Vol. 26, No. 1/4, 2017

Machine GRAPHICS & VISION

International Journal

Published by
The Faculty of Applied Informatics and Mathematics – WZIM
Warsaw University of Life Sciences – SGGW
Nowoursynowska 159, 02-776 Warsaw, Poland
in cooperation with
The Association for Image Processing, Poland – TPO

Applied Inverse Kinematics for Bipedal Characters Moving on the Diverse Terrain

Łukasz Burdka, Paweł Rohleder
Techland Sp. z o.o.
ul. Żółkiewskiego 3, 63-400 Ostrów Wlkp, Poland
http://techland.pl

Abstract.

A solution to the problem of adjusting the pose of an animated video game character to the diverse terrain and surroundings is proposed. It is an important task in every modern video game where there is a focus on animated characters. Not addressing this issue leads to major visual glitches such as legs hovering above the ground surface, or penetrating the obstacles while moving. As presented in this work, the described problem can be effectively solved by examining the surroundings in real-time and applying Inverse Kinematics (IK) as a procedural post process to the currently used animation.

Key words: inverse kinematics, animations, visual improvements, physics, game systems

1. Introduction

The reception of modern video games relies highly on their visual quality. Such quality is achieved not only by graphics rendering techniques, but also by making sure that the elements of a virtual world fit each other and form a believable whole [1]. In most of the First Person Perspective (FPP) and Third Person Perspective (TPP) games there is a strong focus on animated characters inhabiting the world. Such characters traverse the world guided by human players or Artificial Intelligence (AI). They interact with the game environment, such as terrain or obstacles, using physics systems dedicated for games.

Physics prevents characters from colliding with the world geometry, but in order to operate efficiently they provide only a very rough approximation of the character's shape. Traditionally it is a capsule or an ellipsoid. While the entire character is moved by the physics system, the motion of its body parts is realized by the Forward Kinematics (FK) animations. FK animations provide a time varying position and orientation of each of the character's virtual bones in its local coordinate system (Fig. 1).

Since the bones form a hierarchy and the position of the hierarchy root is controlled by another system (physics or any other locomotion controller) the absolute position of each bone can be determined. However the main issue with this standard approach is that animations do not adapt to the character's surroundings. Although different animations can be played while walking on a pavement and on the stairs, it is virtually

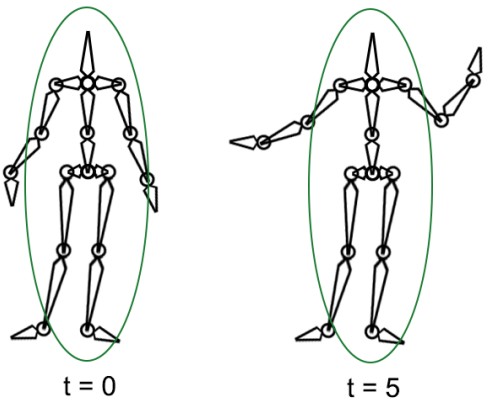


Fig. 1. Animated skeleton and its physical approximation.

impossible to prepare different versions of all animations for every possible terrain type. Obstacles lying on the ground pose an additional challenge to overcome. As a result, legs either hover above the terrain when they are above the bottom of the physical capsule, or penetrate obstacles when they move outside of it as a result of playing an animation (Fig. 2).

In order to tackle this issue, the character's surrounding would need to be examined and the animation would need to be adjusted. Further sections of this paper propose a solution of this problem by applying a trace-based and IK-based system. The overview of our method is presented in Section 2. Section 3 describes, in detail, how Inverse Kinematics works and how it is applied to an animated character's skeleton. In section 4 our work is summarized and further research directions are outlined.

2. Method overview

In video games, there is often a need to modify existing FK animations in real-time to fit the virtual environment where a character operates. Exemplary applications are adjusting the hand position while climbing, positioning hands on the steering wheel while driving, or adjusting feet to the terrain while standing and moving. In this paper we focus on the last case, since it is a very common problem in TPP and FPP games that utilize animated characters, regardless of the subject of the game. The general solution to all these issues can be divided into three steps. First, a position where the limb should be located needs to be found. Then IK can be applied to determine the correct configuration of the skeleton to match this requirement. Finally a blending between

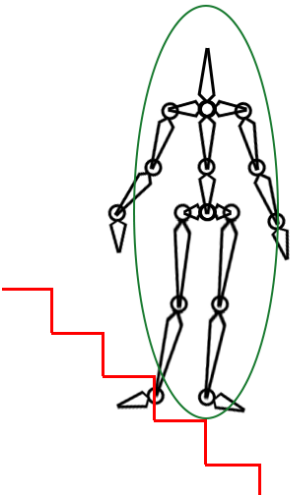


Fig. 2. One foot of the skeleton penetrates the stairs while the other hovers above them.

the FK and IK pose can be applied to make the animation look natural. The hardest problem in most real-life applications is how to find the desired position of the limbs?

When a human player looks at the game character, it is obvious for them where the legs should be, because they can see and interpret the whole rendered scene. This data is not as easily accessible and interpretable for game systems. Instead, the systems can use other means to examine the game environment in a certain location. Tracing is one of the most common tools used to analyze the geometry at the points of interest. Tracing is explained in Algorithm 2.1.

By tracing the surroundings of the character, the information how far the limbs can move, and where exactly they should be, can be precisely obtained. There are two sub-scenarios in which the correct configuration of legs needs to be determined. One of them

Algorithm 2.1 Level geometry tracing.

```
function trace(From, To, Geometry):
for all Piece in Geometry: do
    if Piece intersects Segment(From, To): then
        return Pair(IntersectionPoint, SurfaceNormal)
    end if
end for
return NoIntersection
```

is when the character stands still with both feet placed on the ground (static case) and the other is when the character is moving and at least one foot at a time moves above the ground (dynamic case). The static case is the simpler one. The solution for this problem is presented in Algorithm 2.2.

The terrain is traced vertically where each foot is located. When it is determined if each foot should be positioned above or below its FK pose, the Inverse Kinematics Equation (see Section 3) is applied for the bones of the leg to find the correct bone configuration. Thus IK poses for legs are obtained. If one of the legs was supposed to be positioned below the possible range of the leg, pelvis can be lowered to enable the pose. The dynamic case is more complex because it is not obvious where the tracing should be performed. The leg, which is on the ground, can be handled the same way as in the static case. However when the foot is in the air, its whole trajectory needs to be modified to provide believable movement without rapid changes of leg position. The modified trajectory should respect the altitude of the place where the foot is going to land on the ground. This position can be calculated using movement speed and the data about pacing extracted from the animation. Additionally the foot should be able to avoid obstacles crossing its trajectory so that the path it is going to traverse needs to be analyzed to find obstacles and modify the trajectory accordingly. The animations used with our system were manually annotated to mark moments when the foot lands on the

Algorithm 2.2 Determining leg configuration for the static case.

```
function positionLegsInStaticCase(FKSkeletonPose, Geometry):
LeftFootFK, RightFootFK, LeftCalfFK, RightCalfFK, LeftTighFK, RightTighFK,
        PelvisFK = getBonesFK(FKSkeletonPose)
LeftPosition, LeftOrientation = trace(over(LeftFootFK), under(LeftFootFK),
        Geometry)
RightPosition, RightOrientation = trace(over(RightFootFK), under(RightFootFK),
        Geometry)
LeftFootIK, LeftCalfIK, LeftTightIK = getConfigurationIK(LeftPosition,
        LeftFootFK, LeftCalfFK, LeftTightFK)

setOrientation(LeftFootIK, LeftOrientation)
RightFootIK, RightCalfIK, RightTightIK=getConfigurationIK(RightPosition,
        RightFootFK, RightCalfFK, RightTightFK)
setOrientation(RightFootIK, RightOrientation)
PelvisIK = PelvisFK
lowerPelvisIfNeeded(PelvisIK, LeftTightIK, RightTightIK)
blendPoses(FKSkeletonPose, LeftFootIK, RightFootIK, LeftCalfIK, RightCalfIK,
        LeftTighIK, RightTighIK, PelvisIK)
```

ground and is lifted into the air. The attempts to annotate the animations automatically gave poor results and led to many visual glitches. The simplified version of the algorithm used in the dynamic case is presented in Algorithm 2.3.

At first glance, the solution for the dynamic case looks as simple as for the static case, once the trajectory is obtained. The target of the trajectory is determined by tracing the predicted step location. Additionally, obstacles modify the trajectory if they intersect the predicted path (Fig. 3). The current IK position of the foot can be easily read from the trajectory. In both the static and dynamic case the foot position should be modified only vertically to avoid any modification of the animation's character. The problem, however, arises when the character changes the walking direction when the foot is in mid-air or modifies the walking speed. As a result, the trajectory needs to be reevaluated repeatedly when the foot is in mid-air. The new evaluated trajectory and the old one need to be blended smoothly to avoid rapid foot movement. In general, the ability to predict the position where the foot is going to land as soon as it takes

Algorithm 2.3 Determining leg configuration in the dynamic case.

```
function getTrajectory(CurrentStepPositionFK, Animation, WalkSpeed, Geometry):
NextStepFrame = getNextStepFrame(Animation)
NextStepPositionFK = getLegPosition(Animation, NextStepFrame,WalkSpeed)
RightPosition, RightOrientation = trace(over(RightFootFK),under(RightFootFK),
    Geometry)
LeftFootIK, LeftCalfIK, LeftTightIK = getConfigurationIK(LeftPosition,
    LeftFootFK, LeftCalfFK, LeftTightFK)
Trajectory = getAnimationTrajectory(Animation)
FinalPointIK = trace( over(Trajectory.end),under(Trajectory.end), Geometry)
modifyTrajectory(Trajectory.end, Trajectory, FinalPointIK)
for all substep in Range( CurrentStepPositionFK, NextStepPositionFK) do
   TracedPosition = trace(over(substep), under(substep), Geometry)
end for
AnimationPosition = getTrajectoryPoint(substep, Trajectory)
if isAbove(TracedPosition, AnimationPosition): then
   modifyTrajectory(substep, Trajectory, TracedPosition)
   return Trajectory
end if

function positionLegInDynamicCase( FKSkeletonPose, NewTrajectory, StepProgress):
PositionIK = getTrajectoryPosition( NewTrajectory, StepProgress)
//Apply analogically to the static case
getConfigurationIK(PositionIK, FKSkeletonPose)
```

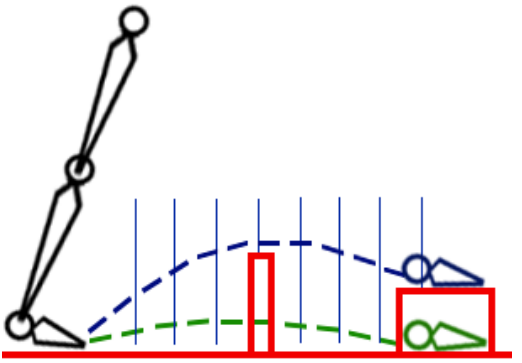


Fig. 3. The green dashed line is the FK trajectory. The blue dashed line is the IK trajectory modified by the vertical tracing of the FK trajectory.

off from the ground greatly improves the visual quality of applying Inverse Kinematics. Additionally, tracing the path only once can greatly reduce the computational power needed to obtain the correct trajectory. It is because in complex scenes the cost of precise trajectory tracing is much higher than the cost of calculating the IK equation itself and applying it to the skeleton.

3. Applying Inverse Kinematics

One of the most common algorithms for solving the IK problem is Cyclic Coordinate Descent (CCD) [2]. The main idea behind this method is to iteratively rotate each joint to move the effector closer to the target position. In case of CCD application for legs, the foot acts as an effector while the knee and hip form a two degree of freedom system [3]. The pseudo code for CCD algorithm is presented in Algorithm 3.1.

The algorithm converges fast for short bone chains but can be used for longer chains as well. It can be very effectively used for the case of finding the correct leg configuration of a bipedal character. The only challenge in this rather non-complex algorithm is to find the rotation angle for each joint. The sought angle and auxiliary vectors are shown in Fig. 4. To compute the angle, the following formulas must be solved:

$$\alpha = \arccos\left(\frac{d_T}{||d_T||} \bullet \frac{d_E}{||d_E||}\right), \tag{1}$$

$$r = \frac{d_T}{||d_T||} \times \frac{d_E}{||d_E||}, \tag{2}$$

where d_T is the vector to the target and d_E is the vector to the effector, and α is the angle, as shown in Fig. 4. The vector r is needed to determine the orientation of the angle α. Repeating these calculations until the effector's position error is below some arbitrary tolerance threshold leads to reaching the target location. The maximum number of iterations must be set to protect the algorithm in case the target is beyond the effector's reach. Additionally, in the unconstrained case, it would be possible to rotate the knee or hip beyond its anatomical movement range. Thus, IK constraints need to be applied to joints in order to limit the motion and allow the rotation only between certain angles. The only modification that needs to be done is to apply the clamp function when the angle is computed in each iteration for each joint.

Algorithm 3.1 Cyclic Coordinate Descent for series of joints.

```
function solveCCD(Joints, Effector, Target):
while isTooFar(Effector, Target): do
    //Start next to the effector and move to the last joint
    for all Joint in Joints: do
        ToTargetVector = Target - Joint
        ToEffectorVector = Effector - Joint
        RotationAngle = GetAngleToAlignEffectorAndVector(ToEffectorVector, ToTargetVector)
    end for
    GetAngleToAlignEffectorAndVector(ToEffectorVector, ToTargetVector)
    RotateJoint(Joint, RotationAngle)
end while
```

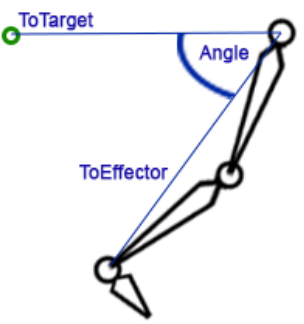


Fig. 4. Calculating the rotation of the hip joint, to move the foot effector closer to the target.

Once the correct pose is determined, it needs to blended with the FK animation pose to obtain a natural result. Parametric motion blending [4] is mostly applicable in this case, since there is often a need to control how much IK and how much FK should be used or to find a smooth transition between previous and current leg trajectory. In case of skeletal animations, where the pose is given by rotations of subsequent joints, the most effective way of blending poses is using Spherical Linear Interpolation (Slerp) [5]. For each blended joint the source and target quaternion must be known as well as the weight of blending. Given these three arguments, the sought angle of the joint can be calculated by solving the formula for Spherical Linear Interpolation between two joints with rotation given by quaternions q_1 and q_2:

$$\mathrm{Slerp}(q_1, q_2, w) = \frac{\sin\left((1-w)\cdot\alpha\right)}{\sin(\alpha)} q_1 + \frac{\sin(w\cdot\alpha)}{\sin(\alpha)} q_2 \tag{3}$$

where w is the blending weight and α is the angle between effector and target as introduced before.

4. Conclusion

Adjusting the legs of a bipedal animated character to diverse terrain is an important issue in modern video games. Failure to address this problem leads to glitches that decrease the overall visual quality of a game. Inverse Kinematics can be effectively applied to solve the leg configuration problem. Determining the correct feet positions for a moving character is not a trivial task. Tracing the scene's geometry allows to examine the character's path. The higher the speed of movement the more difficult it gets to produce a smooth and believable leg motion using the procedural post processing of IK. Frequent changes of movement direction make the step prediction inefficient and introduce the need of repeated trajectory reevaluation. At high speed, e.g. while running, it is recommended to gradually blend from the IK controlled pose towards the FK animated pose. It is because when legs move fast enough it is difficult to see if steps hit the ground precisely or hover above it. On the other hand repeated plan changes may lead to sharp and unnatural motion. While walking or standing, however, it is crucial to align the feet to the terrain and to make sure the legs do not penetrate obstacles while moving.

Acknowledgement

The paper was created as the result of the project "Industrial Research on the Technology of Scaled City for the Needs of Computer Games" (original title in Polish: „Badania przemysłowe nad technologią skalowanego miasta na potrzeby gier komputerowych").

The project is co-financed by the European Regional Development Fund under the Innovative Economy Operational Programme.

References

[1] C. C. Bracken and P. Skalski. Telepresence and video games: The impact of image quality. *PsychNology Journal*, 7(1):101–112, 2009. Retrieved Jan 10, 2017, from www.psychnology.org. Online: http://www.psychnology.org/File/PNJ7(1)/PSYCHNOLOGY_JOURNAL_7_1_BRACKEN.pdf].

[2] L. C. T. Wang and C. C. Chen. A combined optimization method for solving the inverse kinematics problems of mechanical manipulators. *IEEE Transactions on Robotics and Automation*, 7(4):489–499, Aug 1991. doi:10.1109/70.86079.

[3] R. H. Cannon. *Dynamics of Physical Systems.* McGraw-Hill, New York, 1967.

[4] A. Feng, Y. Huang, M. Kallmann, and A. Shapiro. An analysis of motion blending techniques. In M. Kallmann and K. Bekris, editors, *Motion in Games. Proc. 5th Int. Conf. on Motion in Games MIG 2012*, pages 232–243, Rennes, France, 2012. Springer Berlin Heidelberg. doi:10.1007/978-3-642-34710-8_22.

[5] K. Shoemake. Animating rotation with quaternion curves. *SIGGRAPH Computer Graphics*, 19(3):245–254, July 1985. doi:10.1145/325165.325242.

An Ensemble Feature Method for Food Classification

Niki Martinel, Claudio Piciarelli, Christian Micheloni
University of Udine, Italy
niki.martinel@uniud.it, claudio.piciarelli@uniud.it, christian.micheloni@uniud.it

Abstract. In the last years, several works on automatic image-based food recognition have been proposed, often based on texture feature extraction and classification. However, there is still a lack of proper comparisons to evaluate which approaches are better suited for this specific task. In this work, we adopt a Random Forest classifier to measure the performances of different texture filter banks and feature encoding techniques on three different food image datasets. Comparative results are given to show the performance of each considered approach, as well as to compare the proposed Random Forest classifiers with other feature-based state-of-the-art solutions.

Key words: food recognition, texture filter banks, feature encoding, Random Forest classifier

1. Introduction

According to the World Health Organization, in the last years there has been a rapid increase of diseases related to excessive or wrong food intake [39]. Obese people should constantly take note of their daily meals, both for self-monitoring and to acquire useful statistics for dietitians. This justifies the large amount of food diary applications that have recently been developed [4]. However, these applications typically require a manual annotation of the food intake, a tedious task that often discourages potential users. To face this problem, many food recognition works have been recently proposed, whose aim is to automatically classify food (and possibly its amount) directly from pictures.

Regardless of the specific application, automatic food recognition is a tough problem with many specific challenges. Differing from other common image classification tasks, in food recognition, generally, there is no spatial layout information to be exploited. While for example the recognition of paintings [22] or humans [11, 21] can benefit from prior knowledge on the spatial relationships between the parts to be detected (e.g., the head being always over the torso) this is rarely the case when considering food. More generally, food is typically non-rigid, and thus no structure information can be easily exploited. Intra-class variation is another source of uncertainty, since the recipe itself for the same food can vary depending on the location, the available ingredients and, last but not least, the personal taste of the cook. Finally, inter-class confusion is a source of potential problems too, since different foods may look very similar, as in many soups where the main ingredients may be hidden below the liquid level. On the other hand, food images often have distinctive properties, especially in terms of colors and textures,

which humans are able to exploit to recognize foods even from a single example, so the task is still tractable, despite the non-trivial challenges.

Recently, several excellent results have been obtained in the field of food recognition by using Convolutional Neural Networks and more in general Deep Learning approaches (e.g., [24, 28]). However, in practical applications this is not necessarily always the optimal approach, e.g. due to high computational requirements in training phase or the need for large datasets. Therefore, there are still many works on food recognition based on alternative approaches, typically relying on explicit feature extraction with texture filter banks. However, there is still a lack of a study showing which features are likely to be more useful among the many available ones. The aim of this work is thus to evaluate and compare different approaches based on texture filters, in order to gain a better insight on their performances. In this paper, we have considered five of the most important texture filter banks, namely Laws filters [19], Gabor filters [38], Schmid filters [35], Leung-Malik filters [20] and Maximum Response filters [36], and measured their performances when used in food classification tasks. Since the responses of such filters have high dimensionality, we also analyzed the effects of adopting different feature encoding schemes to obtain a compact feature representation. The task is performed by computing either the Bag-of-Words, the Fisher Vector or the Vector of Locally Aggregated Descriptors (VLAD) representation for each filter bank. We finally compared the discriminative power of each texture filter alone, and show that better results can be achieved using a proper combination of different filters. This is obtained by using a Random Forest of Decision Trees (RF) [3] classifier, which automatically detects and uses only the relevant features to produce a reliable classification. The main novelty of this work thus lies in giving a better understanding of texture filter approaches through comparative results, despite the adopted techniques are not novel *per se*. This work extends the preliminary results we discussed in [26].

2. Related work

During the last few years, the topic of food recognition for health-oriented applications has gained increasing popularity. For example, the work by Chen *et al.* [6] introduced a system exploiting different classifiers trained on multiple features. In particular, a Support Vector Machine is trained for each texture separately, and the results are fused to form a single classifier using a multi-class AdaBoost algorithm. Farinella *et al.* [10] exploit the texture information by applying a bank of rotation and scale invariant filters to each class of food images, in order to extract the texture-oriented features known as Textons. The feature space is then quantized via K-means to create a codebook of textons for each class. All the textons prototypes are collected in a single visual dictionary which is used to represent each image. A Support Vector Machine is finally used in the classification stage. Yang *et al.* [41] claim that spatial relationships between different

ingredients could be exploited for some specific food types, like sandwiches, where the meat is always between the bread slices. They perform a soft pixel-level segmentation of the image into eight ingredient types using a Semantic Texton Forest. Then, they compute pairwise statistics over the detected local ingredients, such as distance, orientation, etc. The statistics are accumulated in a multi-dimensional histogram, which is then used as a feature vector classified with a Support Vector Machine. Anthimopoulos *et al.* [1] investigate the application of Bag-of-Features approach to food recognition. In their work they systematically analyze the system performances under different setups, e.g. by changing the key point extraction techniques, feature descriptors and classifiers. Martinel *et al.* [25, 27] exploit different image features and train a committee of Extreme Learning Machines, each one specialized on a single feature, to perform food classification. The obtained results are processed by a Structural Support Vector Machine, which considers the full rankings from each single classifier, rather than just the best match, in order to merge them in a final global ranking.

Many works are explicitly tuned for food diary applications on smartphones and other mobile devices [15, 17, 18, 42, 43]. Kawano and Yanai [15, 17] for example are particularly concerned with real-time performances on an Android-based smartphone. To speed up the process, the user is asked to manually select a proper bounding box delimiting the food to be recognized. Then, the system extracts both color histograms and SURF-based Bag of Features, and uses them to assign the acquired image to one of 15 possible classes using a Support Vector Machine. Kong *et al.* [18] have developed DietCam, a smartphone-based system to help assessing daily food intakes. The system requires three images of each food to be recognized to increase the robustness against partial occlusion or lighting conditions. Classification is done using a SIFT-based Bag of Visual Words, and then searching for the best match against a database of known foods using a nearest-neighbor classifier. Zhu *et al.* propose a system for calories estimation via mobile phones [43]. Their approach consists in segmenting the images using different techniques (connected component analysis, active contours and normalized cuts). Then, they extract both color and texture features using color histograms and Gabor filters, and classify the images using a Support Vector Machine. Volume is estimated by means of a single-view calibrated camera and a reference marker in the scene. Calories estimation has been proposed also by Zhang *et al.* in [42], where BoW-encoded texture features are classified with Support Vector Machines and geometric considerations are used to estimate the food amount. Puri *et al.* [32] perform a similar task, but using 3D volume estimation. Some works also consider the problem of multiple foods in the same picture. For example, Matsuda and Yanai [30] merge the outputs of different region detectors to identify different foods, which are later classified using several texture features and a Support Vector Machine. They also exploit co-occurrence statistics on food items to improve the classification results.

Finally, in the last years deep learning techniques have obtained excellent results

in the field of food recognition. This is a novel approach since features are automatically learned, rather than manually defined. The typical works in this field rely on pre-trained Convolutional Neural Networks, such as ImageNet, and fine-tune them on a specific dataset. This is the approach adopted for example by Yanai and Kawano [40], although they also proposed a combined use of both CNN and conventional manually-defined image features to improve the achieved results [16]. Kagaya *et al.* [14] instead trained a CNN from scratch, but this required a large dataset, consisting in more than 170 000 food pictures acquired by a mobile app. Christodoulidis *et al.* [7] also use an CNN, but it requires a previous image segmentation step. They give performance results with varying choice of network hyperparameters. As mentioned in section 1, despite deep learning approaches have obtained excellent results, often outperforming traditional techniques, they are not always an optimal solution in real-world scenarios, e.g. for computational requirements or the need for large datasets. For this reason, this work focuses on traditional feature-extraction-based techniques only, and no comparative results with CNN or other deep learning approaches will be given.

All of the above mentioned works, except the deep learning ones, rely on some specific type of features to be extracted and processed. However, there is a lack of comparisons between different filter banks or feature encoding schemes. The main contribution of this work with respect to the state of the art is thus to provide comparative results by testing both different feature types and feature encoding schemes, in order to identify the best feature-based strategies that can be adopted in food recognition tasks.

3. Testing framework

The pipeline of the proposed system consists of three main main phases: (i) texture feature extraction, (ii) feature encoding, and (iii) random forest classification.

Since the proposed approach introduces a classification scheme based on a learning mechanism, the system requires a training stage before its deployment. In the training stage, each image is given to the feature extraction module that extracts the texture features by means of filter banks. To reduce the high dimensionality of the obtained features, these are processed by the feature encoding module. The encoding features are finally given to the RF classifier that learns the parameters of the decision boundaries that best separate the food classes.

During the classification phase, given a test image to classify, the same types of features are extracted and nonlinear encoding procedure is applied using the learned codebook. Then, the obtained encoded features are given to the trained RF classifier that produces the final classification decision.

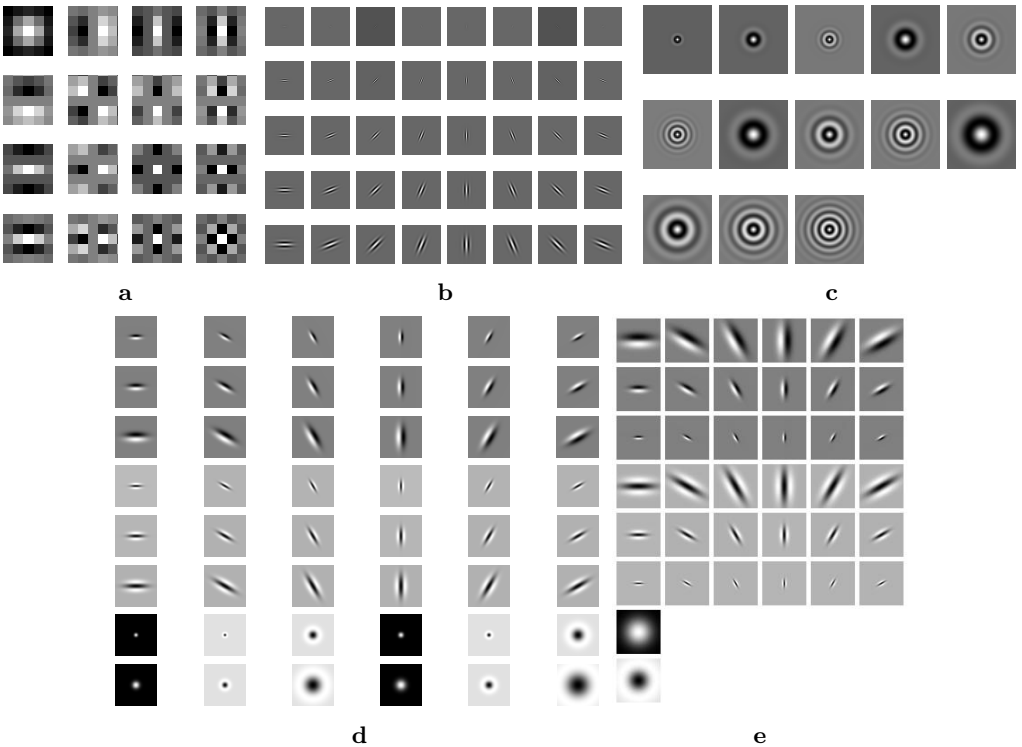


Fig. 1: (**a**) Laws filter bank consisting of 16 5x5 masks; (**b**) Gabor filter bank with 8 orientations and 5 sizes; (**c**) Standard Schmid filter bank; (**d**) Leung-Malik filter bank. The set consists of first and second derivatives of Gaussians at 6 orientations and 3 scales making a total of 36 filters; 8 Laplacian of Gaussian filters; and 4 Gaussians. (**e**) MR8 filter bank. The set consists of 2 oriented filters with 6 orientations and 3 scales plus 2 isotropic filters.

3.1. Texture Feature Extraction

Given a grayscale image $\mathbf{I} \in \mathbb{R}^{M \times N}$ of a particular food type, we have considered the most widely used bank of filters that produce features robust to object scale changes and rotations.

Laws filters The Laws filters [19] combine multiple texture energy features computed in a local neighborhood to obtain rotation invariance properties. Texture energy features are obtained by combining the 4 Laws mask into 16 2D convolutional kernels

(see Fig. 1**a**). The outputs of the 16 convolutions are then combined into 9 feature vectors which are separately vectorized, then collected in the matrix $\boldsymbol{\Upsilon}(\mathbf{I}) \in \mathbb{R}^{(MN)\times 9}$.

Gabor filters The Gabor filters [38] with 5 different sizes and 8 orientations have been used (see Fig. 1**b**). The results of the convolution process with each single filter is then vectorized. The resulting vectors computed for every Gabor filter are finally stacked to obtain $\boldsymbol{\Gamma}(\mathbf{I}) \in \mathbb{R}^{(MN)\times G}$, where $G = 40$ indicates the number of exploited Gabor filters.

Schmid filters These filters [35] are obtained by convolution with isotropic "Gabor-like" filters and have rotation invariant properties. A zero DC component of the filter is also computed to achieve invariance to intensity translations. The convolution of the image with the $S = 13$ Schmid filters (see Fig. 1**c**) results in the stacked filter responses $\boldsymbol{\Psi}(\mathbf{I}) \in \mathbb{R}^{(MN)\times 13}$.

Leung-Malik filter bank The Leung-Malik (LM) [20] filter bank has also been used to get texture features. The LM filter bank consists of first and second derivatives of Gaussians at 6 orientations and 3 scales, 8 Laplacian of Gaussian (LoG) filters, and 4 Gaussians (see Fig. 1**d**). After convolving the image with such filters the responses are collected in the feature vector $\boldsymbol{\Lambda}(\mathbf{I}) \in \mathbb{R}^{(MN)\times 48}$.

Maximum Response 8 filter bank The Maximum Response 8 (MR8) filter bank [36] has been also considered. The MR8 bank inherits from the Root Filter Set (RFS) consisting of 38 filters similar to the LM ones (see Fig. 1**e**). However, RFS filters are not rotation invariant and the MR8 ones were conceived to sidestep such a problem. This is achieved by taking only the maximum RFS filter responses across all orientations for the two anisotropic filters. In particular, measuring only the maximum response across orientations allows to reduce the number of responses from 38 (6 orientations at 3 scales for 2 oriented filters, plus 2 isotropic) to 8 (3 scales for 2 filters, plus 2 isotropic). Thus, the MR8 filter bank consists of 38 filters but only 8 filter responses which are finally collected in $\boldsymbol{\Delta}(\mathbf{I}) \in \mathbb{R}^{(MN)\times 8}$.

3.2. Feature Encoding

Feature encoding techniques are commonly exploited to reduce the dimensionality of the feature vector used to represent an image. Such methods rely on the idea that an image can be described as a composition of "visual words". Such visual words form a codebook which is commonly learned in an unsupervised fashion during a training phase. More specifically, common encoding schemes define a "visual word" as the centroid of a set of clusters into which the feature space is split.

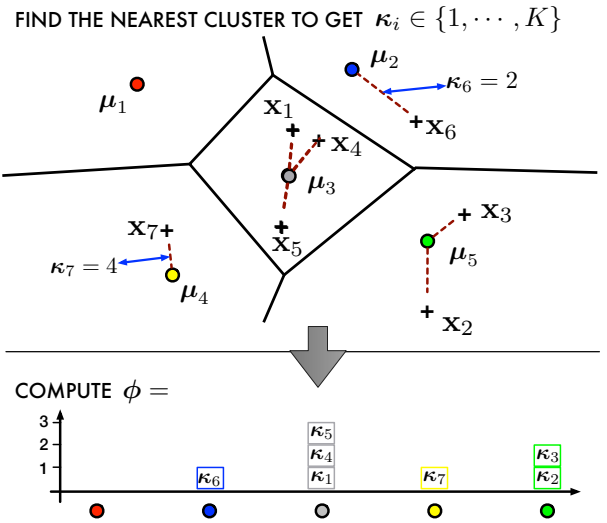


Fig. 2: Bag-of-Words encoding of feature vectors $\mathbf{x}$. First row shows the nearest cluster out of the K possible ones obtained by k-means. Second row describes the final encoded vector obtained by computing the histogram counting the frequency of each nearest cluster.

Bag-of-Words Encoding The Bag-of-Words (BoW) encoding model [8] inherits ideas from document processing techniques where a document is seen as a set of words each of which has a different frequency of appearance. In the BoW encoding for images, each image is represented by counting the frequency of a particular feature present in the learned codebook. See Fig. 2 for the BoW feature encoding scheme.

Fisher Vector Encoding The Fisher Vector (FV) encoding [12,31] defines an aggregation mechanism based on the Fisher Kernel (FK) principle by combining the benefits of generative and discriminative approaches to pattern classification. In the FV encoding scheme, clustering is performed through a Gaussian Mixture Model (GMM). Such an encoding is not limited to the number of occurrences of each visual word but it also includes additional information about the distribution of the descriptors. Recent works [31, 34] have shown that such an approach leads to better classification performances with respect to the standard BoW encoding methods.

VLAD Encoding The Vector of Locally Aggregated Descriptors (VLAD) image encoding was initially proposed by Jegou *et al.* [13] with the objective to provide an excellent search accuracy with a reasonable vector dimensionality. VLAD is a feature encoding and pooling method, similar to Fisher vectors. It encodes the feature vector

set by first constructing a dictionary of such features. This is usually done by applying a clustering method such as a Gaussian Mixture Model (GMM) or k-means on a set of features extracted from training images.

3.3. Food Classification

Let $\mathbf{I}$ be a given image of a particular food type. From such an image, all the aforementioned features are separately extracted and encoded. Once the encoding process is completed, the resulting vectors are concatenated to obtain the joint encoded representation $\boldsymbol{\Phi}(\mathbf{I}) = [\boldsymbol{\phi}(\boldsymbol{\Upsilon}(\mathbf{I}))\boldsymbol{\phi}(\boldsymbol{\Gamma}(\mathbf{I}))\boldsymbol{\phi}(\boldsymbol{\Psi}(\mathbf{I}))\boldsymbol{\phi}(\boldsymbol{\Lambda}(\mathbf{I}))\boldsymbol{\phi}(\boldsymbol{\Delta}(\mathbf{I}))] \in \mathcal{F}$, where $\boldsymbol{\phi}$ is the chosen feature encoder. Then, the goal of classification is to learn a mapping from the joint encoded feature space $\mathcal{F}$, to the label space, $\mathcal{Y}$. Where each element $y \in \mathcal{Y}$ defines a particular food type.

However, it might be that not every component in the joint encoded representation has the same discriminative power [23]. Motivated by this, and due to the multi-class classification nature of the food classification problem, we exploit a Random Forest classifier [3]. The RF classifier is able to automatically detect, and hence exploit, only the relevant features for the given task. It builds a large collection of de-correlated trees with the objective of reducing the variance of an estimated prediction function by pooling many noisy but approximately unbiased models.

To learn the parameters of the decision surfaces that separate the food types in $\mathcal{F}$ we trained an RF classifier as follows. Let $\mathcal{M} = \{\mathcal{T}_t\}_{t=1}^{T}$ be a forest of T trees each one denoted as $\mathcal{T}_t$. Each tree is trained with a set of Tr data points $\mathcal{S} = \{(\boldsymbol{\Phi}(\mathbf{I}_i), y_i)\}_{i=1}^{\mathrm{Tr}}$. Since bagging can also be applied, each tree can be trained with a different set of randomly selected data points $\mathcal{S}^* \subseteq \mathcal{S}$. The dimensionality of $\mathcal{S}^*$ is controlled through the bagging hyper-parameter η as $\mathcal{S}^* = \eta|\mathcal{S}|$. Let also $\mathcal{S}_j \subseteq \mathcal{S}^*$ be the set of data points reaching node j. Each j-th node is associated with a binary split function $h(\boldsymbol{\Phi}(\mathbf{I}), \boldsymbol{\Theta}_j) \in \{0, 1\}$, i.e., the weak learner, which is characterized by its parameters $\boldsymbol{\Theta}_j = \{\boldsymbol{\kappa}, \boldsymbol{\psi}, \boldsymbol{\tau}\}$ where $\boldsymbol{\psi}$ defines the geometric primitive used to separate the data (e.g. an axis-aligned hyperplane). The parameter vector $\boldsymbol{\tau}$ captures thresholds for the inequalities used in the binary test. The filter function $\boldsymbol{\kappa}$ randomly selects some features of choice out of the entire vector $\boldsymbol{\Phi}(\mathbf{I}_i)$. In the current framework, we select

$$h\left(\boldsymbol{\Phi}(\mathbf{I})_i, \boldsymbol{\Theta}_j\right) = \mathbb{1}_{(\langle \boldsymbol{\kappa}(\boldsymbol{\Phi}(\mathbf{I})_i), \boldsymbol{\psi} \rangle > \tau)}\,, \tag{1}$$

where $\tau = 0$ and $\boldsymbol{\psi}$ denotes a hyperplane.

All the aforementioned parameters are optimized at each split node as

$$\boldsymbol{\Theta}_j^* = \underset{\boldsymbol{\Theta}_j^* \in \mathcal{P}_j}{\arg\max}\, J(\mathcal{S}_j, \mathcal{S}_L, \mathcal{S}_R, \boldsymbol{\Theta}_j)\,, \tag{2}$$

where

$$J(\mathcal{S}, \mathbf{\Theta}) = H(\mathcal{S}) - \sum_{i=\{L,R\}} \frac{|\mathcal{S}_i|}{|\mathcal{S}|} H(\mathcal{S}_i)\,, \tag{3}$$

with Shannon entropy $H(\cdot)$ and $\mathcal{S}_L$ and $\mathcal{S}_R$ denoting the subsets of training points going to the left and to the right children, respectively. $\mathcal{P}_j \subset \mathcal{P}$ is the random subset of available parameters $\mathbf{\Theta}$. The dimensionality of such a random subset is controlled by $\rho = |\mathcal{P}_j|$. The training continues to split the samples until the maximum depth D is reached, a node contains a single sample or all the samples in the node belong to the same class.

Once the training of the whole forest is completed, a set of leaves $\mathcal{L}_t$ is associated to each tree $\mathcal{T}_t$. To each leaf $\ell_t \in \mathcal{L}_t$ is associated a probabilistic model $p_t(y|\mathbf{\Phi}(\mathbf{I}_i))$ with $y \in \mathcal{Y}$ indexing the class. Therefore, to classify a new image $\hat{\mathbf{I}}$, first its feature-encoded representation is computed. Then, for each tree, $\mathbf{\Phi}(\hat{\mathbf{I}})$ follows the path from the root node down to a leaf one. Once a leaf is reached for each tree, the RF classifier assigns to $\mathbf{\Phi}(\hat{\mathbf{I}})$ the class probability

$$p(y|\mathbf{\Phi}(\hat{\mathbf{I}})) = \frac{1}{T} \sum_{t=1}^{T} p_t(y|\mathbf{\Phi}(\hat{\mathbf{I}}))\,. \tag{4}$$

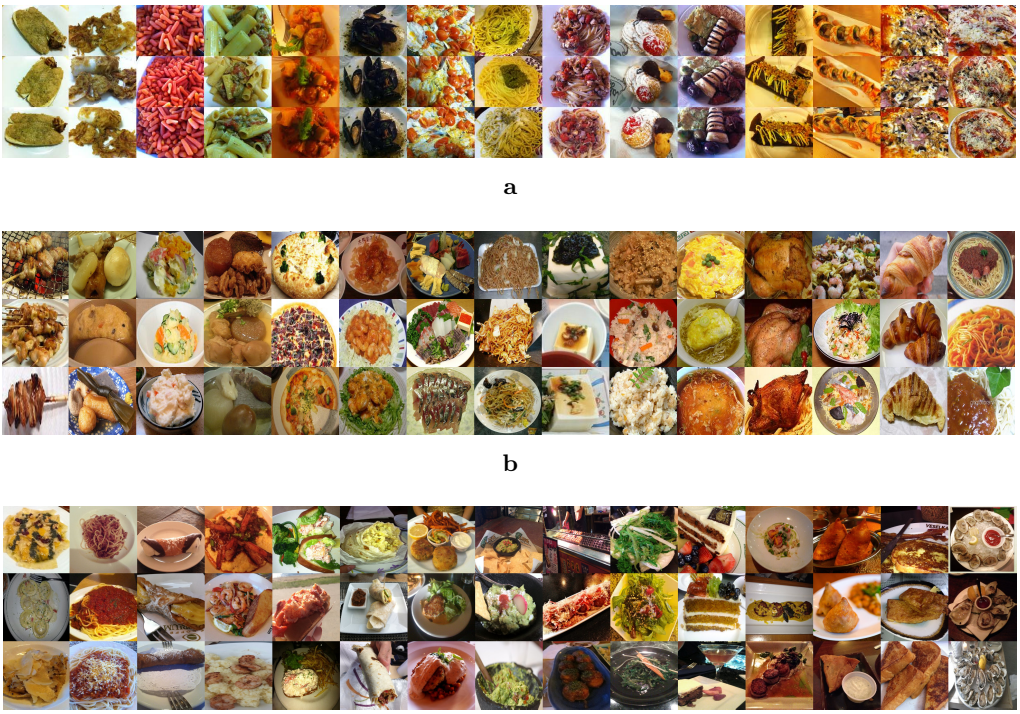
The final class label is computed as $\hat{y} = \arg\max_y p(y|\mathbf{\Phi}(\hat{\mathbf{I}}))$.

4. Experimental Results

To validate the proposed approach, results on two benchmark datasets have been computed. As commonly performed in the evaluation of food recognition approaches [5,9,10], the achieved performances are provided in terms of recognition accuracy (total number of correct classifications over total number of samples). The performance achieved by the existing methods have been taken from the corresponding works or have been provided by the authors of the present work.

4.1. Experimental Settings

In the current framework, we have used a Gabor filter bank with 8 orientations and 5 sizes, therefore $G = 40$. When not explicitly specified, the feature encoding has been obtained by means of the FV approach with $K = 300$ clusters. Similarly, by default, the RF classifier has been trained with $T = 2000$ trees having maximum depth $D = 100$. Each tree has been trained by setting $\eta = 0.6$. At each node the function κ randomly selects $\sqrt{d}$ features, where d is the dimensionality of the input feature vector.



a

b

c

Fig. 3: 15 randomly selected samples from the (**a**) UNICT-FD889, (**b**) UECFood100, and (**c**) Food101 datasets. In each figure, columns correspond to different food classes, rows show the appearance variations within the same class.

Finally, images have been rescaled to 128×128 to remove the effect of different image sizes on the classifier performances.

4.2. Datasets

UNICT-FD889 Dataset The UNICT-FD889 Dataset has been recently introduced by Farinella *et al.* [9]. The UNICT-FD889 dataset has the largest number of different classes to recognize. It comes with 3583 images related to 889 distinct dishes of food of different nationalities (e.g., Italian, English, Thai, Indian, Japanese, etc.) which have been collected in a real and uncontrolled scenario (e.g., different backgrounds and illumination conditions) by means of smartphones. Hence, the UNICT-FD889 dataset is a collection of food images acquired by users in real cases of meals. Each food belonging to

a particular class has been acquired multiple times (four on average) to ensure geometric and photometric variabilities (see Fig. 3**a** for a few examples).

To provide a fair comparison with existing methods, the following results have been computed by averaging the performance on the same three splits, as it has been advised by Farinella *et al.* [9].

UECFood100 Dataset The UECFood100 Dataset is one of the largest food recognition datasets [29]. This dataset contains approximately 100 images for each of the 100 different food categories. Thus it contains approximately 14 000 real-world food images. The UECFood100 dataset was built to implement a practical food recognition system which was intended to be used in Japan. Because of this, it was collected in such a way that multiple food items were present in a single image, thus with the objective to perform both the detection and the recognition tasks.

Since the proposed system is designed to focus only on the recognition task, the given ground truth bounding boxes have been used to obtain a dataset of images containing single food items only (see Figure 3**b**). Despite this, the same protocol proposed by Matsuda *et al.* [29] has been followed to fairly compare the obtained performance with existing methods.

Food101 Dataset

The Food-101 Dataset is the largest food recognition dataset [2]. It has been collected by downloading images from foodspotting.com. The top 101 most popular and consistently named dishes were selected. Then, for each category 750 training and 250 test images were collected and manually cleaned. On purpose, the intense colors and sometimes wrong labels included in the training images were not cleaned. As a result the dataset contains 101 000 real-world food images (see Figure 3**c**).

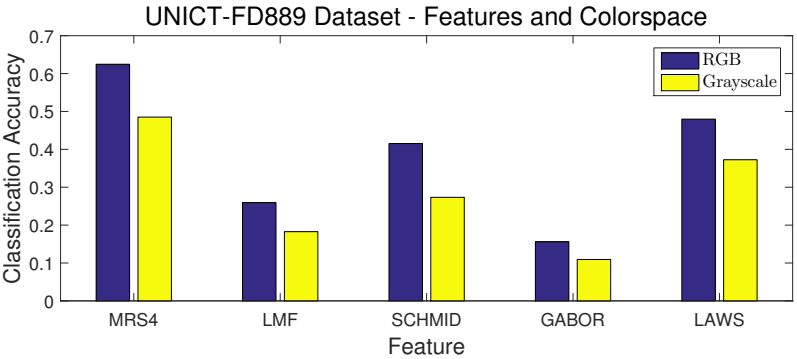
4.3. Performance Analysis

To evaluate the proposed approach, we have analyzed the following aspects: (i) performance of single filter banks using grayscale or RGB color images. In the latter case, each image plane is separately processed; (ii) performance of different encodings, also as a function of the codebook size; (iii) influence of the RF hyper-parameters on the classification accuracy.

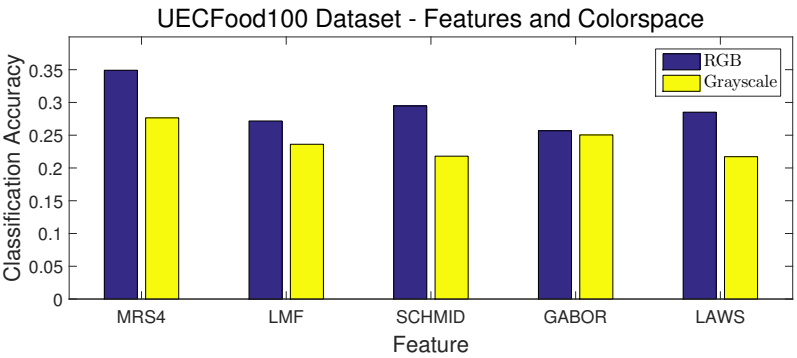
4.3.1. Filter Banks

To analyze the accuracy performance of each filter bank we have computed the results shown in Fig. 4.

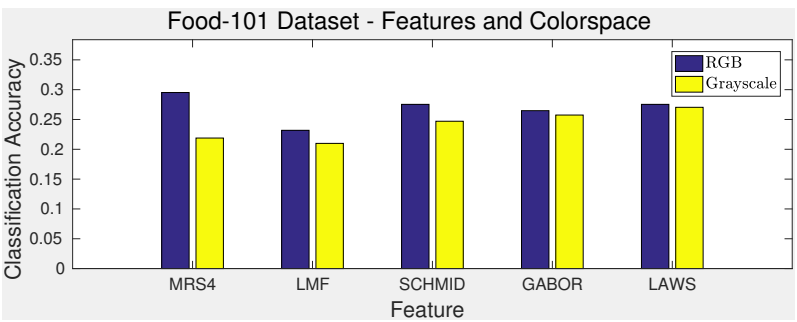
UNICT-FD889 Results in Fig. 4**a** show that, for every considered filter bank, the best performances are achieved when the full color information provided by three



a



b



c

Fig. 4: Accuracy performance on the (**a**) UNICT-FD889, (**b**) UECFood100, and (**c**) Food101 datasets achieved by filtering RGB and grayscale images with each single filter bank.

channels is exploited. In particular, when Schmid filters are used, the performance drops from 41.54% to 27.34% if only grayscale information is used. The best performances are obtained by filtering the RGB color images with the MR8 filter bank. In such a case the accuracy is 62.46%. More interestingly, results show that while being simpler compared to other filters, Laws filters yield the second best accuracy (i.e., 47.97%).

UECFood100 Results in Fig. 4**b** reflect those obtained considering the UEC-Food100 dataset. Indeed, overall, the best performances are achieved when color information is considered. However, for the considered dataset the accuracy difference between grayscale and RGB filtered images is never more than 9% (Schmid filters). The best performances are obtained by using the MR8 filter bank on RGB images. In such a case the accuracy is 34.71%. More interestingly, results show that there is less than 1% accuracy difference when Gabor filters are applied on RGB images instead of grayscale ones.

Food101 Food101 is the most complex of the three datasets, as it has a large intra-class variance. This affects the results shown in Fig. 4**c**, where the best performing filter bank (MR8) achieved just a 29% of classification accuracy. The other banks, except LMF, reached similar performances. It is worth noting that, with the exception of MR8, in the other cases working on RGB or grayscale data does not drastically affect the performances. This is probably due to the intra-class shape variance, leading to spatial features be more relevant than color information.

4.3.2. Feature Encoding

In the preceding section we have reported on the performance of each single filter bank obtained by using the FV encoding scheme. To evaluate the performance of different encodings and the influence of the codebook dimensionality, we have conducted the following experiments. Each filter bank response is encoded using the same method and the same number of "visual words" K. Then, the 5 obtained encoded vectors obtained for each image are stacked and input to the RF classifier.

In addition, to see if the RF classifier is able to select the optimal features as well as the optimal encoding, we have run one additional experiment. Specifically, the three encoding methods are separately applied to each filter bank response. The value of K has been kept same for each of them. The so obtained 15 encoded vectors are finally stacked and given to the RF classifier. In the following, results obtained with such a procedure are referred to as *All*.

Results in Fig. 5 show that for VLAD, FV and *All* optimal accuracy performance are achieved when the number of clusters K is either 20 or 50. In particular, 20 clusters are considered in the FV encoding scheme the classification accuracy is about 63.29% which is very close to the performance of the single MR8 filter bank encoded with the

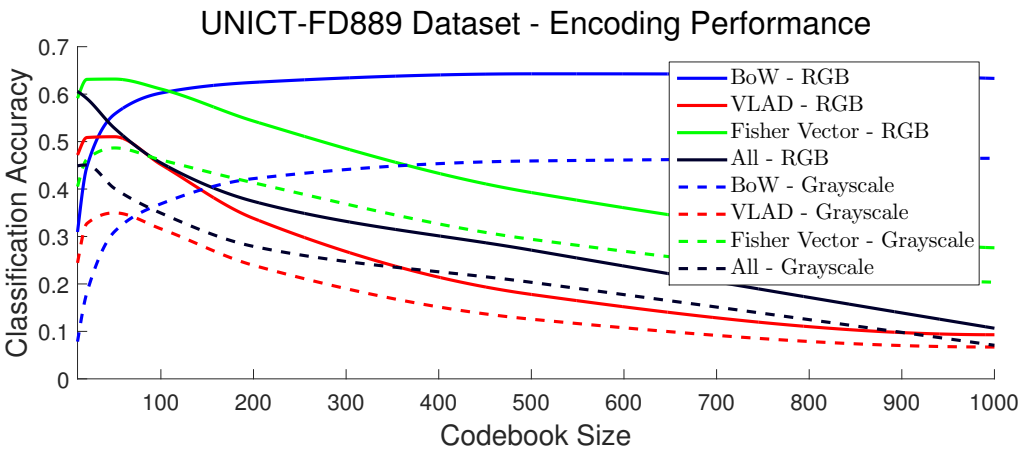


Fig. 5: Accuracy performance on the UNICT-FD889 dataset achieved by using different encoding schemes and color spaces.

same scheme but with 300 clusters (see Fig. 4**a**). For all the encodings, the performance decrease when the number of clusters increase. In particular, it is worth noticing that when the *All* solution is considered, performance are even lower than a single adopted encoding scheme. Such a behavior is mainly due to the exploding dimensionality of the encoded feature vectors which yields to a very sparse space, hence introduces the curse of dimensionality issue. Such a problem could be addressed by increasing the number of trees in the RF model.

On the other hand, performance obtained using the BoW encoding increases if a large number of visual words is used. The best accuracy under such a scheme is obtained when the number of clusters is 500. When more clusters are considered performance decrease. Differently from VLAD and FV, BoW encoding does not suffer from the curse of dimensionality problem since the number of dimensions in the encoded feature space is determined by the codebook size (see Section 3.2).

Finally, as previously shown in the filter banks analysis, better performances are achieved when RGB color information is considered.

4.3.3. Random Forest Classifier

The RF classifier model is controlled by different hyper-parameters: the number of trees T, the maximum depth D, the bagging size defined by η and the number of features

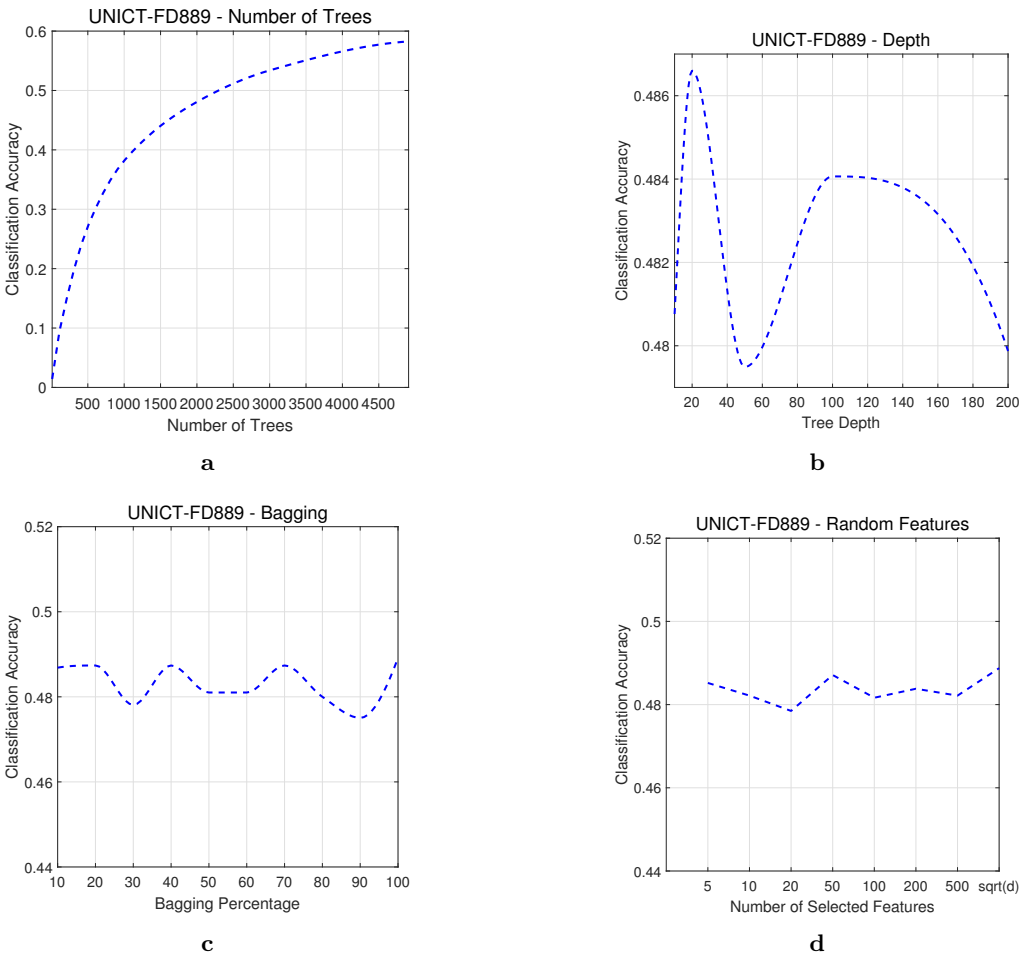


Fig. 6: Performances of the random forest classifier on the UNICT-FD889 dataset while varying the hyper-parameters. Results have been computed by starting with the hyper-parameters described in Section 4.1, and then varying, as follows: (**a**) number of trees in the forest, (**b**) maximum depth of each single tree, (**c**) percentage of features selected at each node, and (**d**) number of randomly selected features at each node.

that are selected through the filter function κ. To evaluate the influence of such hyper-parameters on the performance, we have proceeded by fixing the values of the hyper-parameters as defined in Section 4.1, then we have varied the value of a single hyper-parameter. The results of such analysis, shown in Figs. 6, 7, and 8 have been computed

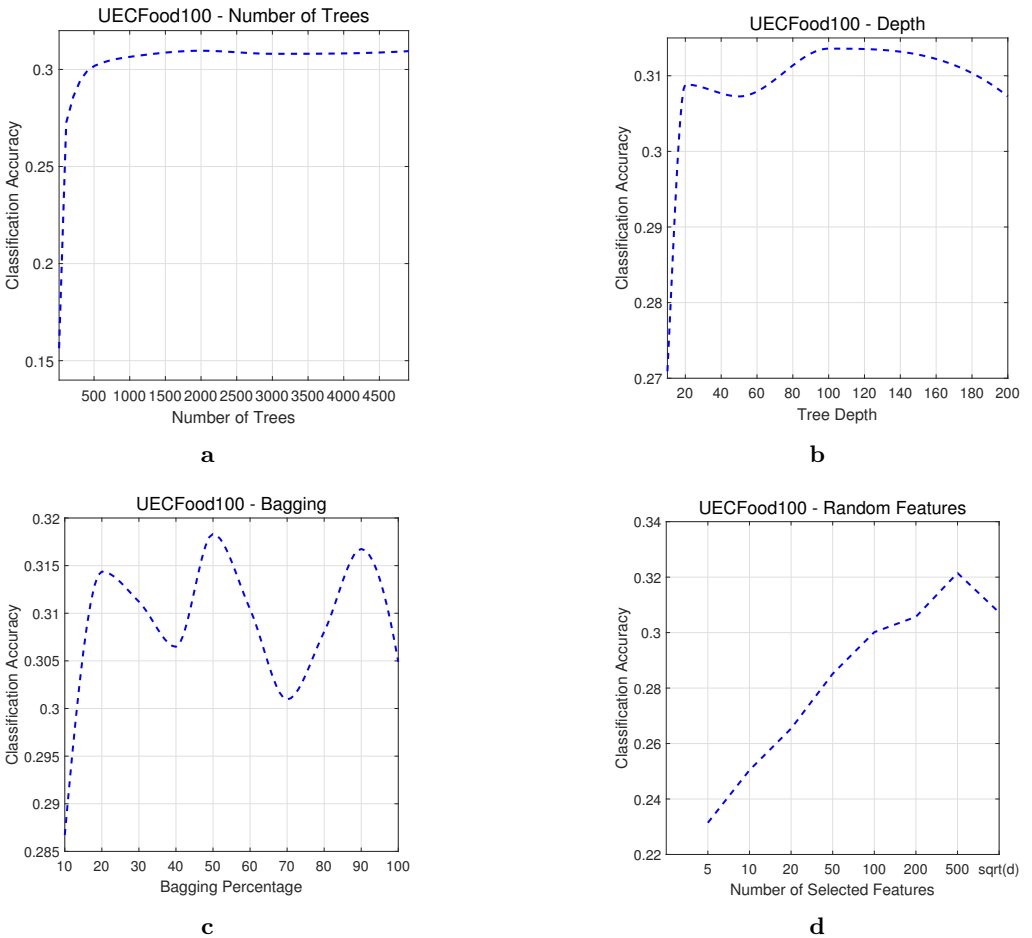
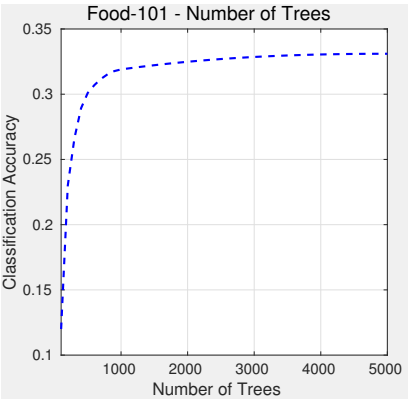
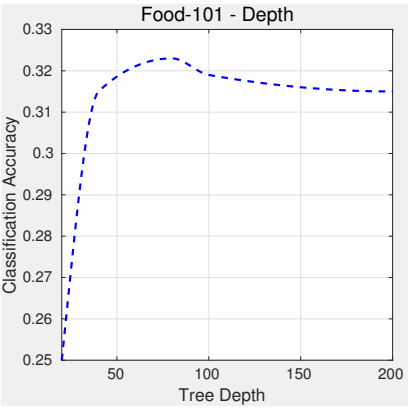


Fig. 7: Performances of the random forest classifier on the UECFood100 dataset while varying the hyper-parameters. Results have been computed by starting with the hyper-parameters described in Section 4.1, and then varying, as follows: (**a**) number of trees in the forest, (**b**) maximum depth of each single tree, (**c**) percentage of features selected at each node, and (**d**) number of randomly selected features at each node.

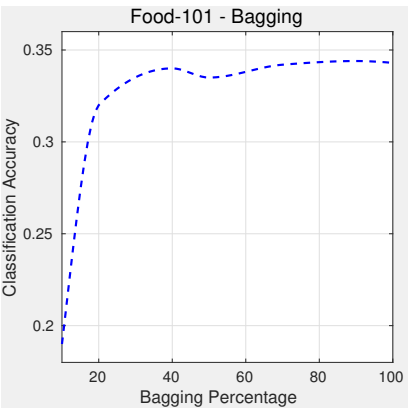
considering all the filter banks encoded using 300 clusters and the FV method. Color information has been used.



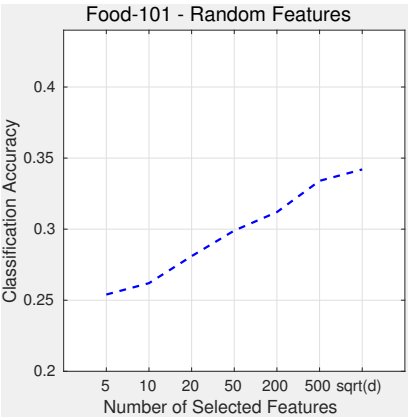
a



b



c



d

Fig. 8: Performances of the random forest classifier on the Food101 dataset while varying the hyper-parameters. Results have been computed by starting with the hyper-parameters described in Section 4.1, and then varying, as follows: (**a**) number of trees in the forest, (**b**) maximum depth of each single tree, (**c**) percentage of features selected at each node, and (**d**) number of randomly selected features at each node.

UNICT-FD889 Results in Fig. 6**a** have been computed by varying the number of trees such that $T \in [1, 5000]$. Under such scenario, the classification accuracy drastically increases when T grows from 1 to 1000. For larger values the performance improves but with less gain. The best performance is obtained when $T = 5000$ trees have been used.

Such results demonstrate the benefits of the RF classifier, and in general of aggregation methods, which are able to strongly improve the accuracy produced by a single weak classifier.

The results on the analysis of the depth hyper-parameter are shown in Fig. 6**b**. Results show that by varying D in the range $[10, 200]$ two main classification accuracy peaks can be found: the first is at $D = 20$, the second at $D = 100$. For smaller and larger values performance decreases due to underfitting and overfitting problems respectively. Indeed, with a very shallow tree it is not possible to have enough decision boundaries to discriminate all the samples well. Similarly, if the tree is too deep, the number of decision boundaries is too high, thus causing the RF model to not generalize well on new samples. The performance decrease for values in between the two peaks is caused by the two random components of the RF, i.e., bagging and the random feature selection at each node. Despite this, it should be noticed that for all the considered cases the gap between the best (48.67%) and the worst performance (47.94%) is less than 0.8%, thus showing the robustness of the model to such hyper-parameter.

Finally, in Fig. 6**c** and Fig. 6**d** the RF model performance are computed by varying the bagging percentage and the number of selected random features at each node, respectively.

The results in Fig. 6**c** show that there is consistency in the results when the bagging hyper-parameter η is changed from 0.1 to 1.0 (notice that in the latter case no bagging is used since all the available data is used by each tree). Indeed, no matter what bagging percentage is considered, the classification accuracy is never less than 47.71% ($\eta = 0.9$) nor higher than 48.86% ($\eta = 1.0$).

Similarly, results in Fig. 6**d** show that the RF model is insensitive to the number of randomly selected features at each node. Varying such a hyper-parameter the classification accuracy gap between the best and the worst performance is less than 0.9%. The optimal performance is obtained when the number of selected features is computed as $\sqrt{d}$.

UECFood100 Results in Fig. 7**a** have been computed by varying the number of trees such that $T \in [1, 5000]$. Under such scenario, the classification accuracy drastically increases when T grows from 1 to 500. For larger values the performance remains stable with a peak at $T = 2000$.

In Fig. 7**b**, results of the proposed approach are given as a function of the depth RF hyper-parameter. Differently from the results shown in Fig. 6**b**, the accuracy performance obtained using a very shallow tree (i.e., $D = 10$) is much worse than the one achieved using a deeper forest. This is due to the complexity of the dataset, which requires more separating hyperplanes to well discriminate between the 100 classes. The optimal results are achieved when $D = 100$.

Results in Fig. 7**c** show the performance of the RF model varying the bagging percentage. The depicted results demonstrate that by considering an extreme bagging in which each tree sees only 10% of the total training samples the worst performance are achieved. However, it should be noticed that in such case the accuracy performance is only about 3% less than the optimal one (obtained by setting $\eta = 0.5$).

Finally, results in Fig. 7**d** show that the RF model trained on the UECFood100 dataset is more sensitive to the selected number of randomly selected features at each node. Indeed, compared to the case when only 5 features are selected at each node, almost a 10% improvement is obtained by varying such a hyper-parameter to select 500 features.

Food101 The performances on Food101 are similar to the ones achieved on UECFood100 and confirm the conclusions described above. Figure 8**a** confirms the initial performance boost in the range $T \in [1, 500]$, even though there is a small but constant improvement when augmenting the number of trees up to 5000.

Figure 8**b** again shows very poor results when the tree depth is too low ($D < 30$), due to the complexity of the dataset, and a slight performance decrease due to overfitting when $D >= 100$. The best results have been obtained with $D = 80$.

Regarding the bagging percentage, in Figure 8**c** it can again be seen that this hyper-parameter drastically influence the results only when $\eta < 0.1$, while giving substantially stable results for any value higher than that.

Finally, the system performances with respect to the number of randomly selected features at each node has been analyzed (Figure 8**d**). As already seen in UECFood100 dataset, there is an approximately linear increase in the classification accuracy while increasing the number of selected features from 5 to $\sqrt{d}$, where the system reaches its best result.

To summarize, the RF hyper-parameter analysis has shown that the number of trees and the number of randomly selected features at each node in the forest should be carefully selected to obtain the best possible performance. On the other hand, the model performances do not heavily depend on the tree depth and the bagging percentage.

4.3.4. Discussion

As a result of the conducted performance analysis, we can draw the following conclusions:

1. While more computationally demanding color information should be retained to obtain better recognition performance.
2. The number of clusters determining the codebook size for encoding should be carefully selected on the basis of the chosen encoding scheme.
3. Exploiting more encoding schemes simultaneously may yield to performance degradation due to the very high-dimensionality of the obtained encoded feature vector.

4. For the specific tasks, the selected RF model has demonstrated to be sensitive to two hyper-parameters, namely the number of trees in the forest and the number of randomly selected features at each node. This can be exploited to simplify the cross-validation procedure usually required to select the optimal hyper-parameters values.

To qualitatively evaluate the performance of the proposed approach we have computed the results shown in Figs. 9 and 10. The performance achieved by the proposed method are shown for 18 query images from the UNICT-FD889 and the UECFood100 datasets, respectively (see caption for additional details). The depicted results demonstrate that, even only texture information is exploited, the proposed approach is able to well capture the global appearance of the images and it also has the capacity to reliably find the true match under challenging conditions (see the 3 cases in the first row). When the query image is not correctly classified, or the considered cases are very challenging, the resulting scores are very close to each other, thus meaning there is uncertainty in the given answer.

In the following section, the given results have been computed following the aforementioned conclusions. Thus, RGB color information, FV encoding with $K = 20$ clusters and $T = 5000$ trees have been considered. The other RF model hyper-parameters have been kept as defined in Section 4.1.

4.4. State-of-the-art Comparisons

Comparisons with state-of-the-art methods are shown in Fig. 11. Performances achieved by our method are labeled as TFE-RF. The reported accuracies for other methods have been taken from the papers describing the three main datasets. For this reason, the results shown in Fig. 11 are not uniform, meaning that TFE-RF is the only algorithm tested on all the three datasets.

UNICT-FD889 In Fig. 11**a**, the obtained performance are compared to 4 state-of-the-art methods, namely PRICoLBP [9], SIFT [9], BoT [9] and FB [26]. Results demonstrate that the proposed approach outperforms existing ones by improving the previous best accuracy by more than 5%. This is an interesting result since other approaches (i) require complex procedure for feature extractions (e.g., PRICoLBP [33]); (ii) use the original input images of size 320×240, which carry more information than the resized ones.

UECFood100 A comparison with state-of-the-art approaches is given in Fig. 11**b**. The obtained performance are compared to the ones achieved by 6 state-of-the-art methods [29]. Methods like Circle, JSEG, DCR, DPM, and Whole use a detector to identify the location of the food, while GTBB [29] uses the same ground truth as TFE-RF. Results demonstrate that TFE-RF outperforms detector-based approaches but has lower

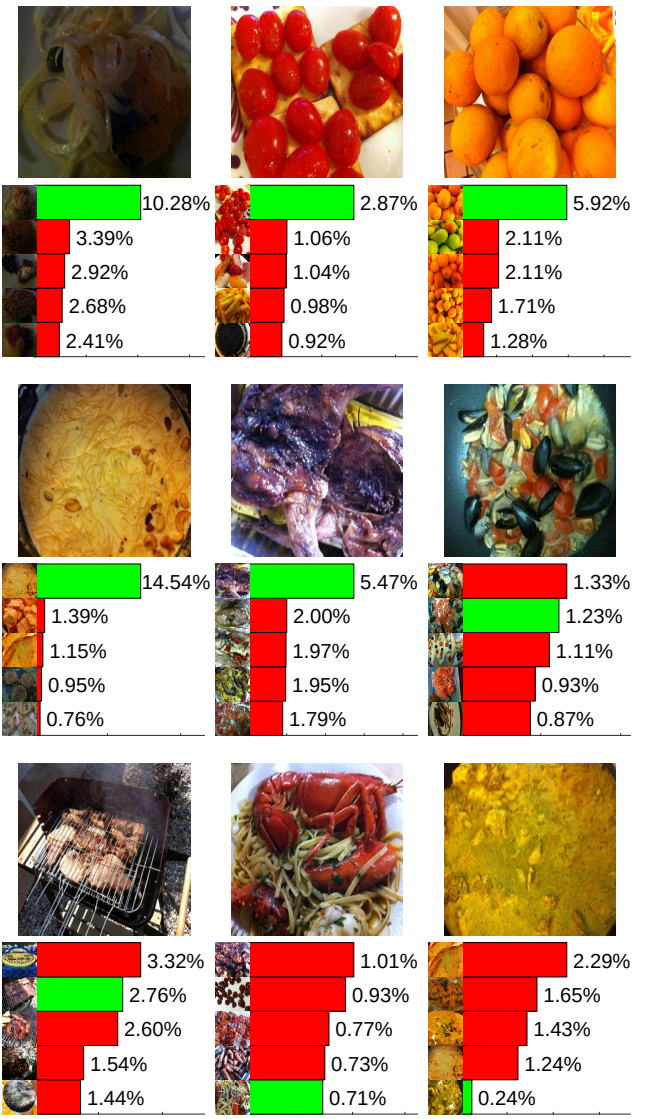


Fig. 9: Performances achieved by the proposed method on the UNICT-FD889 dataset are shown for 9 query images. At the bottom of each image, the bar histograms show the score (in percentage) of the proposed approach for the true match (in green) and for the remaining top 4 matches (in red). Beside each bar histogram, a randomly selected training image corresponding to the food class is depicted.

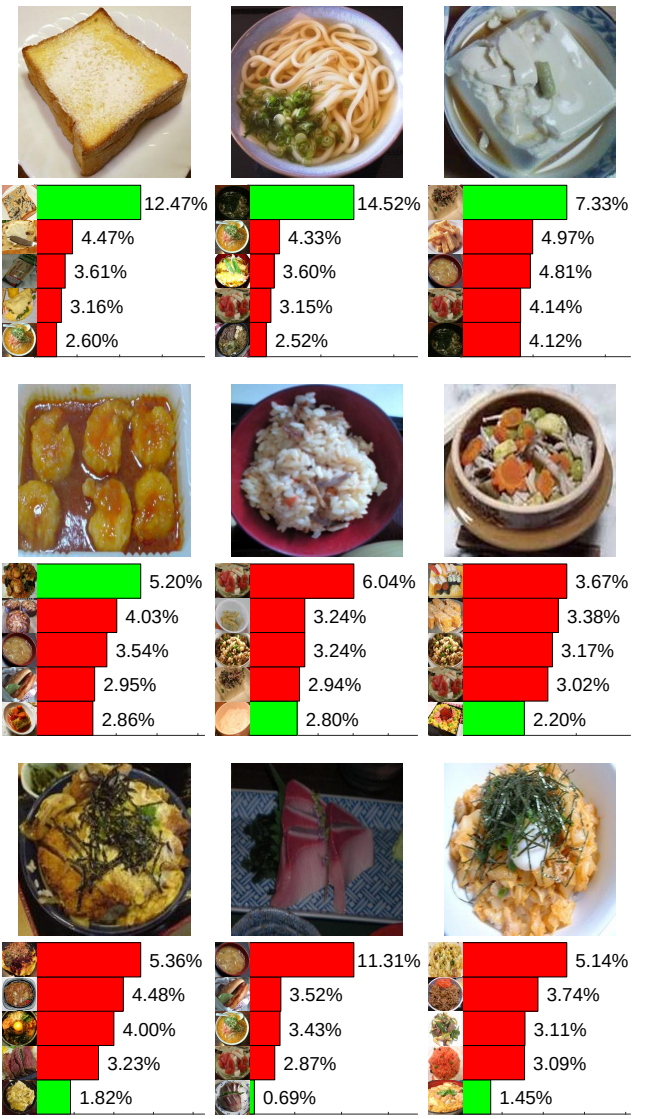
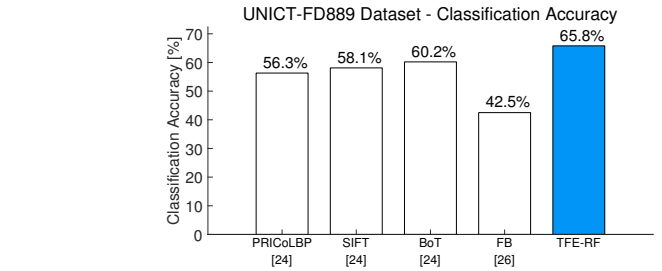
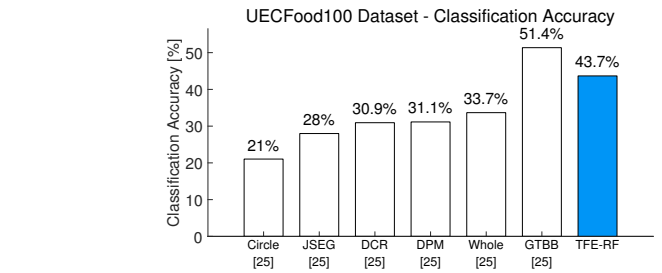


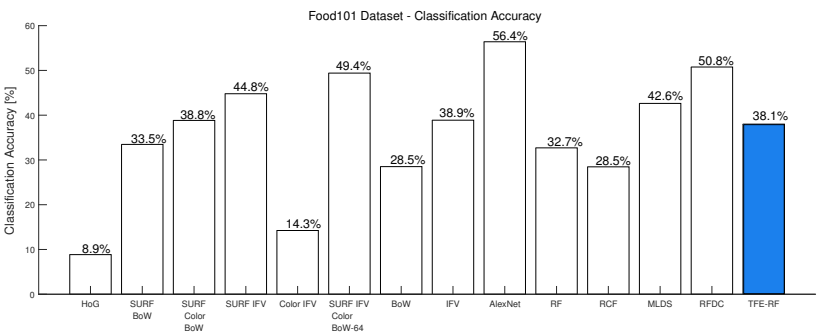
Fig. 10: Performances achieved by the proposed method on the UECFood100 dataset are shown for 9 query images. At the bottom of each image, the bar histograms show the score (in percentage) of the proposed approach for the true match (in green) and for the remaining top 4 matches (in red). Beside each bar histogram, a randomly selected training image corresponding to the food class is depicted.



a



b



c

Fig. 11: Performance of the proposed TFE-RFapproach are compared to state-of-the-art ones. Results are shown for the (**a**) UNICT-FD889, (**b**) UECFood100, and (**c**) Food101 datasets.

performance than GTBB. This is mainly due to (i) the discriminative power of the

additional color and shape features which these methods exploits [29] and (ii) the relevant information which full-size images have with respect to the resized ones which our approach considers.

Food101 Figure 11**c** shows a comparison of the proposed method with state-of-the-art approaches on the Food101 dataset. The competitors performances have been taken from [2], and include among others approaches based on HoG, BoW, SURF, AlexNet, Random Forests, etc. (see [2] for exact references for each method). As it can be seen, in this challenging dataset the proposed method reaches an accuracy of 38.1%, comparable to the other approaches based on random forests, but outperformed by the results obtained by deep learning strategies, as in the case of AlexNet, based on a Convolutional Neural Network approach.

5. Conclusions

In this paper an analysis of existing filter banks and feature encoding schemes for food classification has been provided. The work has been motivated by the lack of studies showing which filters for texture features are likely to be the most useful for the task. In particular, we have considered five of the most widely used filter banks in computer vision, namely Laws, Gabor, Schmid, Leung-Malik and MR8. To reduce the high dimensional representations produced by filter banks we have used and analyzed the performance of the three main encoding schemes in literature: BoW, FV and VLAD. Then, the food classification task is accomplished by exploiting such encoded representations within the RF classifier. Results on three food classification benchmark datasets have been provided. Specifically, an in-depth analysis of the performance of each single filter bank and encoding scheme has been provided. The proposed method has comparable performances with respect to state-of-the-art approaches. It is worth noting that all the tested methods have a moderate accuracy (maximum reached is 65.8% on the UNICT-FD889 dataset), confirming that food recognition is a complex task. These results could also be a consequence of explicit feature descriptors having sub-optimal performances in this specific application field. For this reason, as a future work, we will investigate novel architectures based on feature learning methods [37] to reduce the required on-board computational costs and to improve the accuracy.

References

[1] 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, 2014. doi:10.1109/JBHI.2014.2308928.

[2] L. Bossard, M. Guillaumin, and L. Van Gool. Food-101 – Mining discriminative components with random forests. In *Proc. 13th European Conf. Computer Vision ECCV 2014*, Part VI, volume

8694 of *Lecture Notes in Computer Science*, pages 446–461, Zurich, Switzerland, Sep 6-12, 2014. doi:10.1007/978-3-319-10599-4_29.

[3] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001. doi:10.1023/A:1010933404324.

[4] M. C. Carter, V. J. Burley, C. Nykjaer, and J. E. Cade. Adherence to a smartphone application for weight loss compared to website and paper diary: pilot randomized controlled trial. *Journal of Medical Internet Research*, 15(4):e32, 2013. doi:10.2196/jmir.2283.

[5] M. Chen, K. Dhingra, W. Wu, L. Yang, R. Sukthankar, and J. Yang. PFID: Pittsburgh fast-food image dataset. In *Proc. 16th IEEE Int. Conf. Image Processing ICIP 2009*, pages 289–292, Cairo, Egypt, Nov 7-10, 2009. doi:10.1109/ICIP.2009.5413511.

[6] M.-Y. Chen, Y.-H. Yang, C.-J. Ho, et al. Automatic Chinese food identification and quantity estimation. In *SIGGRAPH Asia SA 2012 Technical Briefs*, pages 29:1–29:4, Singapore, Singapore, Nov 28-Dec 1, 2012. ACM. doi:10.1145/2407746.2407775.

[7] S. Christodoulidis, M. Anthimopoulos, and S. Mougiakakou. Food recognition for dietary assessment using deep convolutional neural networks. In *Proc. New Trends in Image Analysis and Processing – ICIAP 2015 Workshops*, volume 9281 of *Lecture Notes in Computer Science*, pages 458–465, Genoa, Italy, Sep 7-8, 2015. doi:10.1007/978-3-319-23222-5_56.

[8] G. Csurka, C. R. Dance, L. Fan, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints. In *Workshop on Statistical Learning in Computer Vision SLCV 2004, European Conference on Computer Vision ECCV 2004*, pages 1–22, Prague, Czech Republic, May 15, 2004. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.72.604.

[9] G. M. Farinella, D. Allegra, and F. Stanco. A benchmark dataset to study the representation of food images. In *Proc. European Conf. Computer Vision ECCV 2014 Workshops*, Part III, pages 584–599, Zurich, Switzerland, Sep 6-7 and 12, 2014 (published in 2015). doi:10.1007/978-3-319-16199-0_41.

[10] G. M. Farinella, M. Moltisanti, and S. Battiato. Classifying food images represented as Bag of Textons. In *Proc. IEEE Int. Conf. Image Processing ICIP 2014*, pages 5212–5216, Paris, France, Oct 27-30, 2014. doi:10.1109/ICIP.2014.7026055.

[11] J. Garcia, N. Martinel, A. Gardel, I. Bravo, G. L. Foresti, and C. Micheloni. Discriminant context information analysis for post-ranking person re-identification. *IEEE Transactions on Image Processing*, 26(4):1650–1665, 2017. doi:10.1109/TIP.2017.2652725.

[12] T.S. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. In *Proc. 11th Int. Conf. Neural Information Processing Systems*, pages 487–493, Devner, USA, Dec 1-3, 1999. http://dl.acm.org/citation.cfm?id=3009055.3009124.

[13] H. Jégou, M. Douze, C. Schmid, and P. Pérez. Aggregating local descriptors into a compact image representation. In *Proc. IEEE Computer Society Conf. Computer Vision and Pattern Recognition CVPR 2010*, pages 3304–3311, San Francisco, USA, Jun 13-18, 2010. doi:10.1109/CVPR.2010.5540039.

[14] H. Kagaya, K. Aizawa, and M. Ogawa. Food detection and recognition using convolutional neural network. In *Proc. 22Nd ACM Int. Conf. Multimedia MM 2014*, pages 1085–1088, New York, NY, USA, Nov 3-7, 2014. ACM. doi:10.1145/2647868.2654970.

[15] Y. Kawano and K. Yanai. Real-time mobile food recognition system. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition CVPR 2013 Workshops*, pages 1–7, Portland, USA, Jun 23-28, 2013. doi:10.1109/CVPRW.2013.5.

[16] Y. Kawano and K. Yanai. Food image recognition with deep convolutional features. In *Proc. 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication, UbiComp 2014 Adjunct*, pages 589–593, New York, NY, USA, 2014. ACM. doi:10.1145/2638728.2641339.

[17] Y. Kawano and K. Yanai. Foodcam: A real-time food recognition system on a smartphone. *Multimedia Tools and Applications*, 74(14):5263–5287, 2015. doi:10.1007/s11042-014-2000-8.

[18] F. Kong and J. 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.

[19] K. I. Laws. Rapid texture identification. In *Proc. 24th Ann. Tech. Symp. Image Processing for Missile Guidance*, volume 0238 of *Proc. SPIE*, pages 376–380, San Diego, USA, Jul 29-Aug 1, 1980. doi:10.1117/12.959169.

[20] T. Leung and J. Malik. Representing and recognizing the visual appearance of materials using three-dimensional textons. *International Journal of Computer Vision*, 43(1):29–44, 2001. doi:10.1023/A:1011126920638.

[21] N. Martinel and C. Micheloni. Sparse matching of random patches for person re-identification. In *Proc. Int. Conf. Distributed Smart Cameras ICDSC 2014*, pages 8:1–8:6, Venezia Mestre, Italy, Nov 4-7, 2014. doi:10.1145/2659021.2659034.

[22] N. Martinel, C. Micheloni, and G. L. Foresti. Robust painting recognition and registration for mobile augmented reality. *IEEE Signal Processing Letters*, 20(11):1022–1025, 2013. doi:10.1109/LSP.2013.2279014.

[23] N. Martinel, C. Micheloni, and C. Piciarelli. Learning pairwise feature dissimilarities for person re-identification. In *Proc. 7th Int. Conf. Distributed Smart Cameras ICDSC 2013*, pages 1–6, Hong Kong, China, Oct 29-Nov 1, 2013. doi:10.1109/ICDSC.2013.6778209.

[24] N. Martinel, C. Piciarelli, G.L. Foresti, and C. Micheloni. Mobile Food Recognition with an Extreme Deep Tree. In *Proc. Int. Conf. Distributed Smart Cameras*, pages 56–61, Paris, France, 2016. doi:10.1145/2967413.2967428.

[25] N. Martinel, C. Piciarelli, and C. Micheloni. A supervised extreme learning committee for food recognition. *Computer Vision and Image Understanding*, 148:67–86, 2016. doi:10.1016/j.cviu.2016.01.012.

[26] N. Martinel, C. Piciarelli, C. Micheloni, and G. L. Foresti. On filter banks of texture features for mobile food classification. In *Proc. 9th Int. Conf. Distributed Smart Cameras ICDSC 2015*, pages 14–19, Seville, Spain, Sep 8-11, 2015. ACM. doi:10.1145/2789116.2789132.

[27] N. Martinel, C. Piciarelli, C. Micheloni, and G. L. Foresti. A structured committee for food recognition. In *Proc. IEEE Int. Conf. Computer Vision Workshop ICCVW 2015*, pages 484–492, Santiago, Chile, Dec 9-13, 2015. doi:10.1109/ICCVW.2015.70.

[28] Niki Martinel, Gian Luca Foresti, and Christian Micheloni. Wide-Slice Residual Networks for Food Recognition. In *Proc. Winter Conference on Applications of Computer Vision*, 2018. In Press.

[29] Y. Matsuda, H. Hoashi, and K. Yanai. Recognition of multiple-food images by detecting candidate regions. In *Proc. Int. Conf. Multimedia and Expo*, pages 25–30, Melbourne, Australia, Jul 9-13, 2012. doi:10.1109/ICME.2012.157.

[30] Y. Matsuda and K. Yanai. Multiple-food recognition considering co-occurrence employing manifold ranking. In *Proc. 21st Int. Conf. Pattern Recognition ICPR 2012*, pages 2017–2020, Tsukuba, Japan, Nov 11-15, 2013. https://ieeexplore.ieee.org/document/6460555.

[31] F. Perronnin, J. Sánchez, and T. Mensink. Improving the Fisher kernel for large-scale image classification. In *Proc. European Conf. Computer Vision ECCV 2010*, volume 6314 of *Lecture Notes in Computer Science*, pages 143–156, Heraklion, Crete, Greece, Sep 5-11, 2010. doi:10.1007/978-3-642-15561-1_11.

[32] M. Puri, Z. Zhu, Q. Yu, A. Divakaran, and H. Sawhney. Recognition and volume estimation of food intake using a mobile device. In *Proc. Workshop on Applications of Computer Vision WACV 2019*, pages 1–8, Dec 7-8, 2009. doi:10.1109/WACV.2009.5403087.

[33] X. Qi, R. Xiao, C. Li, Y. Qiao, J. Guo, and X. Tang. Pairwise rotation invariant co-occurrence local binary pattern. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 36(11):2199–2213, 2014. doi:10.1109/TPAMI.2014.2316826.

[34] J. Sanchez, F. Perronnin, T. Mensink, and J. Verbeek. Image classification with the Fisher vector: Theory and practice. *International Journal of Computer Vision*, 105(3):222–245, 2013. doi:10.1007/s11263-013-0636-x.

[35] C. Schmid. Constructing models for content-based image retrieval. In *Proc. IEEE Computer Society Conf. Computer Vision and Pattern Recognition CVPR 2001*, volume 2, pages II-39–II-45, Kauai, Hawaii, USA, Dec 8-14, 2001. doi:10.1109/CVPR.2001.990922.

[36] M. Varma and A. Zisserman. Texture classification: are filter banks necessary? In *Proc. IEEE Computer Society Conf. Computer Vision and Pattern Recognition CVPR 2003*, volume 2, pages II-691–II-698, Madison, USA, Jun 18-20, 2003. doi:10.1109/CVPR.2003.1211534.

[37] M. Vernier, N. Martinel, C. Micheloni, and G. L. Foresti. Remote feature learning for mobile re-identification. In *Proc. 7th Int. Conf. Distributed Smart Cameras ICDSC 2013*, pages 1–6, Hong Kong, China, Oct 29-Nov 1, 2013. doi:10.1109/ICDSC.2013.6778221.

[38] T.P. Weldon, W.E. Higgins, and D.F. Dunn. Efficient Gabor filter design for texture segmentation. *Pattern Recognition*, 29(12):2005–2015, 1996. doi:10.1016/S0031-3203(96)00047-7.

[39] World Health Organization. Obesity and overweight – fact sheet n. 311, 2015. http://www.who.int/mediacentre/factsheets/fs311/en/.

[40] K. Yanai and Y. Kawano. Food image recognition using deep convolutional network with pre-training and fine-tuning. In *Proc. IEEE Int. Conf. Multimedia & Expo Workshops ICMEW 2015*, pages 1–6, Turin, Italy, Jun 29-Jul 3, 2015. doi:10.1109/ICMEW.2015.7169816.

[41] S. Yang, M. Chen, D. Pomerleau, and R. Sukthankar. Food recognition using statistics of pairwise local features. In *Proc. IEEE Computer Society Conf. Computer Vision and Pattern Recognition CVPR 2010*, pages 2249–2256, San Francisco, USA, Jun 13-18, 2010. doi:10.1109/CVPR.2010.5539907.

[42] W. Zhang, Q. Yu, B. Siddiquie, A. Divakaran, and H. Sawhney. "Snap-n-Eat": Food recognition and nutrition estimation on a smartphone. *Journal of Diabetes Science and Technology*, 9(3):525–533, 2015. doi:10.1177/1932296815582222.

[43] F. Zhu, M. Bosch, I. Woo, et al. The use of mobile devices in aiding dietary assessment and evaluation. *IEEE Journal of Selected Topics in Signal Processing*, 4(4):756–766, 2010. doi:10.1109/JSTSP.2010.2051471.