The Visual Computer
https://doi.org/10.1007/s00371-023-03119-y

ORIGINAL ARTICLE

Making paper labels smart for augmented wine recognition

Alessia Angeli[1] · Lorenzo Stacchio[2] · Lorenzo Donatiello[1] · Alessandro Giacchè[1] · Gustavo Marfia[3]

Accepted: 9 September 2023

Abstract
An invisible layer of knowledge is progressively growing with the emergence of situated visualizations and reality-based information retrieval systems. In essence, digital content will overlap with real-world entities, eventually providing insights into the surrounding environment and useful information for the user. The implementation of such a vision may appear close, but many subtle details separate us from its fulfillment. This kind of implementation, as the overlap between rendered virtual annotations and the camera's real-world view, requires different computer vision paradigms for object recognition and tracking which often require high computing power and large-scale datasets of images. Nevertheless, these resources are not always available, and in some specific domains, the lack of an appropriate reference dataset could be disruptive for a considered task. In this particular scenario, we here consider the problem of wine recognition to support an augmented reading of their labels. In fact, images of wine bottle labels may not be available as wineries periodically change their designs, product information regulations may vary, and specific bottles may be rare, making the label recognition process hard or even impossible. In this work, we present augmented wine recognition, an augmented reality system that exploits optical character recognition paradigms to interpret and exploit the text within a wine label, without requiring any reference image. Our experiments show that such a framework can overcome the limitations posed by image retrieval-based systems while exhibiting a comparable performance.

Keywords Augmented reality · Augmented wine recognition · Ubiquitous computing · Artificial intelligence · Optical character recognition

1 Introduction

Situated visualizations (SV) and reality-based information retrieval systems aim at superimposing context-based digital information to real-world entities, such as food, people, buildings, and photographs [1–7]. Such concepts root in the seminal work of Fitzmaurice, who introduced the idea of situated information spaces, i.e., spaces where digital information associated with physical objects is collected, associated, and collocated with those objects which act as anchors [8]. Considering the advancements and affordability of augmented reality (AR), SV becomes viable in several domains, providing information regarding physical objects, and chaperoning a user through a specific process (e.g., learning, making a choice).

In the food and beverage sector, for example, many initiatives have worked on scanning packages to visualize augmented information related to its contents (e.g., nutritional information, reviews) [9–14]. Such applications are built on top of marker or marker-less computer vision approaches for object detection, recognition, and tracking [15], which often require high computational power and the construction of large-scale datasets of reference images. Such approaches, however, may hardly scale with long-tailed products [16] as, for example, wines [17, 18]. In fact, in many cases, it may be difficult to acquire wine labels beforehand making them impossible to be recognized. For example, a winery may have changed its labels or stopped the production of specific wines or there could be a shortage of pictures of old wines labels. It is very challenging to create a complete database covering the present and the past of wine labels.

Alessia Angeli and Lorenzo Stacchio: joint first authors.

✉ Gustavo Marfia
gustavo.marfia@unibo.it

[1] Department of Computer Science and Engineering, University of Bologna, Bologna, Italy

[2] Department for Life Quality Studies, University of Bologna, Rimini, Italy

[3] Department of the Arts, University of Bologna, Bologna, Italy

Published online: 27 October 2023

Considering this scenario, optical character recognition (OCR)-based techniques [19, 20] may represent a solution. In fact, given a wine bottle label, identifying and recognizing it by exploiting the text within, could allow us to overcome all the aforementioned limits. Nevertheless, this approach is practically challenging because of the label visual features, such as complex backgrounds, text of different fonts, and distortions due to the curved surface of bottles, which could mislead the OCR predictions. Even when perfectly recognizing text, an automatic process should be able to focus on only those words that can be used as cues to identify a wine. This step, alone, could require an amount of time not compatible with the implementation of a situated visualization. Therefore, such an approach requires careful integration of methods, algorithms, and technologies to produce an effective system in terms of detection efficacy and efficiency. To this aim, we designed and implemented augmented wine recognition (AWR), an augmented reality (AR) system to automatically recognize a wine type by recognizing and analyzing the text within its corresponding bottle back label. In particular, we tailored AWR on the Italian wine domain knowledge, considering that Italy is a wine top producer and follows European regulations regarding label organization and design [18, 21]. Leveraging on the information reported on such labels due to the European regulations, we modeled a custom tree data structure defining a hierarchy of textual features that discriminates between different types of wine. Moreover, we defined a custom hierarchical search algorithm that explores such structure by matching and branching the tree itself, using as criteria the distance between a sought feature and the words found on a label. The process ends when the best wine type candidates are found. The AWR system [22][1] has been assessed employing a textual database of 2,426 wines belonging to the Italian Emilia–Romagna region (provided by ImageLine S.r.l.). AWR exhibited a performance of 91% in terms of recognition accuracy and an average of 2.37 seconds in terms of inference time, showing that this system may be acceptable from a user perspective. We also compared our solution to a naive solution based on data structures built ignoring wine domain knowledge, showing that AWR is able to drop by two orders of magnitude the wine type recognition time.

With respect to [22], this work expands the description of the key factors that have led to achieving such performance: the required wine domain knowledge and the use of an approach integrating OCR and database search strategies. We will hence show how our approach is able to scale in domains characterized by long tails [16] without requiring high computational power and large-scale datasets of reference images.

Summarizing, the contributions of this work amount to:

- An analysis of the state of the art and a comparison of the AWR system to existing approaches;
- A detailed presentation of the domain knowledge relevant for the design of AWR;
- An in-depth description of the label recognition pipeline;
- A thorough discussion opening possible future improvements and directions of work.

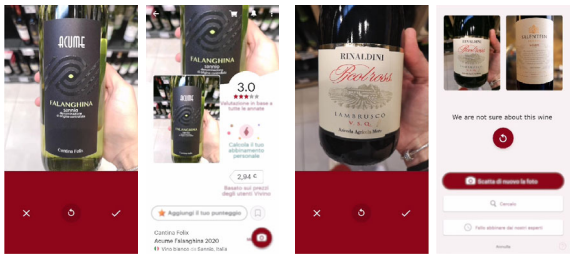
The rest of the paper is organized as follows: Sects. 2 and 3 describe related works and wine theory background. Subsequently, the proposed framework and its components (i.e., textual database, OCR, search algorithm) are detailed in Sect. 4. Section 5 reports experimental results obtained by exploiting our approach with the considered dataset, from both an efficacy and efficiency perspective. Finally, the limitations and possible future directions are discussed in Sect. 6.

2 Related work

Food and beverage product identification is often performed with bar codes or QR codes. Many AR applications exploit image detection and recognition paradigms, which may also be based on codes or on the recognition of a product as it appears [11, 15, 23, 24]. In fact, identifying and exploiting visual cues in food product pictures is an approach that appeared in various research contributions [25–28]. In this scenario, some works, from both industrial and academic contexts, focused on wine label recognition and its application in the AR realm [13, 29–38].

Regarding commercial solutions, *WineEngine* is an online wine label recognition service [29] which exploits a combination of OCRs and image retrieval-based approaches using the wine bottle front label. This approach requires adding reference label images to the considered database and does not provide an AR interface. Another interesting system to recognize wine bottles is *Living Wine Labels* [30]. It also uses image retrieval to recognize the front label of a particular wine bottle and subsequently present customized AR animations. Mostly used for storytelling purposes, it supports eleven brands and requires a database of images for each of the different front-label bottles. Finally, Vivino is the most downloaded app with a community comprising 20 million users around the globe [13, 31] and provides features such as wine exploration, evaluations, and a wine bottle front-label recognizing service. Vivino does not provide an AR interface and implements an image retrieval approach based on the Vuforia Cloud Recognition service that compares incoming front-label scans uploaded by a user to the ones stored in a custom database, to discover the closest match [31]. Given the high number of downloads and its large community, and considering that the approach it adopts is entirely based on image retrieval, we performed a simple experiment

[1] An online demo visualization is accessible here.



Correctly recognized. Not recognized.

Fig. 1 Examples of wine recognition with Vivino

with Vivino to assess its performance in an everyday life scenario. We hence visited a local supermarket and tested its performance with 60 bottles of wine: 47 were correctly recognized (78% accuracy) taking an average time of 2.05 seconds (with a standard deviation of 0.65). Examples of wine labels correctly/wrong detected are reported in Fig. 1.

Considering now academic contributions, they mostly have followed image retrieval-based approaches [32–38]. In Gebru et al. [32], the authors implemented a front-label recognition method computing SURF key points and label descriptors and comparing such descriptors to precomputed ones in a label database to search for a match. Similar approaches can be found in Wu et al. [34] and Jung et al. [36]. In Li et al. [38] the authors proposed a CNN-SIFT framework for wine label retrieval, where a trained CNN model recognizes the wine producer to narrow the search range, while a SIFT descriptor empowered with RANSAC and TF-IDF mechanisms matches the final sub-brand. In Álvarez Márquez and Ziegler [37], the authors presented an AR system running on a Microsoft HoloLens, making use of the Vuforia SDK to recognize markers attached to wine bottles and to display information concerning those bottles [39]. It is also possible to find other approaches in literature that concentrate on recognition subproblems. In Na et al. [33], for example, the authors concentrated on a preliminary step, a region of interest extraction method (GrabCut algorithm) for front labels, that may serve subsequent ones such as image analysis, recognition, and retrieval. All of the aforementioned academic contributions rely on image retrieval-based approaches, and so present the main limit of requiring an extensive image database, which may be very difficult if not impossible considering old, out-of-production, or new wine types (i.e, long-tailed samples).

Differently, [35] implemented an OCR-based solution to read serial numbers from wine labels to provide counterfeit prevention and brand protection. However, this would be required to have access to all the correspondences between serial numbers and related bottle wine types.

Differently from the presented related works, AWR entirely relies on text, as it solely employs an OCR and a custom database to recognize a wine type from the text reported on the back label of a bottle. Table 1 compares the characteristics of our solution against existing ones, where IR stands for image retrieval, AR for augmented reality, AReT for almost real time, OCR indicates the usage of an OCR, TDO for textual database only, and LoTE for long-tailed extensible.

Table 1 Comparison between the characteristics of the different wine recognition systems and AWR

References	IR	AR	AReT	OCR	TDO	LoTE
[29]	✓	×	✓	✓	×	×
[30]	✓	✓	×	×	×	×
[13]	✓	×	×	×	×	×
[32–34, 36, 38]	✓	×	×	×	×	×
[37]	✓	✓	✓	×	×	×
[35]	×	×	✓	✓	✓	×
AWR	×	✓	✓	✓	✓	✓

3 Wine background

A wine bottle usually includes two labels: a front and a back one. The front one is often devoted to brand communication, whereas the back one reports all the information characterizing a given wine, displayed according to its home country regulations [21]. It is also possible to find wines bearing a single label: such a label will include all the required information in a compressed form. From now on, we will use the term "label" to indicate the ones which contain information to discriminate a wine types, as required by Italian regulations.

Recent work has provided a valuable description of the historical development of wine policies in Europe inside the Common Market Organization (CMO) [18]. According to Italian regulations, specific information (e.g., wine appellation, winery) must appear on a wine bottle in the same field of view (i.e., a consumer should not have to turn a bottle to read them all). Italian labels report different information, some mandatory and some not [40–43]. A list of the most important ones is provided in the following.

- *Name* the wine name, typically found at the label top center.
- *Type* wine, varietal wine, appellation wine. In the case of appellation wine, this is related to the geographical area of production.
- *Appellation* The appellation wines may fall into two sub-categories: Protected Geographical Indication (PGI) and Protected Designation of Origin (PDO). Italian PDO wines can be DOC or DOCG, now both included in DOP. PGI wines can be IGT, now included in IGP. DOP and IGP appear after the 2008 CMO reform. DOP (Euro-

pean category) indicates products whose characteristics depend on a specific geographical environment where all the production phases should occur. The IGP European category designates products whose characteristics depend on specific geographical areas where, at least, one of the production phases must take place. We report below the meaning of the following Italian acronyms, translated into English for more clarity: Designation of Origin Controlled (DOC), Designation of Origin Controlled and Guaranteed (DOCG), Protected Designation of Origin (DOP), Typical Geographical Indication (IGT), and Protected Geographical Indication (IGP). With the appellation wines, this information is mandatory, but it is possible to choose whether to indicate the appellation (i.e., DOC, DOCG, IGT) or the corresponding European category (i.e., DOP, IGP).

- *Appellation value* In addition, the appellation value is the "proper name" of the class and it is unique for each wine types (e.g., Pignoletto and Romagna are the concrete appellation Values of wines categorized as DOC). The label should report the appellation field nearby the appellation value.
- *Winemaker/winery* The name of the winery where a wine is bottled should always appear on the label. A winery may work for multiple labels/brands.
- *Region of origin* not required. With appellation wines, this information can be inferred from the appellation value. Otherwise, it can be found once the winery has been identified.
- *Origin trademark* A wine does not necessarily have a trademark of origin. If present, some examples are Quality Sparkling Wine Produced in a specific region (VSQPRD), Quality Aromatic Sparkling Wine Produced in a specific region (VSAQPRD), and Aromatic Sparkling Wine (VSA).
- *Effervescence* still, sparkling, spumante. This information could appear with synonyms also: stationary, moved, etc. If nothing is specified the wine is assumed to be still.
- *Sweetness* The terms change according to the wine effervescence. For still and sparkling wines, *Secco*, *Semisecco*, *Abboccato*, *Amabile*, and *Dolce* are used. For spumante, however, there are many more possible terms, including *Brut nature*, *Extra Brut*, *Brut*, *Extra dry*, *Sec*, *Demi-sec*, *Doux*. Such information is mandatory only for spumante wines. In addition, if the sugar content of the products justifies the use of two terms, the choice is up to the manufacturer.
- *Color* red, white, rosé. It could also appear with synonyms: red/black, etc.
- *Mention* if present, it indicates a particular wine characteristic. Examples of these are: *Riserva*, *Superiore*, *Novello*, *Passito*, *Lambiccato*. There may also be more than one term on a bottle label.
- *Bottling year* mandatory only for DOP wines.
- *Production method* If present, this information is often accompanied by the relative logo. Some examples are *Organic*, *Vegan*, *No sulfites*.
- *Alcohol volume* mandatory and expressed as a percentage value. In the possible value range, only .5 steps are allowed.
- *Bottle capacity* mandatory. This information and the alcohol volume must be in the same field of view and easily visible (e.g., high color contrast between font and background).

The information introduced to this point is valuable to uniquely identify wine types. The main features are *wine name*, *appellation*, *appellation value*, *effervescence*, *sweetness*, and *sweetness*, which are also the wine descriptors adopted in our system to discriminate among wine types. It is worth noticing that those features are usually placed in the label top area using a font that is larger than the rest of the text [40–42].

4 AWR system

The proposed AWR system, visually depicted in Fig. 2, includes two main components: (a) a client AR interface running on a mobile device, used to take pictures of the wine label and present AR content after wine type identification, and (b) a server that executes an algorithmic pipeline. The latter employs an OCR at two different stages to retrieve the text within the image sent by the mobile device and filter the relevant words retrieved by the OCR. These words are fed to a custom hierarchical search algorithm that skims a hierarchical textual database (textual DB) according to them, providing the best wine type candidates.

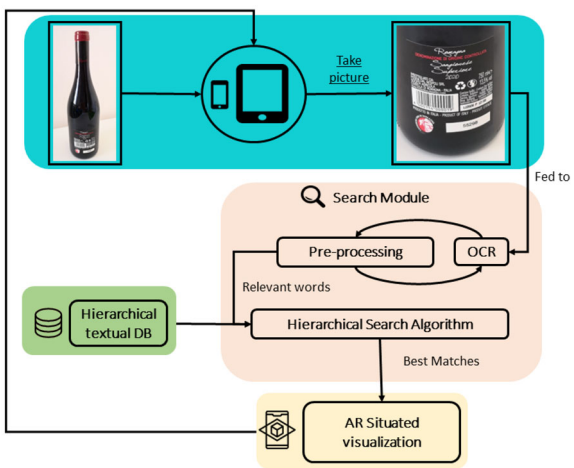


Fig. 2 AWR system

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

Fig. 3 AR interface: **a**, **b** wrong and correct suggestions, **c** correct confirmation, **d** after the confirmed identification, the scan stops

4.1 AR interface

The client side of AWR has been implemented adopting an AR approach for Android-based smartphones. Once activated, the AR interface starts to continuously scan what is framed by the device camera, collecting more and more frames. When a certain number of frames are collected, the system picks the less blurry one, by taking the frame which presents the smallest variance of the Laplacian, as reported in Bansal et al. [44]. This frame is then used by the system to verify whether the camera is pointing at a known label. During this recognition process, a spinning loading icon appears on the screen's bottom left corner. Once a label is recognized, the interface shows the wine name, appellation, region, and region image (if available) related to the first query result. The interface also lists other possible candidates on its right panel. If the right answer is present in this list, a user can select it. In this case, the interface opens a dialogue box asking the user to save the selected result and, if confirmed, only the label related to that result is shown until the "Close" button is pressed. In case the back-end recognition service is not able to match the targeted wine because the related entry is not included in the textual DB, an alert is displayed. In addition, at any moment, the user can stop the interface scan through a toggle on the screen's bottom right corner. The aforementioned processes and features are visually depicted in Fig. 3, which shows four different views of the AWR interface.

4.2 Back-end components

The system back-end components include an algorithmic pipeline employing an OCR to retrieve the text within the image sent by the mobile device and a custom hierarchical search algorithm that skims a hierarchical textual DB based on the previously extracted words, providing the best candidate wine types.

In particular, the OCR is involved multiple times in the text extraction step. At first, during the cropping stage, the OCR is used to reduce the visual search area that encloses the most relevant words, taking advantage of Italian wine label regulations (Sect. 3). For instance, the most discriminative information appears with larger font sizes than any other text. Given this, it is possible to define a bounding box that encloses all relevant text pieces. After this first stage, the OCR predicts the words within the bounding box. The retrieved text is then used to individuate within the hierarchical textual DB the path identifying the sought wine type. This is simplified with the use of a tree characterized by mutually exclusive paths. For example, if a wine is "dolce," it cannot be characterized by any other sweetness value; therefore, the sub-trees not connected to "dolce" will be pruned. In addition, the tree size depends on the presence of specific values. For example, if "DOC" appears on a label, the same label should also report other mandatory information (e.g., harvest year). Leveraging on such type of information, the search space for a given wine type may shrink and its search running time may dramatically drop when compared to a naive approach (details in Sect. 5).

4.2.1 OCR module

An OCR algorithm is employed to find the possible area where relevant words lie and recognize them. Afterward, a search algorithm exploits the textual DB (Sect. 4.2.2) to find such words to identify the wine.

To this date, many OCRs are available in literature [45]. We exploited an off-the-shelf deep learning-based OCR, EasyOCR [46]. The EasyOCR detection component employs the CRAFT algorithm [47], while the recognition model amounts to a CRNN [48] and is trained with the pipeline reported in Baek et al. [49]. Finally, the decoding step utilizes CTC [50]. EasyOCR appeared particularly suited to our use case as its model is trained on images belonging to heterogeneous environments (not only scanned documents). In addition, the results in Smelyakov et al. [51] identified EasyOCR as the best OCR for images of true-to-life scenes. EasyOCR retrieves as output the detected text and the relative bounding boxes encapsulating it, within the original image. It is worth noticing that, in all of our experiments, we executed EasyOCR loading the weights obtained to recognize Italian and English words from a vocabulary utilized by its authors (release version 1.2.2). This means that we did not perform an additional training process, using EasyOCR as it is. Qualitative examples of executing EasyOCR on two Italian wine bottles from the Emilia–Romagna area are reported in Fig. 4.

This first experiment showed the positive and negative aspects of the adopted EasyOCR baseline: Many relevant keywords were recognized, even with different kinds of fonts and backgrounds. However, some text was not detected or correctly transcribed. This can be due to several reasons: Text does not lie on a planar surface, color contrast varies between

Fig. 4 Examples of EasyOCR retrieved words, without considering the confidence factor

background and text, poor light conditions, and the words on the label could be very distant from the ones used to train the EasyOCR decoder (just some technical words like DOC are included). In conclusion, EasyOCR may return values that are mistaken and do not hence support a direct querying mechanism. It may be possible to overcome such limits by exploiting a fine-tuning approach to optimize the EasyOCR model for the wine domain. However, a dataset should be defined from scratch, labeling wine bottle label pictures with the coordinates of the relevant text and its corresponding characters, resulting in a costly procedure in terms of time and workforce. So, a different path was taken: we designed an algorithm that corrects wrong text predictions by leveraging predefined wine domain dictionary terms. As a final note, we named the process of predicting the words by the OCR module as **OCR words inference**.

4.2.2 Wine database

The information (features) that could uniquely identify a type of wine corresponds to the wine name, appellation, appellation value, effervescence, sweetness, and sweetness (check Sect. 3). These features are not independent: a certain value of feature $i \in [1, n]$ defines a subset of all the other possible values for the other features $j \in [1, n]$, $j \neq i$ where n is the total number of features. For this reason, wines are grouped firstly by one feature, like the appellation, and then sub-grouped based on each of the other ones, like the effervescence, and this process continues until one has grouped all the possible features, defining a hierarchical tree data structure. For example, both the *Lambrusco di Sorbara rosato* and the *Reggiano Lambrusco rosato* wine typologies have common appellation value (DOC), color (rosé), and effervescent (sparkling wine). In Fig. 5, an example shows how to transform features from a table-like format to a hierarchical tree structure. This kind of transformation is used to convert our table-like DB composed of 2,426 wine types into a hierarchical tree one.

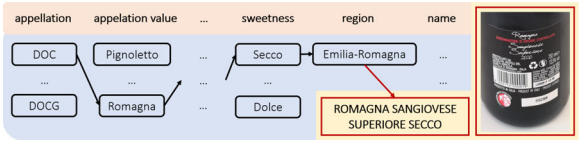


Fig. 5 An example of features conversion from a table-like to a hierarchical tree structure

In particular, the hierarchical textual DB follows a classical non-binary tree structure: each layer of the tree is composed of k nodes, one per each value of a particular feature (e.g., DOC, DOCG, DOP, IGT, IGP for the appellation). Recognizing a wine type means visiting the last level of the hierarchy, returning all the possible hits (i.e., more than one leaf). The considered database follows a specific nested key–value data model in which the key is the value of a feature, and the respective value is the subset of all the wines sharing that particular value. In this way, a single wine tree traversal depends on the number of hierarchy levels (i.e., the features), given a priori by the values of the visited features, and the computational cost in a one-level key–value database is constant. For this reason, efficient NoSQL-like databases appear well suited for this scenario. The schema of the hierarchical tree database follows hence the feature order.

The 2426 textual descriptions of different wines from the Emilia–Romagna region are hence converted in a hierarchy based on the chosen and sorted characterizing features: *appellation*, *appellation value*, *sweetness*, *color*, *effervescence*, *wine name*. This order amounts to a trade-off between (a) pruning as many leaves as possible at the higher node in the tree hierarchy, depending on how many values a feature may take, and (b) the complexity of searching for a given feature, taking into account the average number of words composing a feature by its possible values. In other words, for (a) we considered the number of nodes that may be severed when the value of a feature is known: the more nodes, the higher the position of the feature in the hierarchy. At the same time, we considered the complexity of searching for a match for a given feature with (b). For example, considering (a), in our dataset, there are three possible values for the appellation and thirty for the appellation value. The number of wine types that would remain when choosing an appellation value is, hence, lower than the number that would remain choosing the appellation; therefore, this indicates that the former should be assigned a higher place in the tree hierarchy. When considering (b), instead, the average number of words used in the appellation values by the number of values this may take is higher than the corresponding number for the appellation, thus indicating the appellation should be placed at a higher level of the tree hierarchy. We, therefore, see that (a) and (b) push the organization of the hierarchical tree in different directions. This contrast has been solved using a single factor to evaluate (a) and (b). In fact, we defined for a given feature

f_i a cost function $c_{f_i}(c_{v_{f_i}}, c_{s_{f_i}})$ that synthesizes the contributions of these two components. In particular, the cost function is computed as follows, $c_{f_i} = (w_v \times c_{v_{f_i}}) + (w_s \times c_{s_{f_i}})$, where, $c_{v_{f_i}}$ amounts to the average number of leaves (wine types) remaining in the tree once given feature f_i is chosen, $c_{s_{f_i}}$ to the average number of words used in f_i by the number of values f_i can take, and, w_v and w_s to two constants whose role is to balance the contribution of the two cost components (in our work, these have been set to 0.4 and 0.6, respectively). With the use of $c_{f_i}(c_{v_{f_i}}, c_{s_{f_i}})$, features are ordered within the hierarchical tree with the rationale that a higher hierarchy value is assigned in correspondence of a lower cost.

Summarizing, the algorithm that searches for the wine type in the hierarchical textual DB (detailed in Sect. 4.2.3) first individuates the possible relevant words using an OCR-based cropping and decoding approach. The algorithm then iterates starting at the first level of the hierarchical textual DB, executing a linear search and computing the best match according to a predefined textual distance (e.g., Levenshtein, Hamming, cosine). The linear search compares the text with the term(s) stored at every given level of the hierarchy. In addition, each layer of the hierarchy is composed of a finite set of possible terms, and the algorithm picks the most probable one, which depends on the textual distance computed between the retrieved words and the words contained at the current tree level. Then, the algorithm proceeds to analyze the successive level of the hierarchy only after individuating the best match and after pruning the other branches.

4.2.3 Search module

We here describe the full algorithmic pipeline used in the AWR system to recognize a wine type from a picture of the back label of a wine bottle. In particular, we here describe how the domain-specific wine hierarchical textual DB, EasyOCR, OCR, and the novel text preprocessing and hierarchical search are orchestrated together to return the best possible matches.

As mentioned before, relevant information is usually placed in the top area of the label with a font larger than other text [42]. For the words that compose that text retrieved by EasyOCR, the area is usually larger than any other retrieved one. Assuming this, a preprocess step composed by the **OCR words area detection**, which automatically detects and crops the areas that enclose the words of interest, is implemented after having executed the first **OCR words inference**. Not performing this preprocessing step, would require accounting for all of the words identified on a label. The aim is to avoid such a case. More in detail, only the words enclosed in the bounding boxes larger than the sample median are in the end selected, and the bounding box that includes all of such words become the one used to crop the label. With these conditions, it is possible to detach the word detection phase from the textual inference phase and reduce the number of processed words. At the same time, the words that could be misinterpreted by the OCR due to their size and location are discarded (e.g., small area words placed near the boundaries). An example of the execution of such preprocessing is reported in Fig. 6.

Original frame Area of interest Cropped area

Fig. 6 Example of cropping the area of interest

The **OCR words inference** is then again executed, returning all the words found in the cropped image. The pipeline hence only considers a subset of the words appearing on the label, reducing the computational cost. To further reduce the set, duplicates and stopwords are also removed.

Algorithm 1 Hierarchical search algorithm

```
1: procedure HIERARCHICAL_SEARCH(
   bottle_retrieved_words, h_db, bottle_features, threshold, distance_method)
2:     for f in bottle_features do
3:         dict_match ← {}
4:         for v in f.possible_values() do
5:             matched_terms ← Algorithm 2(
                                 bottle_retrieved_words, v,
                                 threshold, distance_method)
6:             dict_match[v] ← matched_terms
7:         end for
8:         correct_value ← highest_score(dict_match)
9:         bottle_retrieved_words, h_db ← branch_db(
            h_db, bottle_retrieved_words, correct_value)
10:    end for
11:    return h_db
12: end procedure
```

The obtained set of words is then processed in the last step of the entire pipeline: the *hierarchical search algorithm* whose Python-like pseudo-code is detailed in Algorithm 1. The algorithm searches for the best wine type by exploring each level of the hierarchical tree database and searching for a match within the set of words retrieved by the OCR stopping at the node exhibiting the text value with the minimum textual distance (check Algorithm 1) [52–58].

In brief, this algorithm takes as parameters the words returned by the OCR, a copy of the hierarchical textual DB, the sorted list of features, and a distance threshold percentage, which will be used in the Algorithm 2 (line 1). Subsequently, it cycles on the relevant features to initialize the features dictionary (lines 2 and 3). The second cycle (line 4), instead,

iterates on the possible values of the considered feature (e.g., DOC, DOCG, DOP, IGT, IGP for the appellation).

Then, the *linear search (post-OCR) correction algorithm* finds any existing matches for the given feature term (line 5), and all matches are added to the dictionary (line 6). Now, the algorithm selects the feature value that received the highest number of matches. For example, between "Denominazione origine controllata" and "Denominazione origine controllata e garantita," the latter would be chosen in the case that all the terms are matched with some word in the OCR retrieved words. In case of a tie between two features, both would be selected. Then, the dictionary contains all the matches found for the given feature values (line 8). Once a particular feature value(s) is picked, the hierarchical DB is skimmed, branching the specific sub-tree(s) involving that value or those values. After applying the skimming step for all the wine features, the remaining elements of the hierarchical DB contain only one or more elements that possibly include the correct wine (line 9), and the hierarchical DB is returned (line 11). Given this, the *hierarchical search algorithm* computational cost depends on the number of considered features f (i.e., the height of the tree), and on the cost of the *linear search correction algorithm*, described below.

The *linear search correction algorithm* implements two different sub-tasks: detection and correction. The detection task identifies incorrect tokens, and the correction task tries to correct the errors found by the previous one. The algorithm adopts an isolated word approach, relying on specific lexicons, or word unigram language models, and a distance for selecting candidates of OCR errors. In this case, the Levenshtein distance metric has been utilized [52]. The corresponding Python-like pseudo-code appears in Algorithm 2.

Algorithm 2 Linear search correction algorithm

```
1: procedure LINEAR_SEARCH_CORRECTION(
   ocr_retrieved_words, feature_words, threshold, distance_method)
2:     matched_words ← []
3:     for w in feature_words do
4:         for w_ocr in ocr_retrieved_words do
5:             distance ← distance_method(w_ocr, w)
6:             thr_word ← threshold × len(w)
7:             if distance ≤ thr_word then
8:                 matched_words.append(w)
9:                 break
10:            end if
11:        end for
12:    end for
13:    return matched_words
14: end procedure
```

The algorithm works as a two-level nested loop that iterates over the words retrieved by the OCR (first level, line 3) and the values of the word that define a feature (second level, line 4). The algorithm computes the distance adopting the `distance_method` per each couple (`w`, `w_ocr`) (line 5), and if the distance is less or equal than a certain threshold, `w_ocr` is considered to be `w` (lines 6, 7 and 8). When this happens, the algorithm interrupts the inner loop (line 8) and continues to search for the next relevant word inside the label text (line 3). To note that, here, the threshold represents a value of accepted distance: A lower distance indicates that it is more likely that `w_ocr` matches `w`.

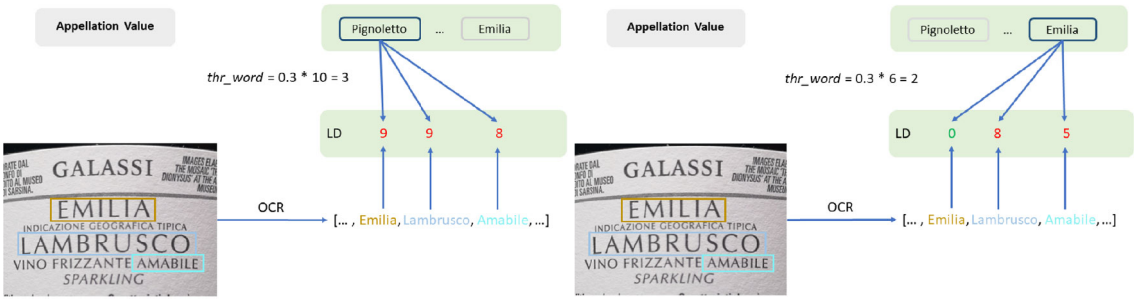
More in detail, `thr_word`) is calculated based on the length of the word to match (`w`) as a simple percentage of the considered `threshold` (line 6). A `threshold` of 0 indicates that the two words must be the same (i.e., no difference), while, if set to 1, indicate that the `w_ocr` would match `w` if their distance is less or equal than p, where p is the number of character of `w`. Figure 7 visually reports a sample *the linear search algorithm* applied to two words of the **appellation value** feature with a `threshold` of 0.3 (i.e., 30%): On the left side, the OCR retrieved words are compared with a non-matching value while on the right side, a matching word was detected.

The computational cost of Algorithm 2 depends on the two lists of size n and m, respectively (i.e., the number of `ocr_retrieved_words` and `feature_words`), which are constants as these may be both upper bounded by some fixed value (named as α). Considering now the cost of the full *hierarchical search algorithm* (Algorithm 1), the height of the traversed hierarchical tree database corresponds to f by construction, while an additional cost is provided to the maximum number of children to match at each level, named z (upper bound). Considering that the cost to search for a match on a single node corresponds to α, the Algorithm 1 total cost corresponds to $f \times z \times \alpha$. It should be noted that the values at stake are all constants that do not exceed a few tens of units, simply meaning that an asymptotic analysis does not apply.

Using this *Search algorithm* it is possible to recognize terms belonging to the wine domain also when distorted by the OCR. In addition, the *Search algorithm* exploits, in general, the wine domain knowledge (Sect. 3): some features (e.g., effervescence) possess a default value, implying that if none of the non-default values are detected, the default one is taken. Again, failures may be due to mistaken feature identification (i.e., no relevant word appears in the OCR list or the words are in part wrongly predicted), and/or relevant information is not reported on the label. Figure 8 visually depicts an example of the hierarchical search algorithm matching all the considered features.

5 Experiments and results

We here report the results obtained by applying the *hierarchical search algorithm* (Algorithm 1, Sect. 4) to 45 different wine bottles coming from the Emilia–Romagna region in Italy, and considering a hierarchical textual DB containing



Wrong correspondence produces high Levesthein Distances.

A perfect correspondence was found between the OCR-detected words and the searched ones.

Fig. 7 Example of linear search correction algorithm iterating over OCR retrieved words and node values for the **appellation value** feature (threshold is set to be equal to the 30% of the label word). In this figure, LD stands for Levenshtein distance

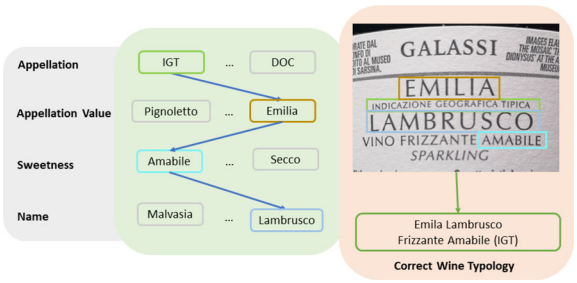


Fig. 8 Example of a correctly matched hierarchical search algorithm traversal considering *appellation*, *appellation value*, *sweetness*, and *wine name*

2, 426 entries. The evaluation has been performed in a multi-frame setting, collecting and using multiple frames per wine.

More in detail, we collected 45 videos, one per each different bottle, lasting an average of 8 seconds, rotating the camera around the bottle label. Then, we selected from each video the less blurred frame per second, adopting the variance of image Laplacian, as reported in Bansal et al. [44]. The use of multiple frames is motivated principally by the expectation that employing videos may be possible to reduce OCR detection errors caused by environmental problems (e.g., light conditions), and the fact that taking a video does not require the user to take a picture of what s/he is seeing each time. Intending to provide a complete picture of the AWR system performance, we considered and reported the results both in terms of *efficacy* and *efficiency*. Per each considered video frame, the algorithmic pipeline has been applied to the words retrieved by EasyOCR. The result is a set composed of all the wines that expose an equal difference between the original name length and the number of matched words. To compute **efficacy**, wine bottles are considered to be correctly recognized if their names are included in the set of retrieved ones. Now, before presenting the final results, it is necessary to highlight that by tuning the hyperparameters it was possible to individuate the threshold values that best identify a word as recognized (Algorithm 2).

Recalling that a word retrieved by EasyOCR is considered to match one value for each different feature if their Levenshtein distance falls below a given threshold, the selection of the threshold value has been performed assessing the pipeline for all the values between 0 and 1 with a step of 0.1. In particular, two distinct thresholds have been set for words and acronyms (e.g., Denominazione Origine Controllata vs DOC), 0.3 and 0.1, respectively, capable of returning the best results: with these values, 41/45 bottles were correctly recognized (91% of accuracy). This accuracy was computed considering that the algorithm could provide more than one wine as output, retrieving on average 1.41 bottles per analyzed video (min 1, max 15). It is worth noticing that the errors could depend on the EasyOCR performance. To verify this, we defined 45 different textual files (one per bottle) which contain each word reported in the wine bottle label, in order to simulate the performance of a perfect OCR, transcribing all the words appearing on labels, respecting their order. With such data and a threshold set to 0, as a perfect OCR should match the exact words, the algorithm reached a 100% accuracy. Always considering that the algorithm could provide more than one wine as an output, the linear search algorithm applied to the perfect OCR setting retrieved on average 1.01 bottles per examined video (min 1, max 8). All such results are summarized and reported in Table 2.

We then measured the system **efficiency**, i.e., the time taken to return results. The algorithm is composed of four steps (Sect. 4): OCR words area detection, cropping by detected areas (image cropping), OCR words inference, and the *hierarchical search algorithm*. The captured times regarding EasyOCR do not include the deep learning model initialization times, as not relevant. Time measurements come from executing a single trial of the complete algorithmic pipeline on the selected video frames for all the bottles. Fig. 9 depicts the min, mean, and max times, provided in seconds, for each algorithm component, averaged over ten trials.

Table 2 Results obtained with chosen confidence values

Performance metric	AWR (words = 0.3; acronyms=0.1)	AWR-POCR (words = 0.0; acronyms=0.0)
Guessed bottles (correct/total)	41/45	45/45
Retrieved bottles (min, mean, max)	1, 1.41, 15	1, 1.01, 8

AWR indicates the performance obtained by using our system with the selected OCR using as thresholds for words and acronyms of 0.3 and 0.1, respectively. AWR-POCR, instead, regards the performance obtained by using our system with a simulated perfect OCR and so accepting only perfect matches (words and acronyms thresholds correspond to 0.0)

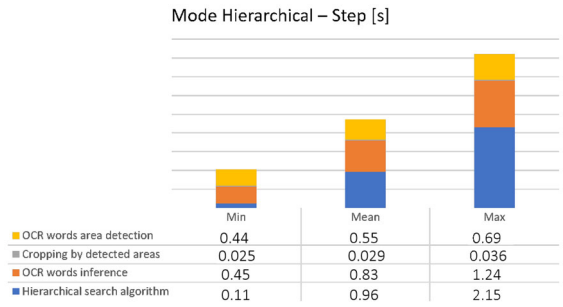


Fig. 9 Total time in seconds over different hierarchical mode algorithm steps

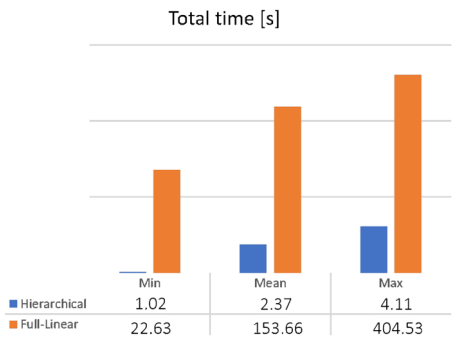


Fig. 10 Hierarchical vs full-linear total time in seconds (plotted in log scale)

We then compared the efficiency of the hierarchical tree search and the classical linear one. We defined a flat database containing all the relevant values of the different distinguishing features, and we applied Algorithm 1 at each step of the data structure. So, the final output includes the wine(s) with the highest number of matching features. We report in Fig. 10 the computed min, mean, and max time values obtained over the same wine bottle set used for the previous evaluations, again over ten trials. It is possible to note that a naive approach, like the *full-linear* one, is roughly a hundred times slower than the *hierarchical* proposed approach. This fact is justified since in the *full-linear* mode the tree is never pruned, so the algorithm compared all the retrieved words with all the words describing the 2, 426 wine types.

6 Discussion and conclusions

This work provides an overall description of the required domain (e.g., wine background) and technical (e.g., OCR approach) knowledge used to define the AWR wine recognition app [22]. Nowadays (augmented) wine recognition systems and services are typically based on pure computer vision approaches, but not without limitations. In fact, a wine could possess different visual features (in time), but many wines could appear similarly. A sole visual analysis may lead to wrong classifications, because of label changes, due to marketing reasons. At the same time, long-tail samples (such as novel wine types or old ones) may not be guessed, because of the lack of images that capture their labels. Avoiding these phenomena requires frequent database/fine-tuning updates. Sometimes those updates cannot be done considering hard-to-find wine labels. We here showed that by adopting a textual-based approach, it is possible to overcome such limitations and speed up the database update process. However, additional work is needed to improve and generalize our system in a more varied context.

Firstly, the presented system solely leverages text for the search of wine types, such an approach could be extended by integrating one based on image recognition. In fact, it is possible to envision a hybrid approach to prune, as much as possible, the hierarchical textual DB and execute an image query on the remaining corresponding wine label images. Secondly, we could exploit OCRs to detect additional information within labels, like bottle capacity and alcohol content. These could then be used to present additional information in AR, such as caloric intake, and maximum recommended dose. Thirdly, additional work could be implemented to increase the performance of the underlying OCR. For example, image preprocessing such as image warping, unwrapping, rectification, and semantic segmentation could be adopted [59–62]. In addition, the performance of deep learning-based OCRs could be improved by implementing a fine-tuning procedure leveraging a large dataset composed of wine label pictures labeled with bounding boxes and corresponding textual annotations. This would ameliorate the erroneous word prediction highlighted in the results reported in Table 2. This will also pose the basis for a more general and

applicable detection pipeline that exceeds the Italian wines domain (e.g., fine-tuning also in other languages).

Hence, many possible future research directions could be explored to improve AWR. Nevertheless, its promising performance shows that an identification mechanism based not only on visual features but also on textual data could be a valid method in many application contexts.

Acknowledgements This work was supported by the University of Bologna with the Alma Attrezzature 2017 grant, by AEFFE S.p.a., the Golinelli Foundation with the funding of two Ph.D. scholarships, and by ImageLine S.r.l. for providing the wine textual dataset.

Funding Open access funding provided by Alma Mater Studiorum - Università di Bologna within the CRUI-CARE Agreement.

Declaration

Conflict of interest We hereby declare that we have no conflicts of interest that could be perceived as influencing the integrity or objectivity of our research work.

References

1. Bayu, M.Z., Arshad, H., Ali, N.M.: Nutritional information visualization using mobile augmented reality technology. Proc. Technol. **11**, 396–402 (2013)
2. Haugstvedt, A.-C., Krogstie, J.: Mobile augmented reality for cultural heritage: a technology acceptance study. In: 2012 IEEE International Symposium on Mixed and Augmented Reality (ISMAR), pp. 247–255 (2012). IEEE
3. Tscheu, F., Buhalis, D.: Augmented reality at cultural heritage sites. In: Inversini, A., Schegg, R. (eds.) Information and Communication Technologies in Tourism 2016, pp. 607–619. Springer, Cham (2016)
4. Stacchio, L., Hajahmadi, S., Marfia, G.: Preserving family album photos with the hololens 2. In: 2021 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW), pp. 643–644 (2021). IEEE
5. Büschel, W., Mitschick, A., Dachselt, R.: Here and now: reality-based information retrieval: perspective paper. In: Proceedings of the 2018 Conference on Human Information Interaction & Retrieval, pp. 171–180 (2018)
6. Bressa, N., Korsgaard, H., Tabard, A., Houben, S., Vermeulen, J.: What's the situation with situated visualization? A survey and perspectives on situatedness. IEEE Trans. Vis. Comput. Gr. **28**(1), 107–117 (2021)
7. Martins, N.C., Marques, B., Alves, J., Araújo, T., Dias, P., Santos, B.S.: Augmented reality situated visualization in decision-making. Multimed. Tools Appl. **81**(11), 14749–14772 (2022)
8. Fitzmaurice, G.W.: Situated information spaces and spatially aware palmtop computers. Commun. ACM **36**(7), 39–49 (1993)
9. Orsini, A., Venkatesan, G., Huang, G., Shah, G., Shah, N.: Augmented Reality Enhanced Cooking with Microsoft Hololens. State University of New Jersey, Rutgers (2017)
10. Rejeb, A., Rejeb, K., Keogh, J.G.: Enablers of augmented reality in the food supply chain: a systematic literature review. J. Foodserv. Bus. Res. **24**(4), 415–444 (2021)
11. Styliaras, G.D.: Augmented reality in food promotion and analysis: review and potentials. Digital **1**(4), 216–240 (2021)
12. Yuka: Yuka. https://yuka.io/it/ (2021)
13. Vivino: Vivino. https://www.vivino.com/ (2021)
14. Vrigkas, M., Lappas, G., Kleftodimos, A., Triantafillidou, A.: Augmented reality for wine industry: past, present, and future. In: SHS Web of Conferences, vol. 102, p. 04006 (2021). EDP Sciences
15. Sonderegger, A., Ribes, D., Henchoz, N., Groves, E.: Food talks: visual and interaction principles for representing environmental and nutritional food information in augmented reality. In: 2019 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct), pp. 98–103 (2019). IEEE
16. Hinz, O., Eckert, J., Skiera, B.: Drivers of the long tail phenomenon: an empirical analysis. J. Manag. Inf. Syst. **27**(4), 43–70 (2011)
17. Stricker, S., Mueller, R.A., Sumner, D.A.: Marketing wine on the web. Choices **22**, 31–34 (2007)
18. Alston, J.M., Gaeta, D.: Reflections on the political economy of European wine appellations. Ital. Econ. J. **7**(2), 219–258 (2021)
19. Breuel, T.M.: High performance text recognition using a hybrid convolutional-lstm implementation. In: 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), vol. 01, pp. 11–16 (2017). https://doi.org/10.1109/ICDAR.2017.12
20. Wick, C., Reul, C., Puppe, F.: Calamari-a high-performance tensorflow-based deep learning package for optical character recognition. arXiv preprint arXiv:1807.02004 (2018)
21. Charters, S., Lockshin, L., Unwin, T.: Consumer responses to wine bottle back labels. J. Wine Res. **10**(3), 183–195 (1999)
22. Stacchio, L., Angeli, A., Donatiello, L., Giacchè, A., Marfia, G.: Rethinking augmented wine recognition. In: 2022 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct), pp. 1–6 (2022). IEEE, *to appear*
23. Penco, L., Serravalle, F., Profumo, G., Viassone, M.: Mobile augmented reality as an internationalization tool in the "made in Italy" food and beverage industry. J. Manage. Governance **25**(4), 1179–1209 (2021)
24. Salim, N.O., Zeebaree, S.R., Sadeeq, M.A., Radie, A., Shukur, H.M., Rashid, Z.N.: Study for food recognition system using deep learning. J. Phys: Conf. Ser. **1963**, 012014 (2021)
25. Gundimeda, V., Murali, R.S., Joseph, R., Babu, N.N.: An automated computer vision system for extraction of retail food product metadata. In: First International Conference on Artificial Intelligence and Cognitive Computing, pp. 199–216 (2019). Springer
26. Hu, B., Zhou, N., Zhou, Q., Wang, X., Liu, W.: Diffnet: a learning to compare deep network for product recognition. IEEE Access **8**, 19336–19344 (2020)
27. Lin, M., Ma, L., Yu, B.: An efficient and light-weight detector for wine bottle defects. In: 2020 16th International Conference on Control, Automation, Robotics and Vision (ICARCV), pp. 957–962 (2020). IEEE
28. Zhu, L., Spachos, P., Pensini, E., Plataniotis, K.N.: Deep learning and machine vision for food processing: a survey. Curr. Res. Food Sci. **4**, 233–249 (2021)
29. TinEye: WineEngine is image recognition for the beverage industry. https://services.tineye.com/WineEngine (2021)

30. livingwinelabels: livingwinelabels. https://www.livingwinelabels.com/ (2021)
31. PTC: Vivino and Vuforia's Image Recognition Solution Make a Great Pairing. https://www.ptc.com/en/case-studies/vivino (2022)
32. Gebru, T., Hazi, O., Yeh, V.: Mobile wine label recognition (2022)
33. Na, I.S., Chen, Y.J., Kim, S.H.: Automatic segmentation of product bottle label based on grabcut algorithm. Int. J. Contents **10**(4), 1–10 (2014)
34. Wu, M.-Y., Lee, J.-H., Kuo, S.-W.: A hierarchical feature search method for wine label image recognition. In: 2015 38th International Conference on Telecommunications and Signal Processing (TSP), pp. 568–572 (2015). IEEE
35. Čakić, S., Popović, T., Šandi, S., Krčo, S., Gazivoda, A.: The use of tesseract ocr number recognition for food tracking and tracing. In: 2020 24th International Conference on Information Technology (IT), pp. 1–4 (2020). IEEE
36. Jung, J.-M., Yang, H.-J., Kim, S.-H., Lee, G.-S., Kim, S.-H.: Wine label recognition system using image similarity. J. Korea Contents Assoc. **11**(5), 125–137 (2011)
37. Álvarez Márquez, J.O., Ziegler, J.: Improving the shopping experience with an augmented reality-enhanced shelf. Mensch und Computer 2017-Workshopband (2017)
38. Li, X., Yang, J., Ma, J.: Cnn-sift consecutive searching and matching for wine label retrieval. In: International Conference on Intelligent Computing, pp. 250–261 (2019). Springer
39. Vuforia: Vuforia SDK. https://developer.vuforia.com/downloads/SDK (2022)
40. Camera di Commercio Molise: Guida etichettature vino. https://www.molise.camcom.gov.it/sites/default/files/guida_etichettatura_vino.pdf (2016)
41. Michele A. Fino: Questione di Etichetta. https://www.spazioprever.it/salabar/vino/pdf/Questione_di_etichetta.pdf (2013)
42. Vittorio Portinari: Elementi di Legislazione Vitivinicola: le norme per l'etichettatura e la tracciabilità dei vini. http://www.sardegnaagricoltura.it/documenti/14_43_20160531144229.pdf (2016)
43. FEDERDOC: I VINI ITALIANI A DENOMINAZIONE D'ORIGINE 2020. https://www.federdoc.com/new/wp-content/uploads/2020/06/vini_italiani_denominazione_origine_2020.pdf (2021)
44. Bansal, R., Raj, G., Choudhury, T.: Blur image detection using laplacian operator and open-cv. In: 2016 International Conference System Modeling Advancement in Research Trends (SMART), pp. 63–67 (2016). https://doi.org/10.1109/SYSMART.2016.7894491
45. Singh, A., Bacchuwar, K., Bhasin, A.: A survey of OCR applications. Int. J. Mach. Learn. Comput. **2**(3), 314 (2012)
46. Easy Ocr: JadedAI. https://github.com/JaidedAI/EasyOCR (2021)
47. Baek, Y., Lee, B., Han, D., Yun, S., Lee, H.: Character region awareness for text detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 9365–9374 (2019)
48. Shi, B., Bai, X., Yao, C.: An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. IEEE Trans. Pattern Anal. Mach. Intell. **39**(11), 2298–2304 (2016)
49. Baek, J., Kim, G., Lee, J., Park, S., Han, D., Yun, S., Oh, S.J., Lee, H.: What is wrong with scene text recognition model comparisons? Dataset and model analysis. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 4715–4723 (2019)
50. Graves, A., Fernández, S., Gomez, F., Schmidhuber, J.: Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In: Proceedings of the 23rd International Conference on Machine Learning, pp. 369–376 (2006)
51. Smelyakov, K., Chupryna, A., Darahan, D., Midina, S.: Effectiveness of modern text recognition solutions and tools for common data sources. In: CEUR Workshop Proceedings, pp. 154–165 (2021)
52. Levenshtein, V.I., : Binary codes capable of correcting deletions, insertions, and reversals. In: Soviet Physics Doklady, vol. 10, pp. 707–710 (1966). Soviet Union
53. Abualigah, L., Diabat, A., Mirjalili, S., Abd Elaziz, M., Gandomi, A.H.: The arithmetic optimization algorithm. Comput. Methods Appl. Mech. Eng. **376**, 113609 (2021)
54. Abualigah, L., Yousri, D., Abd Elaziz, M., Ewees, A.A., Al-Qaness, M.A., Gandomi, A.H.: Aquila optimizer: a novel meta-heuristic optimization algorithm. Comput. Ind. Eng. **157**, 107250 (2021)
55. Abualigah, L., Abd Elaziz, M., Sumari, P., Geem, Z.W., Gandomi, A.H.: Reptile search algorithm (RSA): s nature-inspired meta-heuristic optimizer. Expert Syst. Appl. **191**, 116158 (2022)
56. Oyelade, O.N., Ezugwu, A.E.-S., Mohamed, T.I., Abualigah, L.: Ebola optimization search algorithm: a new nature-inspired meta-heuristic optimization algorithm. IEEE Access **10**, 16150–16177 (2022)
57. Agushaka, J.O., Ezugwu, A.E., Abualigah, L.: Dwarf mongoose optimization algorithm. Comput. Methods Appl. Mech. Eng. **391**, 114570 (2022)
58. Ezugwu, A.E., Agushaka, J.O., Abualigah, L., Mirjalili, S., Gandomi, A.H.: Prairie dog optimization algorithm. Neural Comput. Appl. **34**(22), 20017–20065 (2022)
59. Glasbey, C.A., Mardia, K.V.: A review of image-warping methods. J. Appl. Stat. **25**(2), 155–171 (1998)
60. Zhan, F., Lu, S.: Esir: End-to-end scene text recognition via iterative image rectification. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 2059–2068 (2019)
61. Follmann, P., Drost, B., Böttger, T.: Acquire, augment, segment and enjoy: weakly supervised instance segmentation of supermarket products. In: Brox, T., Bruhn, A., Fritz, M. (eds.) Pattern Recognition, pp. 363–376. Springer, Cham (2019)
62. Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., Xiao, T., Whitehead, S., Berg, A.C., Lo, W.-Y., et al.: Segment anything. arXiv preprint arXiv:2304.02643 (2023)

Alessia Angeli is a Ph.D. student in Computer Science at the Department of Computer Science and Engineering at the University of Bologna. She is a member of the Virtual and Augmented Laboratory (VARLAB) of the University of Bologna. Her research topics and publications concern the application of deep learning and extended reality paradigms.

Lorenzo Stacchio is a Ph.D. student in Computer Science at the Department for Life Quality Studies at the University of Bologna. He is a member of the Virtual and Augmented Laboratory (VARLAB) of the University of Bologna. His research topics and publications concern the application of deep learning and extended reality paradigms in well-being-related domains.

Lorenzo Donatiello is Professor at the Department of Computer Science and Engineering, University of Bologna. His research interests concern Distributed Systems, Simulation, Performance Evaluation, Wireless Networks, Mobile Systems.

Alessandro Giacchè is a Master Student in Computer Science at the University of Bologna.

Gustavo Marfia obtained an Italian Laurea Degree in Telecommunications Engineering in 2003 from the University of Pisa and a Ph.D. in Computer Science from the University of California, Los Angeles, in 2009. Since 2019 he is an Associate Professor in Computer Science with the Department of the Arts of the University of Bologna, where he leads the Virtual and Augmented Reality Lab (VARLAB).