact detection of the regions occupied by food items within the picture. Identifying these regions is essential for recognizing the food type and estimating its quantity. This task has been tackled using many different image segmentation techniques. These include, among others, graph-based algorithms, like normalized cut, graph cut, or connected-component labeling [90]; region growing, that might require that a set of seeds be

specified to initialize the process; edge detection, to find object borders [91]; pixel labeling, based on the distance of pixels' local properties from pre-defined models [92].
Food image segmentation can be very challenging. There are very different ways of cooking the same kind of food and of arranging food on a dish. These problems and many other factors cause food appearance to vary dramatically. Environmental conditions, like lighting or image background, may vary significantly as well.
Some studies impose constraints when taking food images, to make segmentation easier. For example, Zhu [36] assumes that food lies on a dish that is brighter than the table cloth. Several other studies [47, 48, 72, 93, 94, 95] impose that the dish, or another container where the food lies, have a pre-specified shape.
Some studies have combined different segmentation algorithms to achieve better results. For example, Zhu et al. [96] concurrently use different graph-based, region-growing, and edge detection algorithms to segment food items. Although combining different algorithms may improve results in some cases, it does not solve all the problems.
Other cases consider food recognition and segmentation as two cooperating, instead of cascaded, tasks. In [57], the authors combine image segmentation and classification introducing a segmentation refinement step that relies on the information provided by the classifier. Also, in [37], segmentation is based on a region-growing method that considers multiple feature spaces. All the segments extracted are then fed into a set of Support Vector Machines (SVMs). The SVMs assign to each food type/region pair a score based on which one can identify the most likely segment corresponding to each type of food.
Recently, attention has been paid to semi-automatic methods. In [44, 55], the user is requested to draw a bounding box around the food items and in [37] the user must indicate the initial seeds before starting to grow segments.
In this work, a new approach to segmenting food images is proposed. A mobile app, which enables the user to easily mark different parts of the image as food or non-food area, has been developed. Then, a customized iterative graph cut algorithm has been used to perform a complete segmentation of the image. This approach allows one to have highly accurate segmented images in the subsequent phases such as food clas-

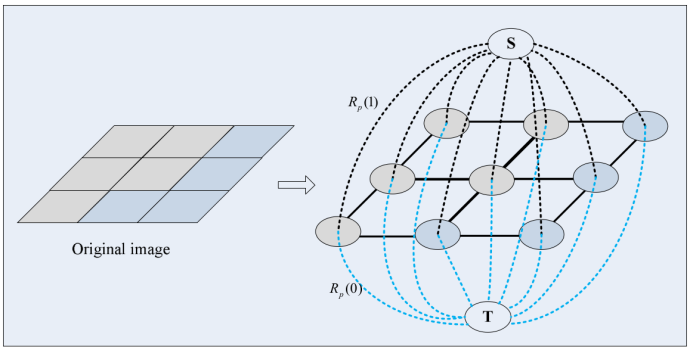


Figure 2.1: Illustration of s-t graph. The image pixels correspond to the neighbor nodes in the graph(except s and t nodes). The solid lines in the graph are n-links and the dotted lines are t-links. [88]

sification and food amount estimation. The application is very easy to use, with no specific limitation about food types or the number of food items in the image. The proposed approach has been tested on various food images taken by mobile phones in different situations.

2.1 Graph Cut

Let an undirected graph be denoted as $G =< V, E >$ where V is a series of vertices and E is the graph edge which connects every two neighbor vertices. V is composed of two different kinds of nodes (vertices). The first kind of vertices is neighborhood nodes which correspond to the pixels and the other kind of vertices are called terminal nodes which consist of s (source) and t (sink). This kind of graph is also called $s-t$ graph where, in the image, s nodes usually represent the object while t nodes denote the background. In this kind of graph, there are also two types of edges. The first type of edges are so-called n-links which connect neighboring pixels within the image (4-connected system in the 2D image is used in this work). And the second type of edge are so-called t-links which connect the terminal nodes with the neighboring nodes. In

this kind of graph, each edge is assigned a non-negative weight denoted as w which is also named cost. A cut is a subset of edges E which can be denoted as C and expressed as $C \subset E$. The cost of the cut $|C|$ is the sum of the weights on edges C which is expressed as follows:

$$|C| = \sum_{e \in C} w_e \quad (1)$$

A minimum cut is the cut that have the minimum cost (called min-cut) and can be found by finding the maximum flow. In [97, 98] it is shown that the min-cut is equivalent to max-flow. The max-flow/min-cut algorithm developed by Boykov and Kolmogorov [98] can be used to find the minimum cut for the s-t graph. Thus, the graph is divided by this cut and the nodes are separated into two disjoint subsets S and T where $s \in S$, $t \in T$ and $S \cup T = V$. The two subsets correspond to the foreground and background in the image segmentation. This kind of graph is depicted in figure 2.1.

Moreover, image segmentation can be regarded as pixel labelling problems. The label of the object (s-node) is set to be 1 while that of the background (t-node) is set to be 0 and this process can be performed by minimizing the energy-function through minimum graph cut. In order to make the segmentation reasonable, the cut should occur at the boundary between object and the background. Namely, at the object boundary, the energy (cut) should be minimized. Let $L = \left\{ l_1, l_2, l_3, ..., l_i, ..., l_p \right\}$ where p is the number of the pixels in the image and $l_i \in \left\{ 0, 1 \right\}$. Thus, the set L is divided into two parts and the pixels labeled with 1 belong to the object while the others are grouped into background. The energy function is defined as following equation which can be minimized by the min-cut in the s-t graph [88].

$$E(L) = \alpha R(L) + B(L) \quad (2)$$

where $R(L)$ is called regional term and incorporates the regional information into the segmentation and $B(L)$ is called boundary term and incorporates the boundary constraint into segmentation; α is the relative importance factor between the regional and the boundary term. When α is set to be 0, it means that the regional information is ignored and that only the boundary information is considered. In the energy function in eq. (2), the regional term is defined as:

$$R(L) - \sum_{p \in P} R_p(l_p) \quad (3)$$

where $R_p(l_p)$ is the penalty for assigning the label l_p to pixel p. The weight of $R_p(l_p)$ can be obtained by comparing the intensity of pixel p with the given histogram (intensity model) of the object and background. The weight of the t-links is defined as following equations [88]:

$$R_p(1) = -ln(Pr(I_p|\text{“}forground\text{”})) \quad (4)$$

$$R_p(0) = -ln(Pr(I_p|\text{“}background\text{”})) \quad (5)$$

From eq. (4) and (5), we can see that when $Pr(I_p|\text{“}forground\text{”})$ is larger than $Pr(I_p|\text{“}background\text{”})$, $R_p(1)$ will be smaller than $R_p(0)$. This means when the pixel is more likely to be the object, the penalty for grouping that pixel into object should be smaller, which reduces the energy in eq. (2). Thus, when all of the pixels have been correctly separated into two subsets, the regional term would be minimized. $B(L)$ in eq. (1) is the boundary term which is defined as [88]:

$$B(L) = \sum_{\{p,q\} \in N} B_{<p,q>}.\delta(l_p, l_q) \quad (6)$$

where p, q are neighboring pixels and $\delta(l_p, l_q)$ is defined as:

$$\delta(l_p, l_q) = \begin{cases} 1 \; if \; l_p = l_q \\ 0 \; if \; l_p = l_q \end{cases} \quad (7)$$

For the regional constraint, it can be interpreted as assigning labels l_p , l_q to neighboring pixels. When the neighboring pixels have the same labels, the penalty is 0, which means the regional term would only add the penalty at the segmented boundary. For the term $B_{<p,q>}$, it is defined to be a non- increasing function of $|I_p I_q|$ as follows: [88]

$$B_{<p,q>} \propto exp\left(-\frac{(I_p - I_q)^2}{2\delta^2}\right) \quad (8)$$

where δ can be viewed as camera noise. When the intensity of two neighboring pixel is very similar, the penalty is very high. Otherwise, it is low. Thus, when the energy function reaches the minimum value, it is more likely to be at the object boundary. In [97], Boykov and Jolly have showed that the minimized energy can be computed by the min-cut through max-flow. Thus, the minimum energy problem is converted into the graph-cut problem. In order to obtain a reasonable segmentation, the assignment of the weight in the s-t graph is very important. In Boykov and Jolly's method, the weight of the s-t graph is given as following:

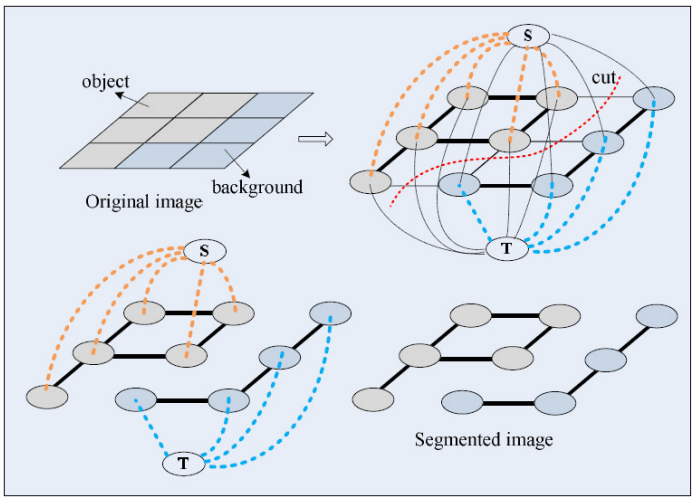


Figure 2.2: Illustration of graph cut for image segmentation. The cut corresponds to the minimal energy of eq.(2). [88]

$$weight = \begin{cases} B_{<p,q>} & \{p,q\} \in Neighboring\ pixel \\ \alpha.R_p(0) & for\ edge\ \{p,S\} \\ \alpha.R_p(1) & for\ edge\ \{p,T\} \end{cases} \quad (9)$$

Eq.(9) can also be explained as follows: in the s-t graph, when the intensity of the pixel is likely to represent the object, the weight between this pixel and the s-node will be larger than that between pixel and t-node, which means the cut is more likely to occur at the edge with smaller weight. For the neighboring pixels, when their intensity is very similar, the weight is very big, so it is not likely to be separated by the cut. Thus, when the minimum cut is achieved from the s-t graph, the location of the cut is close to the object boundary. The implementation of the graph-cut algorithm can be fulfilled by the max-flow/min-cut as described in [97]. In figure 2.2, the graph cut for a 3×3 image segmentation is illustrated. The thickness of the edge denotes the magnitude of the weight.

For most images, it is difficult to perform purely automatic segmentation. Especially for the natural images and images for which the accuracy requirements of target seg-

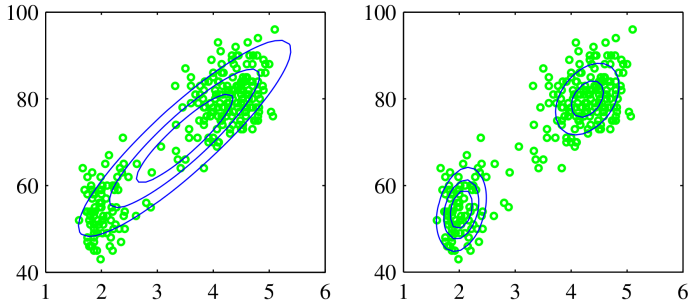


Figure 2.3: Plots of the 'old faithful' data in which the blue curves show contours of constant probability density. On the left is a single Gaussian distribution which has been fitted to the data using maximum likelihood. Note that this distribution fails to capture the two clumps in the data and indeed places much of its probability mass in the central region between the clumps where the data are relatively sparse. On the right the distribution is given by a linear combination of two Gaussians which has been fitted to the data by using a Mixture Model. [101]

mentation are very high, interactive segmentation is inevitable. Interactive graph cut varies from easily choosing the object region of interest or simple seed points to iteratively seed point selection. In [99], bounding boxes are used to select the interested object area and in [100], follows shape prior-based graph cut approach, which take the shape prior of object into consideration to make the segmentation more reasonable.

2.2 Gaussian Mixture Model (GMM)

The Gaussian, also known as the normal distribution, is a widely used model for the distribution of continuous variables. In the case of a single variable x, the Gaussian distribution can be written in the form

$$N(x|\mu,\sigma^2) = \frac{1}{\left(2\pi\sigma^2\right)} exp\left\{-\frac{1}{2\sigma^2(x-\mu)^2}\right\} \quad (10)$$

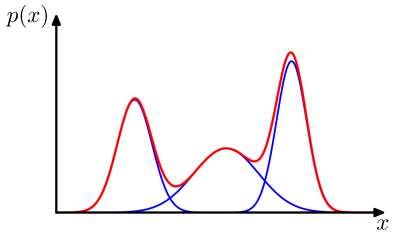


Figure 2.4: Example of a Gaussian mixture distribution $p(x)$ in one dimension showing Two Gaussians in blue and their sum in red [101].

where μ is the mean and σ^2 is the variance. Gaussian distributions are observed in many different contexts and can be motivated from a variety of different perspectives. [101]

However, while the Gaussian distribution has some important analytical properties, it suffers from significant limitations when it comes to modelling real data sets. Consider the example shown in figure 2.3. This is known as the 'Old Faithful' data set, and comprises 272 measurements of the eruption of the Old Faithful geyser at Yellowstone National Park in the USA. Each measurement comprises the duration of the eruption in minutes (horizontal axis) and the time in minutes to the next eruption (vertical axis). We see that the data set forms two dominant clumps, and that a simple Gaussian distribution is unable to capture this structure, whereas a linear superposition of two Gaussians gives a better characterization of the data set.

Such superpositions, formed by taking linear combinations of more basic distributions such as Gaussians, can be formulated as probabilistic models known as mixture distributions. In figure 2.4 we see that a linear combination of Gaussians can give rise to very complex densities. By using a sufficient number of Gaussians, and by adjusting their means and covariances as well as the coefficients in the linear combination, almost any continuous density can be approximated to arbitrary accuracy. [101]

We therefore consider a superposition of K Gaussian densities of the form

$$p(x) = \sum_{k=1}^{K} \pi_k N(x|\mu_k, \Sigma_k) \quad (11)$$

which is called a mixture of Gaussians. Each Gaussian density $N(x|\mu_k, \Sigma_k)$ is called

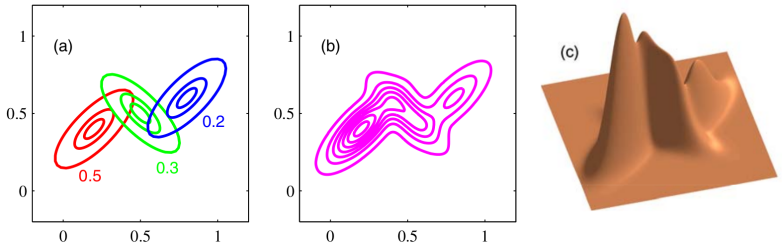


Figure 2.5: Illustration of a mixture of 3 Gaussians in a two-dimensional space.(a) Contours of constant density for each of the mixture components, in which the 3 components are denoted red, blue and green, and the values of the mixing coefficients are shown below each component. (b) Contours of the marginal probability density $p(x)$ of the mixture distribution. (c) A surface plot of the distribution $p(x)$ [101].

a component of the mixture and has its own mean μ_k and covariance Σ_k . Contour and surface plots for a Gaussian mixture having 3 components are shown in figure 2.5. [101]
One way to use gaussian mixture to describe the data distribution is clustering data to a pre-defined number of clusters and then estimate a Gaussian mean and covariance for each cluster.

2.3 Method

In this study, an interactive graph cut algorithm has been developed for segmentation of the food images. The user chooses one of the images captured by the mobile device (see section 3.1). Then she/he marks some regions on the image as foreground(food) and some other as background (not food) (Figure 2.6). Then, all the images are segmented based on this marks by the developed dynamic graph cut. Although, the user can modify the marking if the segmentation result is not satisfying. Thus, every time when the result is not perfect, more seeds will be added and the segmentation result can be revised until the object of interest is represented well enough. So, in the case of difficult images, the user can ideally choose the object and background seeds along

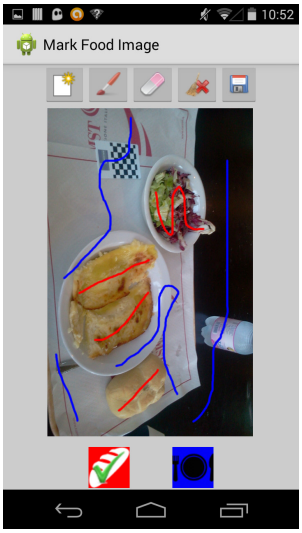


Figure 2.6: The android application designed for semi-automatic image segmentation.

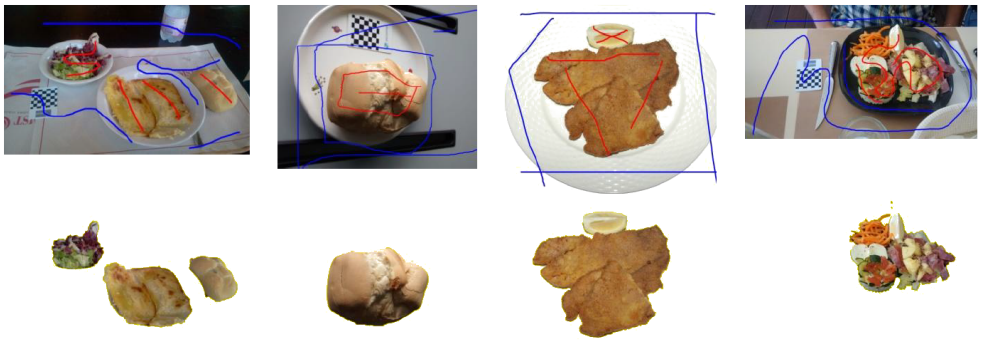


Figure 2.7: A few examples of the marked images and the segmentation results.

the weak edge.

The applied graph-cut algorithm is adapted from the approach presented in [102]. There are two major characteristics in the method which makes it different from a conventional graph cut. First, graph cut performs poorly when segmenting images in which the features of the foreground and background are not highly discriminative. This algorithm works by representing the colour distribution of each pixel position in the image as a Gaussian Mixture Model (GMM). Firstly, a GMM for the pixels included in the marked part of the image is estimated. Then, K-Means is used to divide data into 10 clusters. the mean and covariance of each cluster are estimated to form a GMM. Next, the likelihoods for a pixel to represent background or foreground obtained by this technique are integrated in the graph cut.

Second, since this method uses several images for a 3D reconstruction process, it

Algorithm 1 Segmentation of a marked image using GMM and graph cut.

function SEGMENTATION(OriginalImage, MarkedImage)

$Pixels_F \leftarrow$ **ExtractFoodPixelsFromMarks**$(OriginalImage, MarkedImage)$
$Pixels_B \leftarrow$ **ExtractBackgroundPixelsFromMarks**$(OriginalImage, MarkedImage)$

$GMM_F \leftarrow$ **ComputeFoodGMM**$(Pixels_F)$
$GMM_B \leftarrow$ **ComputeBackgroundGMM**$(Pixels_B)$

$Like_F \leftarrow$ **CalcFoodLikelihood**$(OriginalImage, GMM_F)$
$Like_B \leftarrow$ **CalcBackgroundLikelihood**$(OriginalImage, GMM_B)$

$Segments \leftarrow$ **GraphCut**$(Like_F, Like_B)$

return Segments

end function

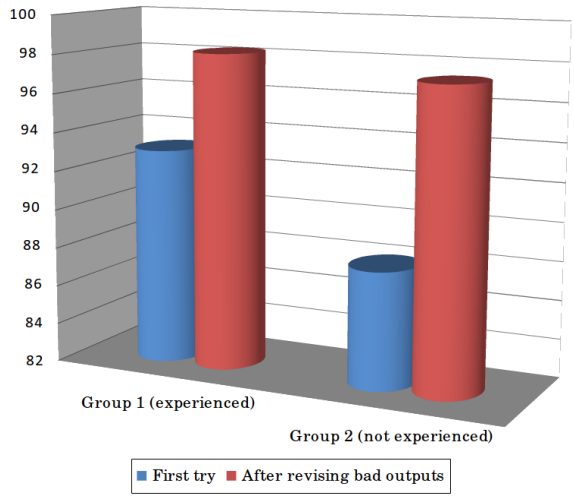
needed to speed up the process while minimizing the user cooperation for the manual part. This algorithm records the flow obtained during the computation of the max-flow corresponding to a particular problem instance. This recorded flow is used as an

initialization in the process of finding the max-flow solution corresponding to the new problem instance. This method belongs to a broad category of algorithms which are referred to as dynamic. These algorithms solve a problem by dynamically updating the solution of the previous problem instance. Their goal is to be more efficient than a re-computation from scratch of the solution after every change. Given a directed weighted graph, a fully dynamic algorithm should allow for unrestricted modification of the graph. The modification may involve addition and deletion of nodes and edges in the graph as well as changes in the cost (capacity) of any graph edge [103]. In particular, the colour distribution of pixels calculated in the first image is used in the rest of the images. Considering the similarity of the objects in the images, this provides a proper starting point while saving computational cost.

2.4 Experiments

A test of the application on a data set including different kinds of images has been carried out. The images were taken by mobile phones in different places and in uncontrolled conditions. Table 2.1 shows the result of the segmentations.
Two groups of users have used the application. The first group included people who are familiar with the application and the second group included users who are not familiar with the application and who had no prior experience with it. The application and its goal were shortly explained to later group. In addition, after marking the images, all participants were shown the results of the segmentation and, in the cases with poor output; they were asked to modify their marks. If the segmentation consisted, in total, of more than 5% of false pixels it is considered as a poor outcome. As can be seen in figure 4, experienced users reached up to 93.1% accuracy in the first try while the second group obtained 88%. However, when they revised their marking for the bad outputs, the two groups had similar results (98.2% and 97.3%, respectively).

Table 2.1: Successful segmentation rates. A segmentation is considered successful if there are less than 5% of false segmented pixels.



2.5 Discussion

A semi-automatic interactive graph cut algorithm was used to tackle the segmentation problem. The user marks some food and non-food areas of images and the system runs a customized graph-cut, which uses GMM for labeling, to segment them. The experiment shows that the system achieves 93% accuracy in the first try. Moreover, there was the possibility of revising the marks by the user to improve the result.
Semi-automatic approaches are worth exploring since every mistake in the segmentation phase could result in larger errors in the following steps of the process. This approach becomes even more worth considering more complex cases. For instance, with the image of a full meal, including different food items and drinks. This is why most studies consider only a single food item at one time. A viable solution to this problem can be offered by semi-automatic approaches, based on practical guidelines or interactive procedures.

Chapter 3

Portion Estimation

Estimating the size or volume of food items, once the food type is known, is the most direct approach to measuring the nutrition properties of the food pictured in an image. However, there are two problems to be solved. One is related to the two-dimensional nature of pictures; the other is related to the fact that object sizes within an image can be measured only up to a scale factor when no size reference is available.

Several methods try to estimate food size from a single image. Martin et al. assume that a correlation exists between the area and the volume. The formula that relates area and volume depends on the food type and the shape of the dish that contains it [92]. In [104], structured light is used for estimating size from a single image. To achieve the same goal, the system described in [105] first detects the camera pose using a fiducial marker placed within the scene and then tries to match the food item generating several pre-defined 3D shape models. This approach works only for the items that can be approximated by the regular shapes it takes into consideration.

Many studies have tried to use more than one image to overcome the problems related to the lack of depth information in two-dimensional images and occlusions that may hide some objects. In [49], two images provide top and side views of the dish. In [35], the user takes three pictures from which the system computes a dense

3D point cloud including all points on the dish and then estimates the volume using Delaunay triangulation. Some studies allow the user to take a short video rather than a few images [70]. However, since these studies based on standard 3D reconstruction methods are still far from achieving perfect reconstructions of the 3D objects, volume estimation by off-the-shelf algorithms may not be a proper choice. Based on this consideration, [106] proposes a new slice-based volume estimation approach customized for food items.

Another major challenge for 3D reconstruction algorithms is the high computational burden they impose. Long processing time is a serious problem if the whole system is designed to run on a mobile device. Dehais et al. [107] suggest a solution that is computationally more affordable. However, it still takes around 15 seconds to estimate the volume when run on a common desktop computer.

As already mentioned, one needs to have a size reference to estimate food size, regardless of the chosen method. Several different objects have been proposed as references. Villalobos [94] suggests that the user include her/his thumb in the scene while taking the photo, and later use it as the size reference. In another work [108], a circular dish of known size is the reference. Some other studies suggest a checkerboard or some other easily detectable pattern as the necessary reference [106, 107, 109].

Asking the user to estimate the food amount would obviously be the simplest approach. The system described in [44, 60] requires that the user roughly indicates the volume of the selected food item using a slider at the bottom of the screen. However, since people are known to be inaccurate in estimating food amount, a fully automated approach would be preferable.

In this chapter a complete solution to estimate food volume is introduced. The system has been designed as a client-server solution. The smartphone acts as the client and runs the semi-automatic part of the system.
In this system, first, the user is asked to take a short video of the food. Then, a set of the frames are automatically extracted from the video aiming to provide a complete panorama from all sides of the food. Next, the user marks some food and non-food parts on one of the selected frames of the video. A mobile application is developed

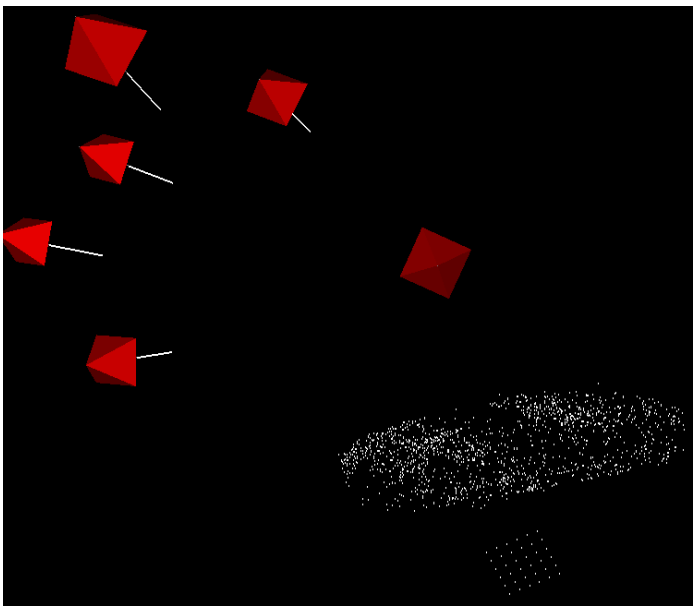
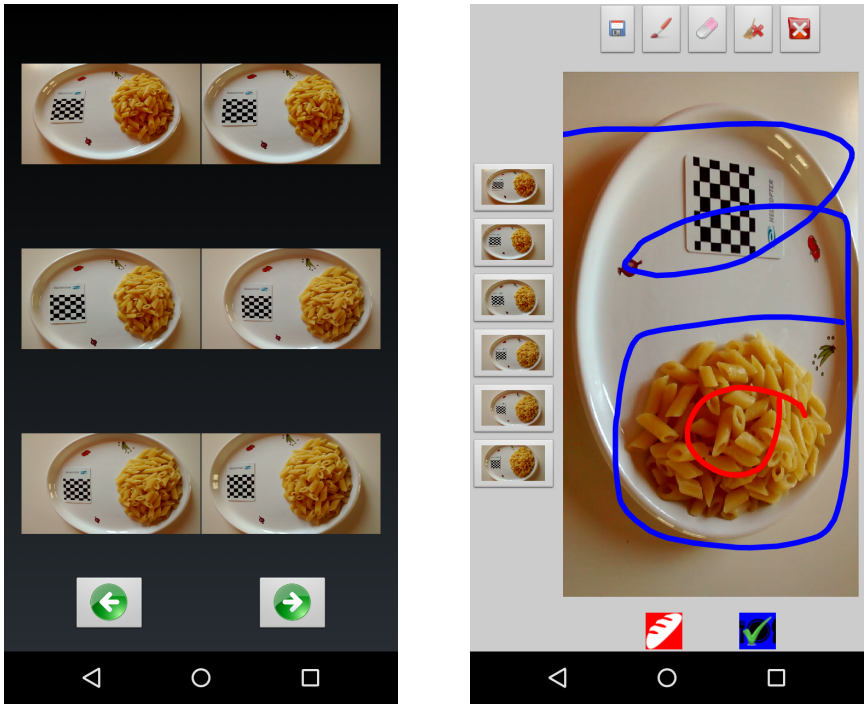


Figure 3.1: Camera positions on the six selected frames are demonstrated alongside the reconstructed model of a sample object.

to facilitate this part. User's marks are used during the segmentation process which is performed on the server side. The segmented images are fed into a customized image-based modeling process resulting in a 3D model of the food items. This model is then used to calculate the volume and to finally estimate the nutrient facts of the food intake. A small checkerboard is used as the size and ground reference through the modeling process.

3.1 Image Acquisition

3D-reconstruction of an object needs enough visual information of the whole object. Usually, in works like [35, 107] the reconstruction is based on two or three images. The main problem with such a design is the lack of visual information of



(a) selected frames (b) manual marking

Figure 3.2: A snapshot of application. After taking a short video by the user, six frames are chosen automatically (a) and marked by the user (b)

all sides of the object which may lead to an incomplete reconstruction. Besides, in 3D-reconstruction approaches based on visual features, there are different steps of the process, such as feature matching or fundamental matrix estimation, which might suffer from noise or lack of relevant information. Consequently, using only two images might generate a very poor model of the object.
In this method more images are used to have better coverage of the whole object. This also allows the method to achieve satisfying results using relatively low-resolution images.

The user is asked to take a short video of the object by moving the camera around the object for a few seconds. Then, the application selects six frames which cover all the sides of the object using the data gathered from the orientation sensor of the smartphone. The frame selection process is performed almost in real time, so the user can use the application smoothly. Figure 3.1 shows an example of camera positions which are automatically selected by the application.
The built-in sensors of the mobile device are used for the selection process. Most of the modern smartphones are equipped with different kind of sensors among which accelerometer and geomagnetic field sensors are very common. These sensors can provide enough data to detect the orientation of the smartphone in each moment. The orientation of the device is defined by three value: pitch, roll, and azimuth, which, respectively, corrispond to the angle of rotation around the x-axis, y-axis, and z-axis. The application records the time-tag of the frames that coincide with the min/max values of pitch, roll, and azimuth recorded while taking the movie. Algorithm 2 reports the pseudo code of the function used to detect the proper frames. Later, six frames presenting the min/max rotational moments are chosen as the input of the modeling process.
In this method, the user neither needs to take several images nor should worry about the proper camera viewpoint. Moreover, if the user decided to eliminate one of the images or replace it by some other frame, this could easily be done by tapping on the screen. Figure 3.2 shows two snapshots of the mobile application. After taking a short video, six frames are extracted and after marking they will be ready for the next steps.

3.2 3D Reconstruction

In this section the main concepts and techniques, which are used in this thesis to reconstruct a 3D model of the food, are described. The next sections specify their participation in our method.

Algorithm 2 Choosing six frame to acquire a good coverage of the object.

```
function ONSENSORCHANGED(SensorEvent event)

    if event.sensors include [ACCELEROMETER, MAGNETIC_ FIELD] then
        pitch, roll, azimuth ← SensorManager.getOrientation()
        if pitch is a new pitch min/max then
            RememberTimeTag()
        end if
        if roll is a new roll- min/max then
            RememberTimeTag()
        end if
        if azimuth is a new azimuth- min/max then
            RememberTimeTag()
        end if
    end if
end function
```

3.2.1 Camera Calibration

Camera calibration, also referred to as camera resectioning, is the process of estimating the parameters of a pinhole camera model approximating some physical specifications of the lens and image sensor of the camera. Each camera lens has unique parameters, such as focal length, and lens distortion model. Also, the principal point defines the relative position of the lens and the image sensor. These parameters, which are called intrinsic camera parameters, can be used to correct lens distortion, measure the size of an object in world units, or determine the location of the camera within the scene. Camera calibration is a basic step of 3D computer vision in extracting the scaled measures from 2D images.
The camera calibration results in two matrices: camera matrix and distortion coefficients matrix.

The camera matrix is a 3×3 matrix which describes the projection of the points in the physical world onto the image:

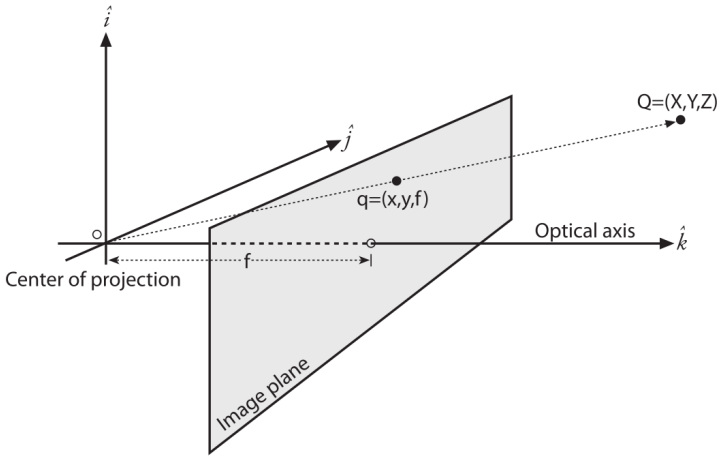
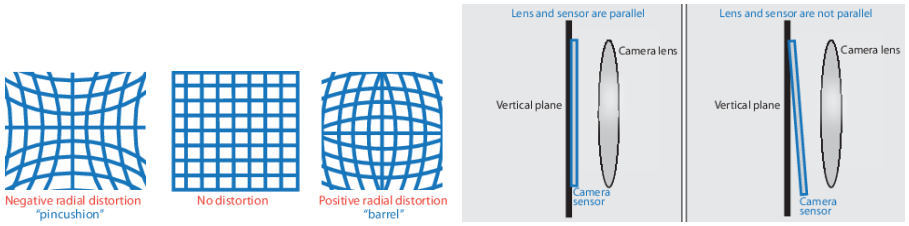


Figure 3.3: Pinhole camera model. [110]

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

The camera matrix essentially describes two parameters: focal length and principal point. In the pinhole camera model, light is envisioned as entering from the scene or a distant object, but only a single ray enters from any particular point. In a physical pinhole camera, this point is then 'projected' onto an imaging surface (see figure 3.3). As a result, the image on this image plane (also called the projective plane) is always in focus, and the size of the image relative to the distant object is given by a single parameter of the camera: its focal length. [110]

The point at the intersection of the image plane and the optical axis is referred to as the principal point. One might consider the principal point to be equivalent to the center of the imager; yet this would imply that some guy with tweezers and a tube of glue was able to attach the imager in your camera with micron accuracy. In fact, the center of the chip is usually not on the optical axis. We thus introduce two new parameters, c_x and c_y , to model a possible displacement (away from the optic axis) of the center of coordinates on the projection screen. The result is that a relatively simple model in which a point Q in the physical world, whose coordinates are (X,Y,Z), is projected



(a) Radial Distortion (b) Tangential Distortion

Figure 3.4: Lens Distortions. [111]

onto the screen at some pixel location given by (x_{screen}, y_{screen}) in accordance with the following equations: [110]

$$x_{screen} = f_x \left[\frac{X}{Z}\right] + c_x \qquad y_{screen} = f_y \left[\frac{Y}{Z}\right] + c_y$$

The projection of a point in the physical world (Q) onto the camera (q with homogeneous coordinates) is now summarized by the following simple form: [110]

$$q = MQ, where \;\; q = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \; M = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}, \; Q = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

Moreover, the distortion coefficients matrix is a 1×5 matrix describing two distortions: radial distortion and tangential distortion (see figure 3.4):

$$D = \begin{bmatrix} k_1 & k_2 & k_3 & p_1 & p_2 \end{bmatrix}$$

The first three parameters define the radial distortion of the lens. In theory, it is possible to define a lens that will introduce no distortions. In practice, however, no lens is perfect. This is mainly related to manufacturing flaws; it is much easier to make a 'spherical' lens than to make a more mathematically ideal 'parabolic' lens. Radial distortions arise as a result of the shape of lens. The lenses of real cameras often noticeably distort the location of pixels near the edges of the imager. This bulging phenomenon is the source of the 'barrel' or 'pincoushion',also called 'fish-eye', effect. [110]

The second-largest common distortion is tangential distortion. This distortion is due to manufacturing defects resulting from the lens not being exactly parallel to the imaging plane. This distortion is defined by the last two parameters of the matrix.

3.2.2 Feature Detection and Matching

In computer vision, tha process of deciding what to focus on is called feature detection. A feature in this sense can be formally defined as "one or more measurements of some quantifiable property of an object, computed so that it quantifies some significant characteristics of the object" [112]. An easier way to think of it, though, is that a feature is an "interesting" part of an image [113].
A feature could be a blob, an edge, a corner, etc. There are different feature detectors like Canny, Sobel, Shi & Tomasi, and Laplacian of Gaussian. Each application needs to select the most proper features and feature detectors.
The task of finding point correspondences between two images of the same scene or object is part of many computer vision applications. Image registration, camera calibration, object recognition, and image retrieval are just a few. [114]
The search for discrete image point correspondences can be divided into three main steps. First, 'interest points' are selected at distinctive locations in the image, such as corners, blobs, and T-junctions. The most valuable property of an interest point detector is its repeatability. The repeatability expresses the reliability of a detector for finding the same physical interest points under different viewing conditions. Next, the neighbourhood of every interest point is represented by a feature vector. This descriptor has to be distinctive and at the same time robust to noise, detection displacements and geometric and photometric deformations. Finally, the descriptor vectors are matched between different images. The matching is based on a distance between the vectors, e.g. the Mahalanobis or Euclidean distance. The dimension of the descriptor has a direct impact on the time this takes, and smaller dimensions are desirable for fast interest point matching. However, lower-dimensional feature vectors are in general less distinctive than their higher-dimensional counterparts. [114] In this work, Speeded-Up Robust Features (SURF) [115] are chosen as the feature extractor. It is a scale-invariant and rotation-invariant feature, which makes it a proper choice for the cases in which the camera pose in different images could change. After extracting the features they need to be described in by numeric values.
The ideal keypoint detector finds salient image regions such that they are repeatably detected despite viewpoint changes; more generally it is robust to all possible image

transformations. Similarly, the ideal keypoint descriptor captures the most important and distinctive information content enclosed in the detected salient regions, such that the same structure can be recognized if encountered. Moreover, besides fulfilling these properties to achieve the desired quality of keypoints, the speed of detection and description needs also to be optimized to fit the time-constraints of the task at hand.

Binary robust invariant scalable keypoints (BRISK) [116] could be used to describe SURF features. BRISK provides a high quality performance as in state-of-the-art algorithms, in spite of a dramatically lower computational cost in compare to many competitors. [116]

Once one has extracted the features and their descriptors from two or more images, the next step is to establish some preliminary feature matches between these images. The approach we take depends partially on the solution introduced in [117]. Also, a brute-force algorithm is employed to find the corresponding features on different images.

3.2.3 RANSAC

The RANSAC (Random Sample Consensus) algorithm [118] is a simple, yet powerful, technique that is commonly applied to the task of estimating the parameters of a model, using data that may be contaminated by outliers. RANSAC estimates a global relation that fits the data, while simultaneously classifying the data into inliers (points consistent with the relation) and outliers (points not consistent with the relation). Due to its ability to tolerate a large fraction of outliers, the algorithm is a popular choice for a variety of robust estimation problems. [119]

RANSAC operates in a hypothesize-and-verify framework: a minimal subset of the input data points is randomly selected and model parameters are estimated from this subset. The model is then evaluated on the entire dataset and its support (the number of data points consistent with the model) is determined. This hypothesize-and-verify loop is repeated until the probability of finding a model with better support than the current best model falls below a predefined threshold (typically 1%-5%). RANSAC

can often find the correct solution even for high levels of contamination; however, the number of samples required to do so increases exponentially, and the associated computational cost is substantial. [119]
RANSAC has many applications in image processing. In this particular work, It has been used several times during the 3D reconstruction process, including:

- Calculation of the fundamental matrix based on the matched feature.
- Finding homography between images to select the best pair among the set of images.
- Finding an object pose from 3D-2D point correspondences.

In each case, different tentative values have been used for RANSAC parameters to obtain the best results.

3.2.4 Epipolar Geometry

The epipolar geometry is the intrinsic projective geometry between two views. It is independent of scene structure, and only depends on the camera's internal parameters and relative pose[120].
Suppose a point X in 3-space is framed in two views, at x in the first, and x' in the second. What is the relation between the corresponding image points x and x'? As shown in figure 3.5a the image points x and x', space point X, and camera centres are coplanar. Denote this plane as π. Clearly, the rays back-projected from x and x' intersect at X, and the rays are coplanar, lying in π. It is this latter property that is of highest significance in searching for a correspondence.[120]
Supposing now that we know only x, we may ask how the corresponding point x' is constrained. The plane π is determined by the baseline and the ray defined by x. From above we know that the ray corresponding to the (unknown) point x' lies in π, hence the point x' lies on the line of intersection l' of π with the second image plane. This line l is the image in the second view of the ray back-projected from x. In terms of a stereo correspondence algorithm the benefit is that the search for the point corresponding to x need not cover the entire image plane but can be restricted to the

line l'.
The geometric entities involved in epipolar geometry are illustrated in figure 3.6. The terminology is [120]:

- The epipole is the point of intersection of the line joining the camera centres (the baseline) with the image plane. Equivalently, the epipole is the image in one view of the camera centre of the other view. It is also the vanishing point of the baseline (translation) direction.
- An epipolar plane is a plane containing the baseline. There is a one-parameter family (a segment) of epipolar planes.
- An epipolar line is the intersection of an epipolar plane with the image plane. All epipolar lines intersect at the epipole. An epipolar plane intersects the left and right image planes in epipolar lines, and defines the correspondence between the lines.

3.2.5 Fundamental Matrix

The fundamental matrix is the algebraic representation of epipolar geometry. In the following we derive the fundamental matrix from the mapping between a point and its epipolar line, and then specify the properties of the matrix[120].
Given a pair of images, it is seen in figure 3.5a that for each point x in one image, there exists a corresponding epipolar line l' in the other image. Any point x' in the second image matching the point x must lie on the epipolar line l' . The epipolar line is the projection onto the second image of the ray from the point x through the camera centre C of the first camera. Thus, there is a map

$$x \mapsto l'$$

from a point in one image to its corresponding epipolar line in the other image. It is the nature of this map that will now be explored. It will turn out that this mapping

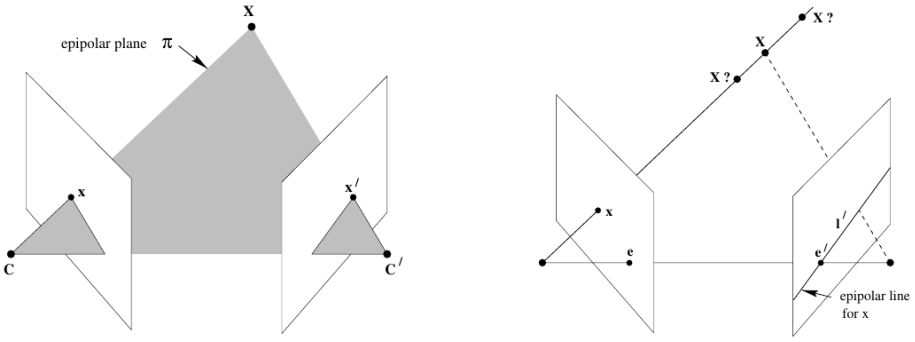


Figure 3.5: Point correspondence geometry. (a) The two cameras are indicated by their centres C and C' and image planes. The camera centres, 3D-space point X, and its images x and x' lie in a common plane π. (b) An image point x back-projects to a ray in 3-space defined by the first camera centre, C, and x. This ray is imaged as a line l' in the second view. The 3-space point X which projects to x must lie on this ray, so the image of X in the second view must lie on l'.

is a (singular) correlation, that is a projective mapping from points to lines, which is represented by a matrix F, the fundamental matrix.[120]

The fundamental matrix is a 3×3 matrix which satisfies the condition that for any pair of corresponding points $x \leftrightarrow x'$ in the two images

$$x'^T F x = 0.$$

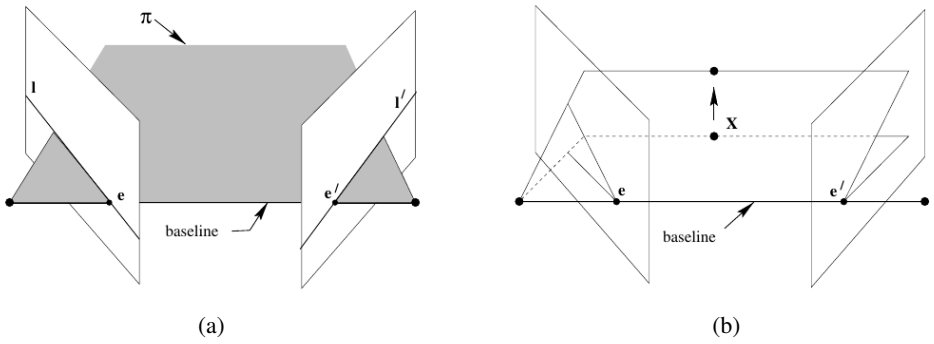
The fundamental matrix is independent of the scene structure. However, it can be computed from correspondences of imaged scene points alone, without requiring knowledge of the camera's internal parameters or relative pose[120].

The epipolar geometry between two views is essentially the geometry of the intersection of the image planes with the pencil of planes having the baseline as axis (the baseline is the line joining the camera centres). This geometry is usually motivated by considering the search for corresponding points in stereo matching.

The fundamental matrix is of rank 2. If we have enough matches like (x,x') from 2 images which satisfy the equation $x'^T F x = 0$, then it is possible to estimate the

matrix.[120]

So, the problem of finding the fundamental matrix is a linear estimation problem, which can be easily solved by linear least-square estimation methods. However, automatic methods for finding point correspondence make mistakes very often in practice, thus introducing outliers in the point correspondences for the fundamental matrix estimation. It is well known that least-square estimation is very sensitive to outliers, therefore several robust estimation techniques have been applied to overcome the outlier problem. Among them, M-estimator and RANSAC technique have been widely used in many computer vision problems [121].



(a) (b)

Figure 3.6: Epipolar geometry. (a) The camera baseline intersects each image plane at the epipoles e and e' . Any plane π containing the baseline is an epipolar plane, and intersects the image planes in corresponding epipolar lines l and l' . (b) As the position of the 3D point X varies, the epipolar planes 'rotate' about the baseline. This family of planes is known as an epipolar pencil. All epipolar lines intersect at the epipole.

3.2.6 Triangulation

Let us suppose that a point X in R^3 is visible in two images. The two camera matrices P and P' corresponding to the two images are supposed known. Let x and x' be

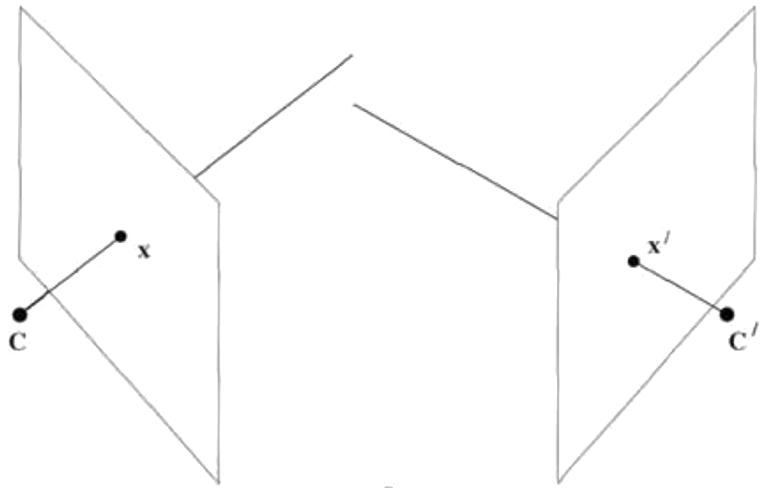


Figure 3.7: Rays back-projected from imperfectly measured points X in R^3 are skew in R^3 in general. [122]

projections of the point X in the two images. From these data, the two rays in space corresponding to the two image points may easily be computed. The triangulation problem is to find the intersection of the two lines in space. At first sight this is a trivial problem, since intersecting two lines in space does not present significant difficulties. Unfortunately, in the presence of noise, these rays cannot be guaranteed to cross. [123]

Therefore, that there will not be a point X which exactly satisfies $x = PX$, $x' = P'X$; and that the image points do not satisfy the epipolar constraint $x'^T Fx = 0$. These statements are equivalent since the two rays corresponding to a matching pair of points $x \leftrightarrow x'$ will meet in space if and only if the points satisfy the epipolar constraint. See figure 3.7.

So, one needs to find the best solution under some assumed noise model. However, regardless of the selected solution, the resulting reconstructed 3D point, $\hat{X}$, would not exactly match X. As it is shown in figure 3.8, the corresponding image points $\hat{x}$ and $\hat{x}'$ satisfy the epipolar constraint. Then, an important measure to evaluate triangulation solution could be reprojection error:

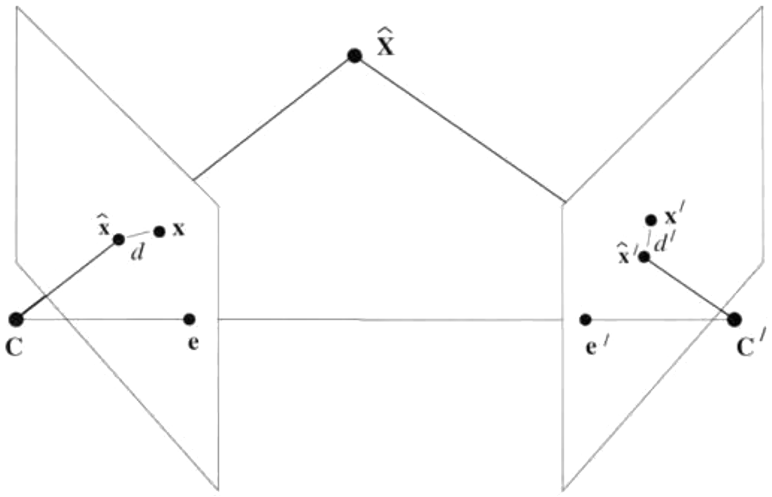
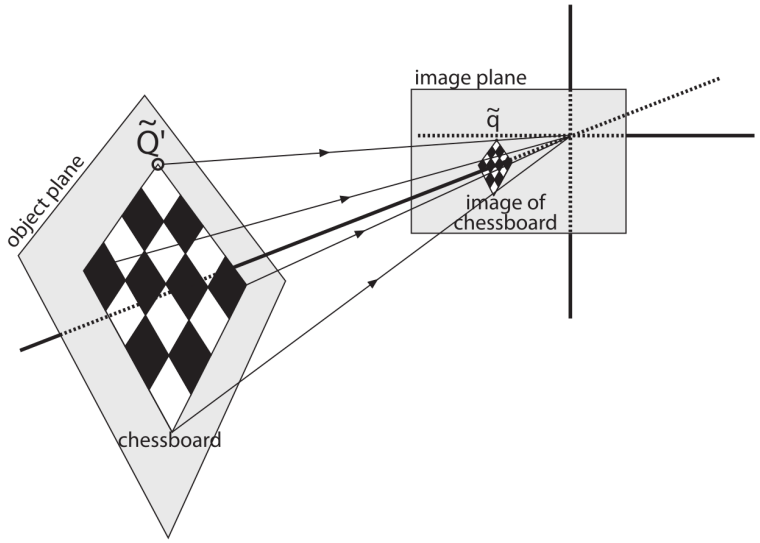


Figure 3.8: The estimated R^3 point X projects to the two images at $\hat{x}$ and $\hat{x}'$. -The corresponding image points $\hat{x}$ and $\hat{x}'$ satisfy the epipolar constraint, unlike the measured points x and x'. [122]

$$d^2 + d'^2$$

To formulate the problem let us assume P_i to be the camera matrix of the ith image and x_1 and x_2 the 2D coordinates of matching points between the images, one could write $x_1 = P_1 X$ and $x_2 = P_2 X$, where X denotes the 3D coordinate of the projected point. Besides, for every point which is not too close to the camera or at infinity, one could say $X = (x, y, z, 1)t$ in witch t is the scaling factor. This creates an inhomogeneous linear system of the form $AX = B$. The least-squares solution of this equation could be found by using the singular value decomposition. However, this method is inaccurate. Hartley and Sturm [123] suggested an iterative method to reduce inaccuracy. They proposed to recalculate X after updating A and B based on the obtained result. By repeating this procedure for several times the accuracy increases significantly.



3.2.7 Homography

'Homography' is the mathematical term for mapping points on one surface to points on another (Figure 3.2.6). Even though homography can be the relation between a planar object and its image, in the context of computer vision, homography usually refers to mapping between points on two image planes that correspond to the same location on a planar object in the real world. It can be shown that such a mapping is representable by a single 3-by-3 orthogonal matrix [110].

In our approach, to further improve our matching quality, we can perform outlier filtration using the random sample consensus (RANSAC) method. As we are working with an image (a planar object) and we expect it to be rigid, it is ok to find the homography transformation between feature points on the pattern image (e.g. object itself) and feature points on the query image (e.g. object's image). Homography transformations will bring points from a pattern to the query image coordinate system. To find this transformation, we use the *cv::findHomography* function of OpenCV. It uses RANSAC to find the best homography matrix by probing subsets of input points. As a side effect, this function marks each correspondence as either inlier or outlier,

depending on the reprojection error for the calculated homography matrix [124]. It can also be a measure of the quality of a pair of images among several images of the same object. Choosing the right pair in multi-view vision as the primary pairs is very important decision, because all the other poses and calculation will be in alignment with the first pair.

3.2.8 Bundle Adjustment

Bundle adjustment is the problem of refining a visual reconstruction to produce jointly optimal 3D structure and viewing parameter (camera pose and/or calibration) estimates. Optimal means that the parameter estimates are found by minimizing some cost function that quantifies the model fitting error, and jointly that the solution is simultaneously optimal with respect to both structure and camera variations. The name refers to the 'bundles' of light rays leaving each 3D feature and converging on each camera centre, which are 'adjusted' optimally with respect to both feature and camera positions. Equivalently, unlike independent model methods, which merge partial reconstructions without updating their internal structure, all of the structure and camera parameters are adjusted together 'in one bundle'[125].
Bundle adjustment is really just a large sparse geometric parameter estimation problem, the parameters being the combined 3D feature coordinates, camera poses and calibrations. Almost everything that we will describe can be applied to many similar estimation problems in vision, photogrammetry, industrial metrology, surveying and geodesy[125]. Bundle adjustment should generally be used as a final step of any reconstruction algorithm.
Consider a situation in which a set of 3D points X_j is viewed by a set of cameras with matrices P^i. Denote by x^i_j the coordinates of the j-th point as seen by the i-th camera. We wish to solve the following reconstruction problem: given the set of image coordinates x^i_j find the set of camera matrices P^i, and the points X_j such that $P^i X_j = x^i_j$. Without further restriction on the P^i or X_j, such a reconstruction is a projective reconstruction, because the points X_j may differ by an arbitrary 3D projective transformation from the true reconstruction.[122]

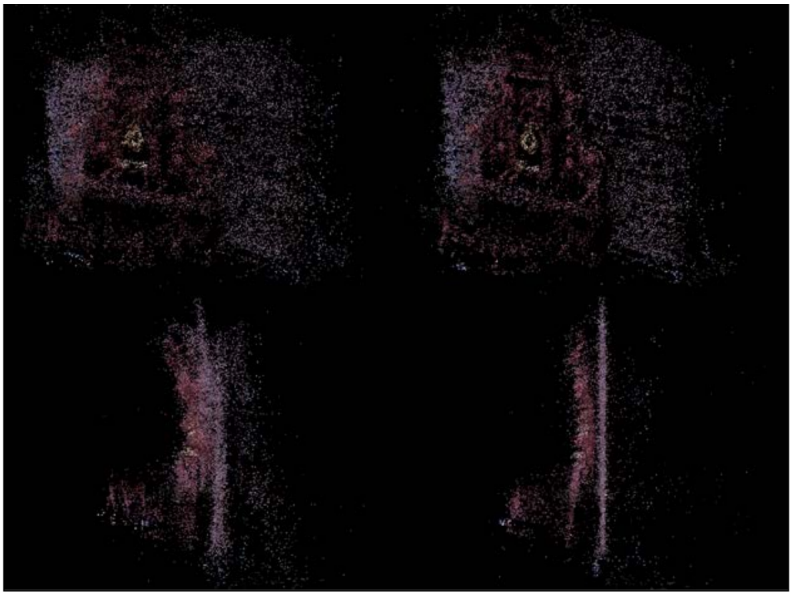


Figure 3.9: The two images on the left are the points of the point cloud before adjustment, from two perspectives, and the images on the right show the optimized cloud. [124]

If the image measurements are noisy then the equations $P^i X_j = x^i_j$ will not be satisfied exactly. In this case we seek the Maximum Likelihood (ML) solution assuming that the measurement noise is Gaussian: we wish to estimate projection matrices $\hat{P}^i$ and 3D points $\hat{X}_j$ which project exactly to image points $\hat{x}^i_j$ as $\hat{x}^i_j = \hat{P}^i \hat{X}_j$, and also minimize the image distance between the reprojected point and the detected (measured) image points $\hat{x}^i_j$ for every view in which the 3D point appears, i.e.

$$\min_{\hat{P}^i \hat{X}^j} \sum_{ij} d\left(\hat{P}^i \hat{X}^j, x^i_j\right)^2$$

where $d(x,y)$ is the geometric image distance between the homogeneous points x and y.[122]
Figure 3.9 shows an example of applying bundle adjustment to improve the result of a 3D reconstruction algorithm.

3.3 Size Reference

Food volume estimation is the most direct approach to automating the computation of calories or nutrients of food intake. Volume estimation from images can be obtained through different procedures, but only up to a scale factor. Therefore, in many studies, an object of known size is used as reference for volume estimation. In [126], for instance, the user puts his/her finger besides the dish. In [92], a specific pattern of known size printed on a card is used as reference. Many studies have followed the same idea using a checkerboard as reference [36, 106].

The simplicity of the checkerboard pattern and the existence of effective algorithms to detect it are some of the reasons for choosing it over other options. Nevertheless, off-the-shelf checkerboard detection algorithms are usually designed to be means for camera calibration or pose-detection processes. They are usually tuned for specific situations like flat checkerboards or a big checkerboard that is often the only object in the image, etc. Thus, for applications which do not satisfy these requirements, different algorithms, or modified versions of available ones, are needed. Checkerboards which are used as size references usually consist of few squares and occupy a relatively small portion of the image. This situation is difficult for 'standard' checkerboard detectors to be as effective as within the settings to which they have been originally tuned. Figure 3.10 shows three examples in which OpenCV and Matlab checkerboard detection algorithms fail. The OpenCV algorithm is available through the *findChessboardCorners()* function, and the Matlab algorithm is available through the *detectCheckerboardPoints* function (hereafter, they will be referred to, as *basic method 1* and *basic method 2*), respectively.

The method developed in this thesis, first, locates the checkerboard in the image using a stochastic model-based approach. Later, the detected region is processed by a customized corner detection algorithm to obtain the exact coordinates of the checkerboard corners. It is shown that this idea outperforms the basic algorithms. Moreover, the stochastic nature of the proposed method gives it an important advantage. In the case of food intake monitoring, missing the checkerboard in an image would cost the user the trouble of providing another image of the food, and even so there would not

be any guarantee that, under the same condition, the algorithm would work with the new image. Instead, the stochastic nature of the proposed method makes it possible to run a new attempt on the same image, when the previous run fails.

3.3.1 Method

The method consists of two main steps: locating the checkerboard in the image and then detecting the checkerboard corners in the located area.

Locating Objects In An Image

The procedure actually estimates the pose of an object based on a 3D model and can be utilized with any projection system and any general object model. Sets of 3D points, which belong to the different parts of the object to be detected and describe its shape, are sampled to build the model. Then, once the object pose has been hypothesized, the points are projected onto the image plane according to a transformation which maps points in the camera reference frame onto the image, and matched with

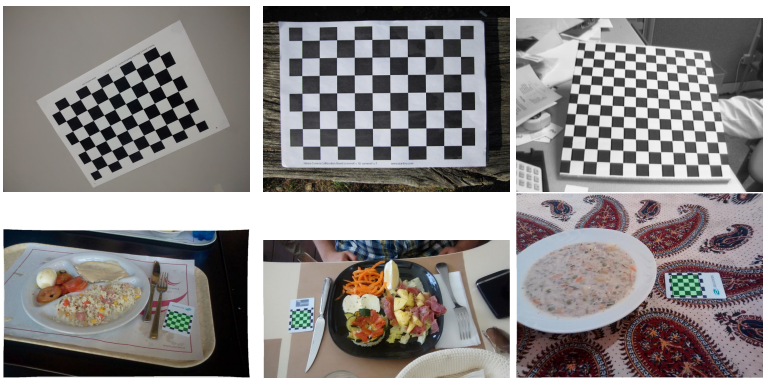
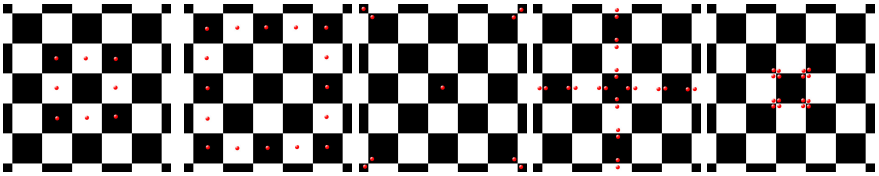


Figure 3.10: First row: typical images used for calibration. Second row: Examples of images in which checkerboard is used as size reference for food volume estimation, on which both OpenCV and Matlab functions for detecting checkerboards fail, while they are correctly located by the model-based algorithm (green areas)



(a) (b) (c) (d) (e)

Figure 3.11: Key points are evaluated in five steps to compute the similarity degree between the projected model and the image.

the actual image content. The likelihood of the detection is evaluated using a similarity measure based on pixel intensity.
This generally-applicable object detection algorithm includes the following steps:

1. Consider a set of key points, of known coordinates with respect to a general model of the object to detect.

2. Guess the pose of the object, and then translate and rotate the point set to the hypothesized position within the camera's field of view and project them onto the image.

3. Verify that the characteristics of the points in the set match those which could be observed in the image region where they would be projected to assess the presence of the object.

In this case, the model of the object (a checkerboard) consists of 73 key points (Figure 3.11), corresponding to the center, edges and corners of each square.

After locating the checkerboard in the image with a degree of confidence represented by the threshold which the similarity function must reach, the region identified in the matching step is cropped and basic methods to detect checkerboard corners are applied to the resulting image. The search method is based on a popular optimization algorithm known as Differential Evolution (DE) [127].

Differential Evolution (DE)

Problems which involve global optimization over continuous spaces are ubiquitous in real life. Any design problem can be virtually considered as a (multi-objective constrained) optimization problem. In general, the task is to optimize certain properties of a system by pertinently choosing the system's parameters which, for convenience, are usually represented as a vector. The standard approach to an optimization problem begins by designing an objective function that can model the problem's objectives while incorporating any constraints to be met[127].
Differential Evolution (DE) is a relatively simple evolutionary optimization method. It iteratively tries to improve a candidate solution with respect to a given measure of quality [127]. In DE, first, a random set of solutions (population) is generated. Then, inside a loop, the solutions are evaluated by the fitness function and new solutions are generated for the next iteration. New individuals that will be part of the next generation are created by combining individuals that are already members of the current population. Every individual acts as a parent vector and, for each of them, a donor vector is created. In the basic version of DE, the donor vector for the ith parent (X_i) is generated by combining three random and distinct individuals X_{r1}, X_{r2} and X_{r3}. The donor vector V_i is calculated by what is called mutation of difference vectors as follows:

$$V_i = X_{r1} + F(X_{r2} - X_{r3})$$

where F (scale factor) is a parameter that strongly influences DE's performances and typically lies in the interval $[0.4, 1]$. Recently, several mutation strategies have been applied to DE, experimenting with different base vectors and different numbers of vectors for perturbations. For example, the original method explained above is called $DE/rand/1$, which means that the first element of the donor vector equation X_{r1} is randomly chosen and only one difference vector (in this case $X_{r2}X_{r3}$) is added. After mutation, every parent-donor pair generates a child (U_i), called trial vector, by means

of a crossover operation.

$$U_{i,j} = \begin{cases} V_{i,j} & \text{if } (rand_{i,j} \leq C_r \text{ or } j = j_{rand}) \\ X_{i,j} & \text{otherwise} \end{cases}$$

As described in the above equation, the jth component/dimension of the ith donor vector is obtained by means of uniform (binomial) crossover, where $rand_{i,j}$ is a random number coming from a uniform distribution in the range [0, 1], C_r is the crossover rate, and j_{rand} is a randomly selected dimension. The newly generated individual U_i is evaluated by comparing its fitness to its parent's. The best survives and will be part of the next generation [128].

There are several reasons for researchers to consider DE as an attractive optimization tool: DE is simple and straightforward to implement, and exhibits much better performance in comparison with several other methods on problems onto which many real-world problems can be mapped. The number of control parameters in DE is very small, and its space complexity is low [129].

One of the main characteristics of DE is its stochastic nature. If it fails to find the checkerboard in a run, it is always possible that it succeeds in another run. A restart strategy in case of failure is allowed by both DE' simplicity and intrinsic parallel nature, which makes it possible to obtain very efficient implementation of the algorithm.This is something that is missing in the basic methods.

Fitness Function

Let $f : \Re^n \rightarrow \Re$ be the fitness function (cost function) which must be maximized (minimized). The function takes a candidate solution in the form of a vector and produces a real number as output. The goal of global optimization algorithms such as DE is to find a solution s for which $f(s) >= f(c)$ for all c in the search-space, which would mean s is the global maximum.

In this work the input argument of the fitness function is a vector of six parameters:

$$V^T = [x, y, z, \alpha, \beta, \gamma]$$

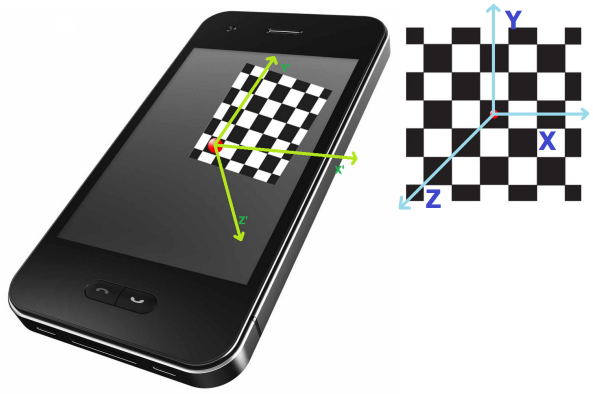


Figure 3.12: Camera coordinate system (green) and world coordinate system (blue). The input vector of the fitness functions represents the transformation (translation and rotation) which matches the world coordinate system to the camera coordinate system.

These parameters represent the pose of the object with respect to the camera. The first three parameters represent translation in the 3D coordinate system and the other three represent rotation around the coordinate system axes. These parameters present the transformation which matches the blue coordinate system in figure 3.12 to the green one.

The fitness function calculates the degree of similarity between the re-projected model and the image. To do so, 73 key points are used to describe the checkerboard model. The similarity degree is calculated in 5 steps. In each step a subset of key points is checked and a corresponding similarity term is added to the fitness. The solution passes to the next step if at least a certain degree of similarity is found, otherwise it rejects the hypothesis returning a bad fitness value. Figure 3.11 presents the key points checked in every step of the fitness function and algorithm 3 shows the pseudo-code of the function. A perfect match will be given a score of 54, however any score higher than 45 can be considered an acceptable match in this work.

Algorithm 3 Fitness Function

```
function FITNESSFUNCTION(PoseVector)

    Calc Rotation & Translation matrices from PoseVector

    Score ← Score + FirstLevelCenterCheck()
    if Score > 6 then
        Score ← Score + SecondLevelCenterCheck()
    end if
    if Score > 20 & the center is black then
        Score ← Score + PoseCheck()
    end if
    if Score > 23 then
        Score ← Score + EdgesCheck()
        Score ← Score + verticesCheck()
    end if
return Score

end function
```

Algorithm 4 Corner detection algorithm

function FINDCHECKERBOARDCORNERS

$Image \leftarrow GetCroppedRGBImage()$
$Image \leftarrow RGB2SinglePrecision(Image)$
▷ Calculate second derivatives at zero and 45 degrees
$D0 \leftarrow CalcSecondDerivative(Image, 0)$
$D45 \leftarrow CalcSecondDerivative(Image, 45)$
▷ Find the pixels with higer values
$Corners0 \leftarrow FindCorners(D0)$
$Corners45 \leftarrow FindCorners(D45)$
▷ Find 3 closest corners to the image center
$CentralCorners0 \leftarrow Find3CentralCorners(Corners0)$
$CentralCorners45 \leftarrow Find3CentralCorners(Corners45)$
▷ Expand the candidate checkerboards based on the centeral corners
$Candidates0 \leftarrow ExpandCentralCorners(CentralCorners0)$
$Candidates45 \leftarrow ExpandCentralCorners(CentralCorners45)$
▷ Choose the best candidate
$BestCandidate \leftarrow ScoreCandidates(Candidates0, Candidates45)$
return *BestCandidate*

end function

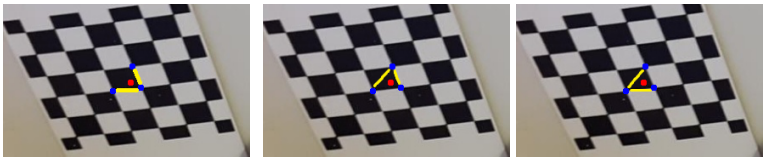
Corners detection

Since the approximate location of the checkerboard is detected during the first phase, it is possible to design a customized algorithm to detect the checkerboard corners on the cropped image. Algorithm 4 describes the process flow of the corner detection method. The method is developed based on the hypothesis that the checkerboard is in the center of the cropped image and covers most of the image.

The process starts by calculating agray-scale image and then a single-precision image of the RGB input. Then, the second derivative of the single precision image is calculated in two directions: zero degree and 45 degrees. This allows the algorithm to detect the corners of the checkerboards with various poses.

In both the derivative images, the pixels with higher values are identified as corners. This set contains checkerboard corners as well as other generic corners inside the cropped image. Then, one of the consequences of the checkerboard being located in the center of the image is utilized: three corners which are the closest to the image center are belong to the central square of the checkerboard (one can not be sure about the fourth one, due to approximation in locating of the checkerboard). Afterwards, the other corners of the checkerboard will be detected by gradually growing it from the central square.

In this step, there are 3 candidates in each derivative image. Figure 3.13 demonstrates 3 different options of checkerboard expansion based of the three selected corners. Obviously there is only one correct choice. Since there are two derivative images, that accounts 6 candidates in total. To find the true checkerboard pose, each candidate



(a) (b) (c)

Figure 3.13: Center of image (red) and 3 closest corners on a located checkerboard. Only one combination (a) is correct whose expansion reproduces the checkerboard.

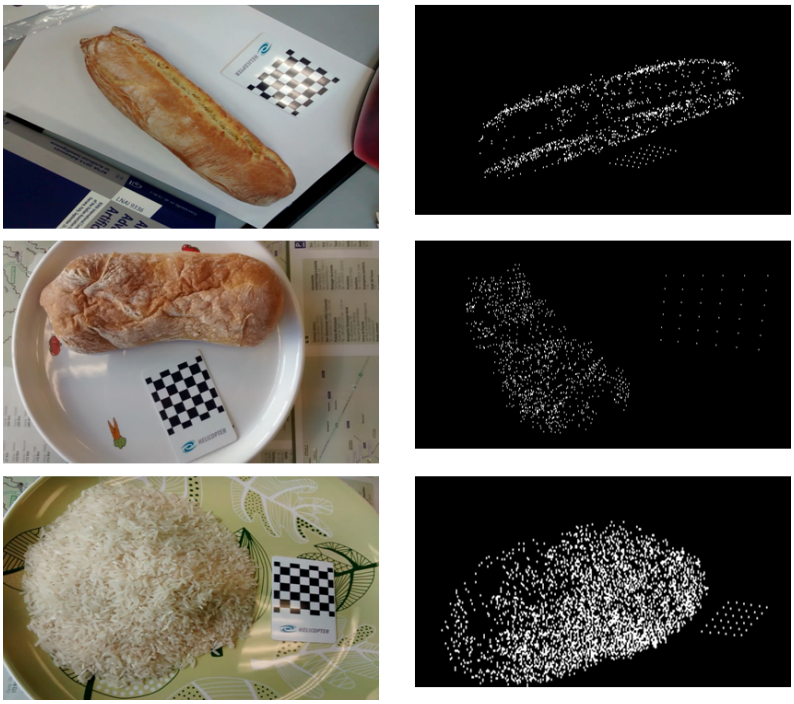


Figure 3.14: Three examples of reconstructed models by the proposed system.

will be scored based on the compliance with the known checkerboard structure (in this case a 5×5 one). The candidate with the highest score is selected as the true checkerboard.

3.3.2 Ground Reference

In this method, the checkerboard can be assumed as the reference of the ground surface. The plane with the least orthogonal distance from the checkerboards corners is the ground surface. Considering vector n to be the normal to the ground plane and $P = [p_1, ..., p_n]$ a $3 \times n$ matrix in which columns are the detected corners' coordinates, the plane can be found by calculating the Singular Value Decomposition (SVD) of $A = [p_1 - c, p_2 - c, ..., p_n - c]$ where c is a point belonging the plane [130]. We set c

equal to the centroid of P and use the left singular vector of the SVD as n, the normal vector to the best-fitting plane [130].
This use of the checkerboard is valuable during 3D-modeling to extract more information about the lower parts of the object which could be invisible in the video taken from above and also to remove noise.

3.4 Image-Based Modeling

In this step, a 3D model of the food item in the form of a point cloud is generated based on a Structure from Motion (SfM) algorithm. This thesis followed the same approach suggested in [131] and [124], properly adapted to this work. In fact, the base approaches have been designed to build a model based on many (up to a few thousands) images. This algorithm is customized to use a lower number of images (six, in this case) and address processing time concerns. Algorithm 5 demonstrates the pseudo-code of the method and Figure 3.14 shows three examples of the reconstructed models.

3.4.1 Method

Algorithm 5 describes the pseudo-code of the method. First, Speeded Up Robust Features (SURF) of the images are detected and their Binary Robust Invariant Scalable Keypoints (BRISK) description are used to find matches between image pairs. A pair should have a minimum of 30 matches to be used in the rest of the process. There are 15 possible pairs in total for six images, which is quite a high number which could increase the total processing time significantly. Therefore, the matches are calculated only for a configurable number of neighboring images and not for all of possible pairs. In the final configuration each image is paired only with two neighbor images. Then, the fundamental matrix[122] is calculated for all the image pairs using an 8-point Random Sample Consensus (RANSAC). The RANSAC outliers threshold is set to 0.6% of the maximum image dimension and the level of confidence is set to

99%. The result of RANSAC is also used to remove outliers from the matches.
In the next step, the best image pair is chosen to start a 3D reconstruction of the model. Homography transformation is used to make the choice. Homography is the mathematical term for mapping points on one surface to points on another. The transformation between images of each pair is calculated using the RANSAC with the outliers threshold set to 0.4% of the maximum image dimension. This method, besides finding the transformation, separates the matching features in two groups of inliers and outliers. The pair with highest inlier rate is chosen to start the reconstruction and for being the baseline of the modeling process. Moreover, the selected pair of images is evaluated to have high-quality matches by checking the validity of its essential matrix[122]. If the pair fails in the test, the next best pair is chosen as the base of 3D reconstruction.

Then, the 2D coordinates of the point matches between the selected images are used to calculate the corresponding 3D coordinate. For this part, the Hartley and Sturm algorithm which is explained in section 3.2.6 is used.

Next, other images join one by one the triangulation process. However, for the reason explained above, the 3D coordinates can be estimated only up to a scale factor t. As a consequence of this, 3D points of every image pair might have a different scale factor and it may be therefore impossible to exploit them together. The other issue is the order by which the cameras(images) join the process. If a camera with lower quality of image joins to the process before the others, the final model can be affected negatively.
In order to tackle these issues, first the best candidate among the remaining cameras is chosen. The number of features, which have a reconstructed match among the processed images, is calculated for each camera and the camera with the highest number of such features joins the process. Then, the position of the new camera is estimated based on the previously constructed point. It is a classic Perspective-n-Point problem (PnP) which determines the position and orientation of a camera given its intrinsic parameters and a set of n correspondences between 3D points and their 2D projec-

tions. The *solvePnPRansac* function provided by OpenCV is used to perform this step. The function finds the pose that minimizes the reprojection error. Also, the use of RANSAC makes the function robust to outliers. This approach ensures that the 3D points added by the new images will have the same scaling factor as the previous ones.

Algorithm 5 3D modeling algorithm.

function MODEL(OriginalImages[], SegmentedImages[], CameraMatrix)

▷ Prepare Matches

$Matches[] \leftarrow$ **Extract-SURF-BRISK-Matches**$(OriginalImages[])$

$Matches[] \leftarrow$ **PruneOutliers**$(Matches[])$

▷ Start Modeling

$BaseLine \leftarrow$ **FindTheBestPair**$(Matches[])$

$PointCloud \leftarrow$ **Triangulate**$(BaseLine, Matches[], CameraMatrix)$

while there is more un-processed image **do**

$NextImage \leftarrow$ **ChooseNextImage**$(PointCloud, Matches[])$

$NewMatrix \leftarrow$ **FindPoseAndCameraMatrix**$(PointCloud, Matches[], CameraMatrix)$

$PointCloud \leftarrow PointCloud +$ **Triangulate**$(BaseLine, Matches[], NewMatrix)$

end while

▷ Find Checkerboard

$Corners \leftarrow$ **FindCheckerboardCorners**$(OriginalImages[], CameraMatrix)$

$Scale \leftarrow$ **EstimateSizeScale**$(Corners)$

$SurfacePlane \leftarrow$ **EstimateSurfacePlane**$(Corners)$

▷ Refine PointCloud

$PointCloud \leftarrow$ **EliminateNoise**$(PointCloud, SurfacePlane)$

return PointCloud, Scale

end function

Bundle adjustment is also applied on the triangulation output in order to refine the results. It is an optimization step where both the position of the 3D points and

the position of cameras are optimized to minimize the reprojection errors (see section 3.2.8).
Until this step, the entire image is used for all the processes. All the visual information of the image is used to achieve more accurate results, even though some parts of the image are not included in the region of interest. However, only the food items of the image are interesting for the portion estimation process. So, in this step, all the non-object points are cut from the result based on the segmentation stage outputs. Usually, the segmented images are not 100% accurate and there are some errors. The fact that a 3D point should appear in at least 2 images compensates for some segmentation errors which might occur in a single image.
Finally, outliers are removed from the point cloud. Since the ground surface has been already determined based on the checkerboard location, it is possible to apply restrictions to the maximum acceptable height of the food item (we set such limit to 20 *cm*). The point cloud usually also includes isolated outliers that are relevantly far from the other points. To deal with this problem, we remove from the point cloud the points whose distance from the closest neighbor is larger than a threshold. In practice, for each point four types of distances are defined: the Euclidean distance, as well as the distances along the three coordinate axes. Acceptable points should be no farther from their nearest neighbor than one standard deviation for each of the measures taken into consideration.
Moreover, if a set of isolated points, including less than five points, is too far from the neighbors, it will be removed as well.
The remaining point cloud provides a 3D model of the food and can be used to estimate its volume.

3.5 System Evaluation

First, the checkerboard detection algorithm is evaluated separately. Then, the whole system to estimate the object's volume is evaluated. The tests are performed on a PC with Intel®Core™$i7$ 2.8GHz CUP running the $64-bit$ version of Windows 7 professional.

Table 3.1: Results of the DE-based checkerboard locating algorithm.

	Images No.	Success	First try Success	DE tries
Motorola MotoG	179	176	143	1.34
Samsung Galaxy Note 1	19	19	11	1.3
Samsung Galaxy S3	130	129	123	1.2
Samsung Galaxy S3 scaled (0.5)	130	127	125	1.13
Total	458	451	402	1.24

3.5.1 Checkerboard Detection

The algorithm is tested over four sets of images, for a total of 458 images in each of which a 5×5 checkerboard, printed on a plastic card is visible. The images of each set are taken by a different mobile device and in different environments. In one case (Samsung Galaxy S3), since the high-resolution default images lead to worse results using basic corner detection algorithms, a scaled version of the images is tested as well. The fourth set therefore contains the images in the third set scaled by a factor of 0.5.

In this work an implementation of DE with binomial crossover is used to locate the checkerboard. DE is iterated up to 1000 times for every image. Moreover, if the algorithm fails to locate the checkerboard in a first attempt, it is repeated from scratch with a new population. For every image, DE is allowed to run up to four times.

Table 3.1 demonstrates the results of the algorithm in locating checkerboards. In more than 98% of the cases the checkerboard is correctly located. In 89% of the cases it is found within the first try of DE; on the average, DE is repeated 1.24 times for each image.

Table 3.2 shows the result of the corner detection algorithms. Five different methods are evaluated. Besides the basic methods and the proposed method, the results of combining the locating method and the basic methods are reported. In the latter case, before applying the basic methods on the original image, the region selected by the locating algorithm is fed to the algorithm. If it fails, the original image is used.

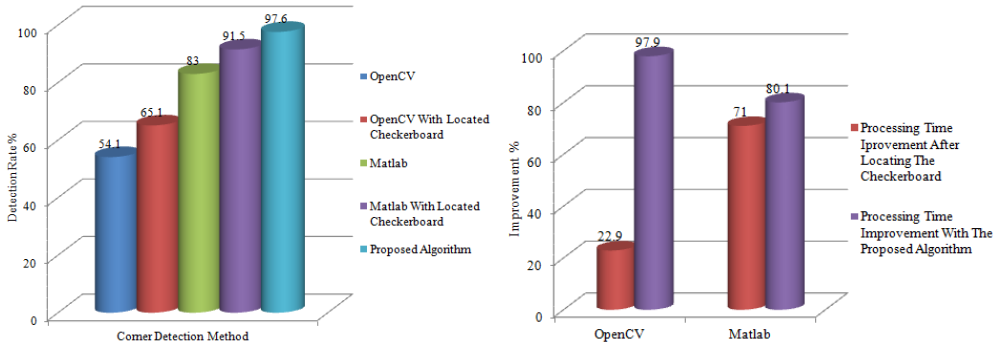
Table 3.2: Results of the checkerboard corner detection algorithms.

		Motorola MotoG	Samsung Galaxy Note 1	Samsung Galaxy S3	Samsung Galaxy S3 scaled (0.5)	Total
Total number of images		179	19	130	130	458
Basic method 1	Detection	118	12	24	94	248
	Time (ms)	3368393	179775	16276198	909017	20733383
Basic method 1 with located checkerboard	Detection	133	14	51	100	298
	Time (ms)	2618700	133823	12512673	710783	15975979
Basic method 2	Detection	159	12	88	121	380
	Time (ms)	1267700	42853	626590	195250	2132393
Basic method 2 with located checkerboard	Detection	171	18	102	128	419
	Time (ms)	269817	21682	222827	104179	618505
Proposed method	Detection	173	19	128	127	447
	Time (ms)	200976	21324	114170	88669	425139

As shown in table 3.2, there is an obvious improvement in both speed and performance of the basic algorithms when they use the pre-processed images (cropped checkerboards found by DE). Locating the checkerboard reduces the total processing time (including pre-processing) of the basic methods respectively by 23% and 71%, whereas the number of correctly detected checkerboards increases by 20% and 10%. Yet, the best results come with the proposed algorithm, both regarding detection rate and processing time. Figure 3.15 shows a summary of the results.
Moreover, one can notice that the proposed method is much less sensitive to image resolution, whereas basic methods are dramatically affected by high resolution images, which are common in new smartphones. This advantage is an outcome of choosing a simple model for both locating and corner-detection phases.

3.5.2 Volume Estimation

The method is evaluated using two common types of food items: food items with a solid shape and food with non-solid shapes. In both procedures, it is assumed that the shapes are more or less convex, in order to keep the volume estimation simple. So, the volume of the generated models are calculated using the Qhull tools [132].
For the first group, six types of bread are chosen as the test cases. Additionally, in order to have a precise ground truth, an industrial 3D laser scanner, *Sick LMS400*, has



(a) Detection Rate (b) Processing Time

Figure 3.15: Results summary: (a) Improvement of the detection rate after applying the locating approach or using the proposed algorithm. (b) Processing time improvement in compare to the basic methods.

Table 3.3: Error rate in the reconstructed models' volume estimations for solid shapes.

Row	Bread	Error Rate(%)
1		7.2%
2		4.9%
3		9.5%
4		1.7%
5		19.1%
6		7.2%
	average	8.27%
	average processing time	21.5*s*

been used to acquire a 3D model of the objects. The device has a spatial resolution of $1mm$ and the scanned model is manually segmented to achieve a more accurate ground truth.

Table 3.3 shows the obtained result on the first test set. The average error rate of the whole process (including segmentation, size reference detection, 3D model reconstruction) is 8.27%. As shown in table 3.3, the error rate of bread number 5 is high due to its poor visual context. It has a quite smooth surface while showing slow variations of shade and color. Also, bread number 6 has a rather high error despite its rich context. This is a consequence of its small dimensions which increases the relative weight of noise.

The average processing time of the first test set, including the whole process from segmentation to volume estimation is around $21s$.

In the second part, pasta and rice are tested as food with non-solid shape. Two type of evaluations are made on this set. First, in order to examine the stability of the method, the same food with the same weight is tested several times. For example, for each type of pasta, the same amount of raw pasta is cooked and tested on two different dishes. Besides, for each dish, the test is repeated with a different layouts as well (in total, four times for the same amount of pasta). As a second assessment, the correlation between food amount and its volume is evaluated.To do so, the same type of pasta with twice the amount as the previous dish is cooked and the correlation between its volume and the latter's volume is checked.

Table 3.4 shows the obtained results on the non-solid food test set. The average error rate is 6.7%. The images included in this test set provide stronger features, which produce more robust results, due to their rougher surface which causes strong variations of light intensity. However, a sparser scattering of the pasta pieces over the dish can lead to a less linear correlation between the volumes estimated for different amounts of the same kind of food. Within this group, the best results were obtained on rice due to the larger number of features it provides and to the smaller gaps between rice seeds with respect to larger-sized pasta. On the contrary, the Chifferi pasta, with its large and curvy pieces is obtained the worst result because it provides fewer visual features and has larger gaps between the pieces. Diagrams 3.16 and 3.17 show the

Table 3.4: Error rate in the reconstructed models' volume estimations for non-solid shapes.

Row	Food	Error Rate(%)
1	Rice 	5.1%
2	Pennette 	6.8%
3	Fusilli 	7.1%
4	Chifferi 	8%
	average	6.7%
	average processing time	24.01s

results of the volume measurement for every test case. As shown, repeating the tests on a same amount of the pasta on different dishes of different shapes, we obtained reasonably stable results.

Although the non-solid food types provided more robust results thanks to their higher number of features, they needed a larger computation effort in the modeling process, which increased the total processing time. The average processing time for the first test set, for the whole process from segmentation to volume estimation, was around 21s, while the average time for the second test set was 24.01s. Globally, the average processing time was 23.52s. Table 3.5 reports the average execution time for each processing step.

3.6 Discussion

A system to calculate food intake quantity using image-based modeling is introduced. The approach includes a semi-automatic segmentation method which was designed

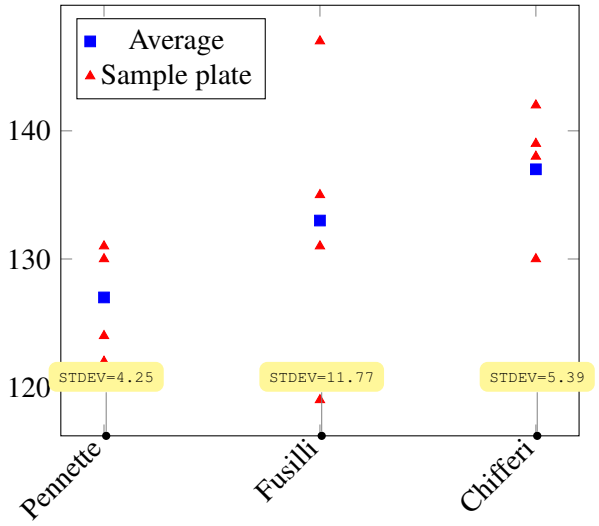


Figure 3.16: Calculated volume for 75 g of different type of pasta. For each one, the calculation is repeated 4 times on two different dishes.

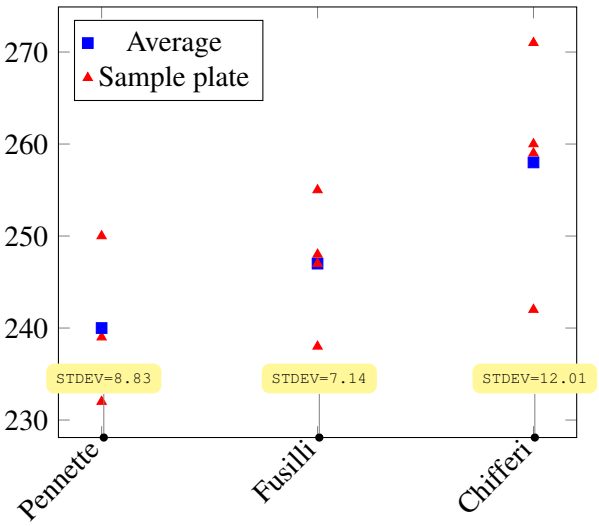


Figure 3.17: Calculated volume for 150 g of different type of pasta. For each one, the calculation is repeated 4 times on two different dishes.

Table 3.5: Processing time of the whole process

Processing Step	Time	Ratio
Segmentation	4.52	19.2 %
Size Reference	2.90	12.3 %
Model Reconstruction	16.1	68.5 %

to run on smartphones and to be interactive and user-friendly.
Instead of taking a few images as usually happens, the user is asked to perform the easier task of taking a short video of the food from which the system extracts the set of six frames that guarantees the best visual coverage of the food. The method generates a 3D model of the food items and uses the model to estimate the food's volume. The image-based modeling approach is customized to limit the processing cost.
A small checkerboard, printed on a PVC card, is used as size reference. However, detecting a small checkerboard, which is expected to occupy only a small region on the image, could be challenging for off-the-shelf algorithms. Therefore, a stochastic model-based algorithm is designed to detect small checkerboard's corners in the images. The proposed algorithm offers very satisfying performance in detecting of small checkerboard corners and can be a valuable asset in automatic size-estimation applications and, in particular, for food amount estimation. It first locates the checkerboard, then a customized corner detection algorithm finds the exact position of the corners. The algorithm outperforms basic methods even after improving them by preprocessing phase. The checkerboard is also used as ground surface reference which is a big help in noise removing step of the modeling process.
The method achieves about 92% accuracy in the test cases. Also, the processing time of the whole method is about 23s on a regular PC. Although the accuracy of the method drops down with low-context items or small food items, the results suggest that the approach still could be a proper choice for food intake monitoring.

Chapter 4

Summary

Citazione (l1)
Citazione (l2)

– Autore Citazione

In this thesis, a new method for automatic diet monitoring is proposed. This method is a client-server system based on processing images taken by a smart phone. The method is explained through three major tasks: food recognition, image segmentation, and portion estimation. Each part is evaluated separately as well as the whole integrated system designed to estimate food portion.

Regarding to food recognition task, a fine-tuned version of a deep convolutional neural network model introduced by Google, Inception V3, is used. The method achieves very good results on three main food image datasets improving the best published results by a relevant amount. It achieves 88.28%, 81.45%, and 76.17% accuracy, respectively, on ETH Food-101, UEC FOOD 100, and UEC FOOD 256 which are about 17.87%, 2.68%, and 8.6% better than the best published results.

Regarding to food portion estimation, a system is introduced to estimate food intake volume using image-based modeling. The approach includes a semi-automatic segmentation method which is designed for smartphones to be interactive and user-friendly.

As to semi-automatic segmentation, an interactive graph cut algorithm is developed. Also, a stocastic method is designed to detect a small checkerborad in images. This checkerborad is used as both size and ground reference . The algorithm outperforms basic methods provided by MATLAB®and OpenCV.
The method achieves about 92% accuracy in the final integrated system of food volume estimation. The processing time of the whole method is about 23s on a regular PC.

As future developments, there are a few main issues to tackle:

- For portion estimation, this study assumes that there is only one food item in the image. Handling more than one food item and estimating the volumes separately could be an important development for the future works.
- Exploiting some techniques like parallelization or GPU programming could potentially improve the processing time significantly, especially during the modeling process which takes most of the processing time.
- A volume estimation algorithm for non-dense point clouds which is able to work also with non-convex models needs to be developed.
- Including context information, such as time, place, personal habits or history, can be a big help both in food recognition and portion estimation.
- In this work, nutrition facts are not extracted. Nutrition facts can be obtained from a recipe dataset, however organizing a comprehensive dataset and managing the diversity of recipes in different places or times, could make it a challenging task.

Bibliography

[1] Diabetes, 2015. URL: http://www.who.int/mediacentre/factsheets/fs312/en/.

[2] Emil Jovanov, Edward Sazonov, and Carmen Poon. Sensors and systems for obesity care and research. In *Engineering in Medicine and Biology Society (EMBC), 2014 36th Annual International Conference of the IEEE*, pages 3188–3191. IEEE, 2014. doi:10.1109/EMBC.2014.6944300.

[3] AE Mesas, M Muñoz-Pareja, E López-García, and F Rodríguez-Artalejo. Selected eating behaviours and excess body weight: a systematic review. *Obesity Reviews*, 13(2):106–135, 2012. doi:10.1111/j.1467-789X.2011.00936.x.

[4] M Barbara E Livingstone and Alison E Black. Markers of the validity of reported energy intake. *The Journal of nutrition*, 133(3):895S–920S, 2003.

[5] Dale A Schoeller. Limitations in the assessment of dietary energy intake by self-report. *Metabolism*, 44:18–22, 1995. doi:10.1016/0026-0495(95)90204-X.

[6] Jelia C Witschi. Short-term dietary recall and recording methods. *Nutritional epidemiology*, 4:52–68, 1990. doi:10.4178/epih/e2014009.

[7] Food scanner prize, 2015. URL: http://ec.europa.eu/research/horizonprize/index.cfm?prize=food-scanner.

[8] Hamid Hassannejad, Guido Matrella, Monica Mordonini, and Stefano Cagnoni. A stochastic approach to detect small checkerboards. In *AI* IA 2015 Advances in Artificial Intelligence: XIVth International Conference of the Italian Association for Artificial Intelligence*, pages 75–86. Springer, 2015. doi:10.1007/978-3-319-24309-2.

[9] Cecilia Bergh, Ulf Brodin, Greger Lindberg, and per Södersten. Randomized controlled trial of a treatment for anorexia and bulimia nervosa. *Proceedings of the National Academy of Sciences of the United States of America*, 99(14):9486–9491, 2002. URL: http://www.jstor.org/stable/3059225, doi:10.1073/pnas.142284799.

[10] Anna L Ford, Cecilia Bergh, Per Södersten, Matthew A Sabin, Sandra Hollinghurst, Linda P Hunt, Julian PH Shield, et al. Treatment of childhood obesity by retraining eating behaviour: randomised controlled trial. *Bmj*, 340, 2010. doi:10.1136/bmj.b5388.

[11] Nana Gletsu-Miller and Megan A McCrory. Modifying eating behavior: Novel approaches for reducing body weight, preventing weight regain, and reducing chronic disease risk. *Advances in Nutrition: An International Review Journal*, 5(6):789–791, 2014. doi:10.3945/an.114.006601.

[12] Jie Li, Na Zhang, Lizhen Hu, Ze Li, Rui Li, Cong Li, and Shuran Wang. Improvement in chewing activity reduces energy intake in one meal and modulates plasma gut hormone concentrations in obese and lean young chinese men. *The American journal of clinical nutrition*, 94(3):709–716, 2011. doi:10.3945/ajcn.111.015164.

[13] Eric Robinson, Eva Almiron-Roig, Femke Rutters, Cees de Graaf, Ciarán G Forde, Catrin Tudur Smith, Sarah J Nolan, and Susan A Jebb. A systematic review and meta-analysis examining the effect of eating rate on energy intake and hunger. *The American journal of clinical nutrition*, pages ajcn–081745, 2014. doi:10.3945/ajcn.113.081745.

[14] José L Santos, Judith A Ho-Urriola, Andrea González, Susan V Smalley, Patricia Domínguez-Vásquez, Rodrigo Cataldo, Ana M Obregón, Paola Amador, Gerardo Weisstaub, and M Isabel Hodgson. Association between eating behavior scores and obesity in chilean children. *Nutr J*, 10(108):10–1186, 2011. doi:10.1186/1475-2891-10-108.

[15] Edward Sazonov and Michael R Neuman. *Wearable Sensors: Fundamentals, implementation and applications*. Elsevier, 2014. doi:10.1016/B978-0-12-418662-0.00027-1.

[16] Edward S Sazonov, Stephanie AC Schuckers, Paulo Lopez-Meyer, Oleksandr Makeyev, Edward L Melanson, Michael R Neuman, and James O Hill. Toward objective monitoring of ingestive behavior in free-living population. *Obesity*, 17(10):1971–1975, 2009. doi:10.1038/oby.2009.153.

[17] S. Movassaghi, M. Abolhasan, J. Lipman, D. Smith, and A. Jamalipour. Wireless body area networks: A survey. *IEEE Communications Surveys Tutorials*, 16(3):1658–1686, Third 2014. doi:10.1109/SURV.2013.121313.00064.

[18] Adnan Nadeem, Muhammad Azhar Hussain, Obaidullah Owais, Abdul Salam, Sarwat Iqbal, and Kamran Ahsan. Application specific study, analysis and classification of body area wireless sensor network applications. *Computer Networks*, 2015. doi:10.1016/j.comnet.2015.03.002.

[19] Oliver Amft, Holger Junker, and Gerhard Troster. Detection of eating and drinking arm gestures using inertial body-worn sensors. In *Proceedings of the Ninth IEEE International Symposium on Wearable Computers*, ISWC '05, pages 160–163, Washington, DC, USA, 2005. IEEE Computer Society. URL: http://dx.doi.org/10.1109/ISWC.2005.17, doi:10.1109/ISWC.2005.17.

[20] Yujie Dong. *Tracking wrist motion to detect and measure the eating intake of free-living humans*. PhD thesis, Clemson University, 2012. doi:10.1109/JBHI.2013.2282471.

[21] Raul I. Ramos-Garcia and Adam W. Hoover. A study of temporal action sequencing during consumption of a meal. In *Proceedings of the International Conference on Bioinformatics, Computational Biology and Biomedical Informatics*, BCB'13, pages 68:68–68:75, New York, NY, USA, 2013. ACM. URL: http://doi.acm.org/10.1145/2506583.2506596, doi:10.1145/2506583.2506596.

[22] O. Amft. A wearable earpad sensor for chewing monitoring. In *Sensors, 2010 IEEE*, pages 222–227, Nov 2010. doi:10.1109/ICSENS.2010.5690449.

[23] J. Liu, E. Johns, L. Atallah, C. Pettitt, B. Lo, G. Frost, and G. Z. Yang. An intelligent food-intake monitoring system using wearable sensors. In *2012 Ninth International Conference on Wearable and Implantable Body Sensor Networks*, pages 154–160, May 2012. doi:10.1109/BSN.2012.11.

[24] Vasileios Papapanagiotou, Christos Diou, Zhou Lingchuan, Janet van den Boer, Monica Mars, and Anastasios Delopoulos. Fractal nature of chewing sounds. In *New Trends in Image Analysis and Processing–ICIAP 2015 Workshops*, pages 401–408. Springer, 2015. doi:10.1007/978-3-319-23222-5-49.

[25] S. PaÃ§ler and W. J. Fischer. Food intake activity detection using a wearable microphone system. In *Intelligent Environments (IE), 2011 7th International Conference on*, pages 298–301, July 2011. doi:10.1109/IE.2011.9.

[26] S. PÃd'Ã§ler and W. J. Fischer. Food intake monitoring: Automated chew event detection in chewing sounds. *IEEE Journal of Biomedical and Health Informatics*, 18(1):278–289, Jan 2014. doi:10.1109/JBHI.2013.2268663.

[27] E. S. Sazonov and J. M. Fontana. A sensor system for automatic detection of food intake through non-invasive monitoring of chewing. *IEEE Sensors Journal*, 12(5):1340–1348, May 2012. doi:10.1109/JSEN.2011.2172411.

[28] C Nederkoorn, FTY Smulders, and A Jansen. Recording of swallowing events using electromyography as a non-invasive measurement of salivation. *Appetite*, 33(3):361–369, 1999.

[29] O. Amft and G. Troster. Methods for detection and classification of normal swallowing from muscle activation and sound. In *2006 Pervasive Health Conference and Workshops*, pages 1–10, Nov 2006. doi:10.1109/PCTHEALTH.2006.361624.

[30] Oleksandr Makeyev, Paulo Lopez-Meyer, Stephanie Schuckers, Walter Besio, and Edward Sazonov. Automatic food intake detection based on swallowing sounds. *Biomedical signal processing and control*, 7(6):649–656, 2012.

[31] Muhammad Farooq, Juan M Fontana, and Edward Sazonov. A novel approach for food intake detection using electroglottography. *Physiological measurement*, 35(5):739, 2014. doi:10.1088/0967-3334/35/5/739.

[32] J. M. Fontana, M. Farooq, and E. Sazonov. Automatic ingestion monitor: A novel wearable device for monitoring of ingestive behavior. *IEEE Transactions on Biomedical Engineering*, 61(6):1772–1779, June 2014. doi:10.1109/TBME.2014.2306773.

[33] Christos Maramis, Christos Diou, Ioannis Ioakeimidis, Irini Lekka, Gabriela Dudnik, Monica Mars, Nicos Maglaveras, Cecilia Bergh, and Anastasios Delopoulos. Preventing obesity and eating disorders through behavioural modifications: The SPLENDID vision. In *Wireless Mobile Communication and Healthcare (Mobihealth), 2014 EAI 4th International Conference on*, pages 7–10. IEEE, 2014. doi:10.1109/MOBIHEALTH.2014.7015895.

[34] Monika Frontczak and Pawel Wargocki. Literature survey on how different factors influence human comfort in indoor environments. *Building and Environment*, 46(4):922–937, 2011. doi:10.1016/j.buildenv.2010.10.021.

[35] M. Puri, Zhiwei Zhu, Q. Yu, A. Divakaran, and H. Sawhney. Recognition and volume estimation of food intake using a mobile device. In *Applications of Computer Vision (WACV), 2009 Workshop on*, pages 1–8, Dec 2009. doi:10.1109/WACV.2009.5403087.

[36] Fengqing Zhu, Marc Bosch, Insoo Woo, SungYe Kim, Carol J Boushey, David S Ebert, and Edward J Delp. The use of mobile devices in aiding dietary assessment and evaluation. *Selected Topics in Signal Processing, IEEE Journal of*, 4(4):756–766, 2010. doi:10.1109/JSTSP.2010.2051471.

[37] Luciano Oliveira, Victor Costa, Gustavo Neves, Talmai Oliveira, Eduardo Jorge, and Miguel Lizarraga. A mobile, lightweight, poll-based food identification system. *Pattern Recognition*, 47(5):1941–1952, 2014. doi:10.1016/j.patcog.2013.12.006.

[38] Stephen O'Hara and Bruce A Draper. Introduction to the bag of features paradigm for image classification and retrieval. *arXiv preprint arXiv:1101.3354*, 2011.

[39] Mei Chen, Kapil Dhingra, Wen Wu, Lei Yang, Rahul Sukthankar, and Jie Yang. PFID: Pittsburgh fast-food image dataset. In *Image Processing (ICIP), 2009 16th IEEE International Conference on*, pages 289–292. IEEE, 2009. doi:10.1109/ICIP.2009.5413511.

[40] M. M. Anthimopoulos, L. Gianola, L. Scarnato, P. Diem, and S. G. Mougiakakou. A food recognition system for diabetic patients based on an optimized bag-of-features model. *IEEE Journal of Biomedical and Health Informatics*, 18(4):1261–1271, July 2014. doi:10.1109/JBHI.2014.2308928.

[41] G. M. Farinella, M. Moltisanti, and S. Battiato. Classifying food images represented as bag of textons. In *2014 IEEE International Conference on Image Processing (ICIP)*, pages 5212–5216, Oct 2014. doi:10.1109/ICIP.2014.7026055.

[42] H. Hoashi, T. Joutou, and K. Yanai. Image recognition of 85 food categories by feature fusion. In *Multimedia (ISM), 2010 IEEE International Symposium on*, pages 296–301, Dec 2010. doi:10.1109/ISM.2010.51.

[43] Yuuki Matsuda and Katsuki Yanai. Multiple-food recognition considering co-occurrence employing manifold ranking. In *Pattern Recognition (ICPR), 2012 21st International Conference on*, pages 2017–2020. IEEE, 2012.

[44] Yoshiyuki Kawano and Keiji Yanai. Foodcam: A real-time mobile food recognition system employing fisher vector. In *MultiMedia Modeling*, pages 369–373. Springer, 2014. doi:10.1007/978-3-319-04117-9-38.

[45] Shaobo Fang, Chang Liu, Fengqing Zhu, Carol Boushey, and Edward Delp. A printer indexing system for color calibration with applications in dietary assessment. In *New Trends in Image Analysis and Processing–ICIAP 2015 Workshops*, pages 358–365. Springer, 2015. doi:10.1007/978-3-319-23222-5-44.

[46] Hongsheng He, Fanyu Kong, and Jindong Tan. Dietcam: Multiview food recognition using a multikernel svm. *IEEE journal of biomedical and health informatics*, 20(3):848–855, 2016. doi:10.1109/JBHI.2015.2419251.

[47] Yulia Eskin and Alex Mihailidis. An intelligent nutritional assessment system. In *2012 AAAI Fall Symposium Series*, 2012.

[48] Minami Wazumi, Xian-Hua Han, Danni Ai, and Yen-Wei Chen. Auto-recognition of food images using SPIN feature for food-log system. In *Computer Sciences and Convergence Information Technology (ICCIT), 2011 6th International Conference on*, pages 874–877. IEEE, 2011.

[49] R. Almaghrabi, G. Villalobos, P. Pouladzadeh, and S. Shirmohammadi. A novel method for measuring nutrition intake based on food image. In *Instrumentation and Measurement Technology Conference (I2MTC), 2012 IEEE International*, pages 366–370, May 2012. doi:10.1109/I2MTC.2012.6229581.

[50] Giovanni Maria Farinella, Marco Moltisanti, and Sebastiano Battiato. Food recognition using consensus vocabularies. In *New Trends in Image Analysis and Processing–ICIAP 2015 Workshops*, pages 384–392. Springer, 2015. doi:10.1007/978-3-319-23222-5-47.

[51] Taichi Joutou and Keiji Yanai. A food image recognition system with multiple kernel learning. In *2009 16th IEEE International Conference on Image Processing (ICIP)*, pages 285–288, Nov 2009. doi:10.1109/ICIP.2009.5413400.

[52] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015. doi:10.1038/nature14539.

[53] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012. URL: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

[54] Stergios Christodoulidis, Marios Anthimopoulos, and Stavroula Mougiakakou. Food recognition for dietary assessment using deep convolutional neural networks. In *New Trends in Image Analysis and Processing–ICIAP 2015 Workshops*, pages 458–465. Springer, 2015. doi:10.1007/978-3-319-23222-5-56.

[55] Keiji Yanai and Yoshiyuki Kawano. Food image recognition using deep convolutional network with pre-training and fine-tuning. In *Multimedia & Expo Workshops (ICMEW), 2015 IEEE International Conference on*, pages 1–6. IEEE, June 2015. doi:10.1109/ICMEW.2015.7169816.

[56] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

[57] Ye He, Chang Xu, Neha Khanna, Carol J Boushey, and Edward J Delp. Context based food image analysis. In *Image Processing (ICIP), 2013 20th IEEE International Conference on*, pages 2748–2752. IEEE, 2013. doi:10.1109/ICIP.2013.6738566.

[58] Keigo Kitamura, Toshihiko Yamasaki, and Kiyoharu Aizawa. Foodlog: Capture, analysis and retrieval of personal food images via web. In *Proceedings of the ACM Multimedia 2009 Workshop on Multimedia for Cooking and Eating Activities*, CEA '09, pages 23–30, New York, NY, USA, 2009. ACM. URL: http://doi.acm.org/10.1145/1630995.1631001, doi:10.1145/1630995.1631001.

[59] Yu Wang, Ye He, Fengqing Zhu, Carol Boushey, and Edward Delp. The use of temporal information in food image analysis. In *New Trends in Image Analysis and Processing–ICIAP 2015 Workshops*, pages 317–325. Springer, 2015. doi:10.1007/978-3-319-23222-5-39.

[60] Yoshiyuki Kawano and Keiji Yanai. FoodCam-256: A large-scale real-time mobile food recognitionsystem employing high-dimensional features and compression of classifier weights. In *Proceedings of the ACM International Conference on Multimedia*, pages 761–762. ACM, 2014. doi:10.1145/2647868.2654869.

[61] M. H. Rahmana, M. R. Pickering, D. Kerr, C. J. Boushey, and E. J. Delp. A new texture feature for improved food recognition accuracy in a mobile phone based dietary assessment system. In *Multimedia and Expo Workshops (ICMEW), 2012 IEEE International Conference on*, pages 418–423, July 2012. doi:10.1109/ICMEW.2012.79.

[62] Giovanni Maria Farinella, Dario Allegra, and Filippo Stanco. A benchmark dataset to study the representation of food images. In *Computer Vision-ECCV 2014 Workshops*, pages 584–599. Springer, 2014. doi:10.1007/978-3-319-16199-0-41.

[63] Hokuto Kagaya and Kiyoharu Aizawa. Highly accurate food/non-food image classification based on a deep convolutional neural network. In *New Trends in Image Analysis and Processing–ICIAP 2015 Workshops*, pages 350–357. Springer, 2015. doi:10.1007/978-3-319-23222-5-43.

[64] Yoshiyuki Kawano and Keiji Yanai. Automatic expansion of a food image dataset leveraging existing categories with domain adaptation. In *Computer Vision-ECCV 2014 Workshops*, pages 3–17. Springer, 2014. doi:10.1007/978-3-319-16199-0-1.

[65] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101–mining discriminative components with random forests. In *Computer Vision–ECCV 2014*, pages 446–461. Springer, 2014. doi:10.1007/978-3-319-10599-4-29.

[66] Parisa Pouladzadeh, Abdulsalam Yassine, and Shervin Shirmohammadi. FooDD: Food detection dataset for calorie measurement using food images. In *New Trends in Image Analysis and Processing–ICIAP 2015 Workshops*, pages 441–448. Springer, 2015. doi:10.1007/978-3-319-23222-5-54.

[67] Giovanni Maria Farinella, Dario Allegra, Filippo Stanco, and Sebastiano Battiato. On the exploitation of one class classification to distinguish food

vs non-food images. In *New Trends in Image Analysis and Processing–ICIAP 2015 Workshops*, pages 375–383. Springer, 2015. doi:10.1007/978-3-319-23222-5-46.

[68] Geeta Shroff, Asim Smailagic, and Daniel P Siewiorek. Wearable context-aware food recognition for calorie monitoring. In *2008 12th IEEE International Symposium on Wearable Computers*, pages 119–120. IEEE, 2008.

[69] Wen Wu and Jie Yang. Fast food recognition from videos of eating for calorie estimation. In *2009 IEEE International Conference on Multimedia and Expo*, pages 1210–1213. IEEE, 2009.

[70] Fanyu Kong and Jindong Tan. DietCam: Automatic dietary assessment with mobile camera phones. *Pervasive and Mobile Computing*, 8(1):147–163, 2012. doi:10.1016/j.pmcj.2011.07.003.

[71] Yoshiyuki Kawano and Keiji Yanai. Food image recognition with deep convolutional features. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication*, pages 589–593. ACM, 2014.

[72] Fengqing Zhu, Marc Bosch, Neha Khanna, Carol J Boushey, and Edward J Delp. Multiple hypotheses image segmentation and classification with application to dietary assessment. *Biomedical and Health Informatics, IEEE Journal of*, 19(1):377–388, 2015. doi:10.1109/JBHI.2014.2304925.

[73] Hamid Hassannejad, Guido Matrella, Paolo Ciampolini, Ilaria DeMunari, Monica Mordonini, and Stefano Cagnoni. Food image recognition using very deep convolutional networks. In *ACM Multimedia Workshops*, page to appear, 2016.

[74] Renfeng Liu. *Food Recognition and Detection with Minimum Supervision*. PhD thesis, The University of Western Ontario, 2016.

[75] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *CoRR*, abs/1312.4400, 2013. URL: http://arxiv.org/abs/1312.4400.

[76] Stuart Geman. Hierarchy in machine and natural vision. In *Proceedings of the Scandinavian Conference on Image Analysis*, volume 1, pages 179–184, 1999.

[77] Kevin Jarrett, Koray Kavukcuoglu, Yann Lecun, et al. What is the best multi-stage architecture for object recognition? In *2009 IEEE 12th International Conference on Computer Vision*, pages 2146–2153, Sept 2009. doi:10.1109/ICCV.2009.5459469.

[78] Inception in TensorFlow, 2016. https://github.com/tensorflow/models/tree/master/inception.

[79] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *CoRR*, abs/1404.7828, 2014. URL: http://arxiv.org/abs/1404.7828.

[80] Sanjeev Arora, Aditya Bhaskara, Rong Ge, and Tengyu Ma. Provable bounds for learning some deep representations. In *ICML*, pages 584–592, 2014.

[81] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. *CoRR*, abs/1512.00567, 2015. URL: http://arxiv.org/abs/1512.00567.

[82] Christian Szegedy, Sergey Ioffe, and Vincent Vanhoucke. Inception-v4, inception-resnet and the impact of residual connections on learning. *CoRR*, abs/1602.07261, 2016. URL: http://arxiv.org/abs/1602.07261.

[83] Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 06(02):107–116, 1998. URL: http://www.worldscientific.com/doi/abs/10.1142/S0218488598000094, arXiv:http://www.worldscientific.com/doi/pdf/10.1142/S0218488598000094, doi:10.1142/S0218488598000094.

[84] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101 – mining discriminative components with random forests. In *European Conference on Computer Vision*, 2014.

[85] Y. Matsuda, H. Hoashi, and K. Yanai. Recognition of multiple-food images by detecting candidate regions. In *Proc. of IEEE International Conference on Multimedia and Expo (ICME)*, 2012.

[86] TensorFlow, 2016. https://www.tensorflow.org/.

[87] Shital Adarsh Raut, M Raghuwanshi, R Dharaskar, and Adarsh Raut. Image segmentation–a state-of-art survey for prediction. In *Advanced Computer Control, 2009. ICACC'09. International Conference on*, pages 420–424. IEEE, 2009.

[88] Faliu Yi and Inkyu Moon. Image segmentation: A survey of graph-cut methods. In *Systems and Informatics (ICSAI), 2012 International Conference on*, pages 1936–1941. IEEE, 2012.

[89] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing (3rd Edition)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2006.

[90] John Hopcroft and Robert Tarjan. Algorithm 447: Efficient algorithms for graph manipulation. *Communications of the ACM*, 16(6):372–378, 1973. doi:10.1145/362248.362272.

[91] R Muthukrishnan and M Radha. Edge detection techniques for image segmentation. *International Journal of Computer Science & Information Technology*, 3(6):259, 2011.

[92] C. K. Martin, S. Kaya, and B. K. Gunturk. Quantification of food intake using food image analysis. In *2009 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 6869–6872, Sept 2009. doi:10.1109/IEMBS.2009.5333123.

[93] Joachim Dehais, Marios Anthimopoulos, and Stavroula Mougiakakou. Dish detection and segmentation for dietary assessment on smartphones. In *New Trends in Image Analysis and Processing–ICIAP 2015 Workshops*, pages 433–440. Springer, 2015. doi:10.1007/978-3-319-23222-5-53.

[94] Gregorio Villalobos, Rana Almaghrabi, Behnoosh Hariri, and Shervin Shirmohammadi. A personal assistive system for nutrient intake monitoring. In *Proceedings of the 2011 international ACM workshop on Ubiquitous meta user interfaces*, pages 17–22. ACM, 2011. doi:10.1145/2072652.2072657.

[95] Fengqing Zhu, Marc Bosch, TusaRebecca Schap, Nitin Khanna, David S Ebert, Carol J Boushey, and Edward J Delp. Segmentation assisted food classification for dietary assessment. In *SPIE Electronic Imaging*, pages 78730B–78730B. International Society for Optics and Photonics, 2011. doi:10.1117/12.877036.

[96] Mabel Mengzi Zhang. Identifying the cuisine of a plate of food. *Univ. of California at San Diego, La Jolla, CA, USA, Tech. Rep. CSE*, 190, 2011.

[97] Yuri Boykov and Gareth Funka-Lea. Graph cuts and efficient nd image segmentation. *International journal of computer vision*, 70(2):109–131, 2006.

[98] Yuri Boykov and Vladimir Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE transactions on pattern analysis and machine intelligence*, 26(9):1124–1137, 2004.

[99] Victor Lempitsky, Pushmeet Kohli, Carsten Rother, and Toby Sharp. Image segmentation with a bounding box prior. In *2009 IEEE 12th International Conference on Computer Vision*, pages 277–284. IEEE, 2009.

[100] Nhat Vu and BS Manjunath. Shape prior segmentation of multiple objects with graph cuts. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[101] C Bishop. Pattern recognition and machine learning (information science and statistics), 1st edn. 2006. corr. 2nd printing edn, 2007.

[102] Pushmeet Kohli and Philip HS Torr. Dynamic graph cuts for efficient inference in markov random fields. *IEEE transactions on pattern analysis and machine intelligence*, 29(12):2079–2088, 2007.

[103] Pushmeet Kohli and Philip HS Torr. Dynamic graph cuts and their applications in computer vision. In *Computer Vision*, pages 51–108. Springer, 2010.

[104] Ning Yao. *Food dimension estimation from a single image using structured lights*. PhD thesis, University of Pittsburgh, 2011.

[105] Chang Xu, Ye He, Nitin Khannan, Albert Parra, Carol Boushey, and Edward Delp. Image-based food volume estimation. In *Proceedings of the 5th international workshop on Multimedia for cooking & eating activities*, pages 75–80. ACM, 2013. doi:10.1145/2506023.2506037.

[106] M. H. Rahman, Q. Li, M. Pickering, M. Frater, D. Kerr, C. Bouchey, and E. Delp. Food volume estimation in a mobile phone based dietary assessment system. In *Signal Image Technology and Internet Based Systems (SITIS), 2012 Eighth International Conference on*, pages 988–995, Nov 2012. doi:10.1109/SITIS.2012.146.

[107] Joachim Dehais, Sergey Shevchik, Peter Diem, and Stavroula G Mougiakakou. Food volume computation for self dietary assessment applications. In *Bioinformatics and Bioengineering (BIBE), 2013 IEEE 13th International Conference on*, pages 1–4. IEEE, 2013. doi:10.1109/BIBE.2013.6701615.

[108] Wenyan Jia, Yaofeng Yue, John D Fernstrom, Zhengnan Zhang, Yongquan Yang, and Mingui Sun. 3d localization of circular feature in 2d image and application to food volume estimation. In *Engineering in Medicine and Biology Society (EMBC), 2012 Annual International Conference of the IEEE*, pages 4545–4548. IEEE, 2012. doi:10.1109/EMBC.2012.6346978.

[109] Hamid Hassannejad, Guido Matrella, Monica Mordonini, and Stefano Cagnoni. Using small checkerboards as size reference: A model-based approach. In *New Trends in Image Analysis and Processing–ICIAP 2015 Workshops*, pages 393–400. Springer, 2015. doi:10.1007/978-3-319-23222-5-48.

[110] Gary Bradski and Adrian Kaehler. *Learning OpenCV: Computer vision with the OpenCV library*. " O'Reilly Media, Inc.", 2008.

[111] Mathworks documentation, 2016. http://it.mathworks.com/help/vision/ug/camera-calibration.html. URL: http://it.mathworks.com/help/vision/ug/camera-calibration.html.

[112] Kenneth R. Castleman. *Digital image processing*. Prentice Hall Press, Upper Saddle River, NJ, USA, 1996.

[113] K. Demaagd, A. Oliver, N. Oostendorp, and K. Scott. *Practical Computer Vision with SimpleCV: The Simple Way to Make Technology See*. Safari Books Online. O'Reilly Media, 2012. URL: http://books.google.it/books?id=4st9l6QlkpUC.

[114] Jianbo Shi and Carlo Tomasi. Good features to track. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*, pages 593–600. IEEE, 1994.

[115] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Computer vision and image understanding*, 110(3):346–359, 2008.

[116] Stefan Leutenegger, Margarita Chli, and Roland Y Siegwart. Brisk: Binary robust invariant scalable keypoints. In *2011 International conference on computer vision*, pages 2548–2555. IEEE, 2011.

[117] Richard Szeliski. *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

[118] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[119] Rahul Raguram, Jan-Michael Frahm, and Marc Pollefeys. A comparative analysis of ransac techniques leading to adaptive real-time random sample consensus. In *European Conference on Computer Vision*, pages 500–513. Springer, 2008.

[120] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[121] Jing-Fu Huang, Shang-Hong Lai, and Chia-Ming Cheng. Robust fundamental matrix estimation with accurate outlier detection. *J. Inf. Sci. Eng.*, 23(4):1213–1225, 2007. doi:http://www.iis.sinica.edu.tw/page/jise/2007/200707_16.html.

[122] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[123] Richard I Hartley and Peter Sturm. Triangulation. *Computer vision and image understanding*, 68(2):146–157, 1997.

[124] Daniel Lélis Baggio. *Mastering OpenCV with practical computer vision projects*. Packt Publishing Ltd, 2012.

[125] Bill Triggs, Philip F McLauchlan, Richard I Hartley, and Andrew W Fitzgibbon. Bundle adjustmentâĂŤa modern synthesis. In *International workshop on vision algorithms*, pages 298–372. Springer, 1999.

[126] R. Almaghrabi, G. Villalobos, P. Pouladzadeh, and S. Shirmohammadi. A novel method for measuring nutrition intake based on food image. In *Instrumentation and Measurement Technology Conference (I2MTC), 2012 IEEE International*, pages 366–370, May 2012. doi:10.1109/I2MTC.2012.6229581.

[127] Rainer Storn and Kenneth Price. *Differential evolution-a simple and efficient adaptive scheme for global optimization over continuous spaces*, volume 3. ICSI Berkeley, 1995.

[128] Roberto Ugolotti, Youssef S.G. Nashed, Pablo Mesejo, Åăpela IvekoviÄ■, Luca Mussi, and Stefano Cagnoni. Particle swarm optimization and differential evolution for model-based object detection. *Applied Soft Computing*, 13(6):3092 – 3105, 2013. doi:10.1016/j.asoc.2012.11.027.

[129] S. Das and P. N. Suganthan. Differential evolution: A survey of the state-of-the-art. *IEEE Transactions on Evolutionary Computation*, 15(1):4–31, Feb 2011. doi:10.1109/TEVC.2010.2059031.

[130] Charles L Lawson and Richard J Hanson. *Solving least squares problems*, volume 15. SIAM, 1995.

[131] Noah Snavely, Steven M Seitz, and Richard Szeliski. Modeling the world from internet photo collections. *International Journal of Computer Vision*, 80(2):189–210, 2008.

[132] Qhull, 2016. URL: http://www.qhull.org/.

Acknowledgements

I would like to express my appreciation to all those who provided me the possibility to complete this thesis.
I would like to thank Professor Guido Matrella who tutored me throughout this PhD. He supported me in all the issues. He was always graciously available for me and provided many valuable guidances during last three years which certainly elevated the quality of this thesis.
I wish to thank Professor Stefano Cagnoni for many things. In the first place, he trusted me and introduced me to the project management. Working in IBIS laboratory was a great opportunity to benefit from his vast knowledge and enjoy his constructive comments. Moreover, his welcoming attitude, whenever I faced a problem or needed some guidance, made the whole experience of PhD much more pleasant.
I would like to thank Professor Paolo Ciampolini for accepting me to collaborate in the project. He was always supportive, specially when I needed it the most.
I would also like to thank Professor Ilaria De Munari who was a main member of the project. She also facilitated many of the bureaucratic procedures, letting me to focus on my job with the least worry about other things.
I would like to thank my fellow colleagues in IBIS lab., in particular Roberto Ugolotti and Luca Donati, who made it possible to work in a friendly and productive environment.
This thesis is dedicated to my parents who have given me all the best that I could ever have. Whatever I am or I have done, is beholden to them. I would like to thank them for their devoted life and their consistent love.

Last but not least, I dedicate this work to my wife Pouya Ahmadian and my daughter Tara. They are the center of my world and the best trophy I could ever wish in my life. Pouya has always been my biggest supporter and Tara, magically, turns everything to be more delightful.