Cooking Action Recognition with *i*VAT: An *Interactive* Video Annotation Tool

Simone Bianco[1], Gianluigi Ciocca[1], Paolo Napoletano[1], Raimondo Schettini[1], Roberto Margherita[2], Gianluca Marini[3], and Giuseppe Pantaleo[3]

[1] DISCo (Dipartimento di Informatica, Sistemistica e Comunicazione)
Università degli Studi di Milano-Bicocca, Viale Sarca 336, 20126 Milano, Italy
{bianco,ciocca,napoletano,schettini}@disco.unimib.it
[2] Almaviva S.p.a.
r.margherita@almaviva.it
[3] Almawave S.r.l.
{g.marini2,g.pantaleo}@almaviva.it

Abstract. Within a video recipe, we are interested in locating and annotating the various ingredients, kitchenwares and relevant cooking actions. To this end we have developed the *i*VAT *interactive* video annotation tool to support manual, semi-automatic and automatic annotations obtained on the basis of the interaction of the user with various detection algorithms. The tool integrates versions of computer vision algorithms, specifically adapted to work in an interactive and incremental learning framework. *i*VAT has been developed to annotate video recipes, but it can be easily adapted and used to annotate videos from different domains as well. In this paper we present some results with respect to the task of cooking action recognition.

Keywords: Interactive video annotation, object recognition, action recognition, tracking, incremental learning.

1 Introduction

The annotation of cooking videos is a task that is becoming very popular in the context of promoting a better nutrition awareness. By developing applications that support users in managing their food consumptions, will help them to prevent nutrition-related health problems. For example, in order to create a cooking assistant application to guide the users in the preparation of the dishes relevant to their profile diets, food preferences and intolerances, it is necessary to accurately annotate the video recipes, identifying and tracking the foods being processed by the cook, and the relevant actions performed.

The manual generation of video annotations by a user is time-consuming, tedious and error-prone. Theoretically, fully automatic tools that integrate computer vision algorithms to extract and identify the elements of interest across frames, could be developed. Unfortunately, state-of-the-art algorithms are not error free, and false positive and false negative detections would require a human

A. Petrosino (Ed.): ICIAP 2013, Part II, LNCS 8157, pp. 631–641, 2013.

effort to correct them in a post-processing stage. As a consequence, several efficient semi-automatic visual tools have been developed [15,19,17,13,1]. Usually, such tools, that support the annotator with *basic* computer vision algorithms (i.e. key frame detection, motion and shape linear interpolation, etc.), have demonstrated to be very effective in terms of the number of user interactions, user experience, usability, accuracy and annotation time [17]. The most recent trend is the development of tools that assist the user in the annotation process by integrating computer vision algorithms for object detection and tracking [9,18]. To the best of our knowledge, a system that integrates automatic object and action recognition, and manual annotation in an interactive framework, does not exist.

For this reason, we developed an interactive annotation tool (iVAT) to support the user during the annotation of cooking videos [2]. Our tool integrates different computer vision modules for object detection, tracking, and action recognition within an incremental learning framework that allows to increase the accuracy of the underlying algorithms over time. As a consequence, the human effort required for the annotations is reduced with respect to completely manual ones.

A preliminary system usability test [3] confirmed the usability of our annotation tool. On the overall, the system received a very good evaluation with an average 4 out of a maximum of 5 points. A more detailed description of these results can be found in [2].

In this paper, after the overview of the iVAT in which we describe the annotation modalities available (i.e. manual, semi-automatic, and automatic) and how they interact with each other, we focus on the action recognition task. After describing the action recognition algorithm available in the iVAT, the action categories considered and the recognition results obtained are reported.

2 *i*VAT Overview

In recent years, several semi-automatic visual tools have been developed [15,19,17,13,1]. Such tools, that support the annotator with algorithms that accomplish elementary computer vision tasks (i.e. key frame detection, motion and shape linear interpolation, etc...), have demonstrated to be quite effective in terms of the number of user interactions, user experience, usability, accuracy and annotation time [17]. However, in order to further simplify the annotation task, the most recent trend is the integration of algorithms that accomplish more complex computer vision tasks, such as unsupervised/supervised object detection, action recognition, object tracking, etc... [9,18].

Following this trend we developed the iVAT, an interactive annotation tool, whose description is reported in Fig. 1. The tool handles a video annotation session as a project. It takes a video and a related list of items as inputs and provides item annotations and templates as outputs. The lists can be given or, if suitable analysis algorithms are available (e.g. [4]), obtained from the input data. The items to be annotated are grouped into different categories, for example, in the case of cooking domain, we have food, kitchenware and action categories. The annotation of an item can be achieved by following three modalities:

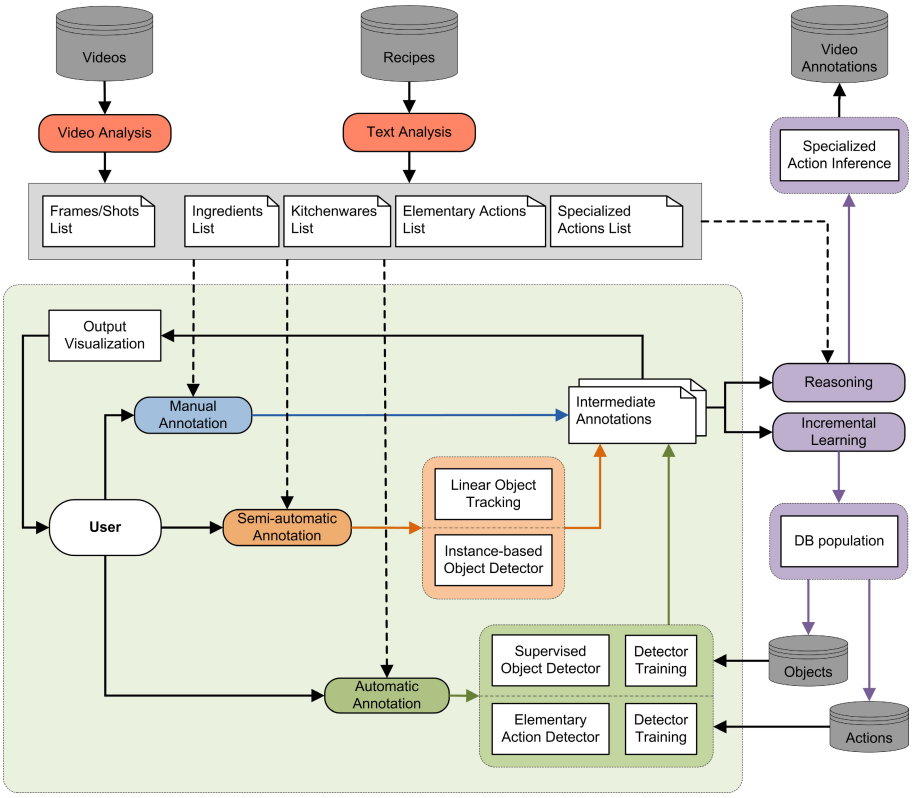


Fig. 1. The *i*VAT overview

1. *manual*: the user provides, frame by frame, the annotations of the item;
2. *semi-automatic*: the user provides an initial annotation and automatic algorithms provide the subsequent annotations;
3. *automatic*: algorithms provide, frame by frame, the annotations of the item.

All the annotation process can be achieved through a simple graphical user interface (GUI) shown in Fig. 2. The upper part contains video related information: list of shots, list of items and more important a video browser which allows the user to seek through frames and sequentially browse shots. The currently annotated items are shown in the displayed video frame. To facilitate the annotation process, the tool provides to the user different interaction modalities: *click-able buttons*, *drag & drop* operations, *context menus* and *short-cuts*. Every time a new item is annotated (manually or automatically), a new item's time-line appears in the lower part of the window. Each time-line allows the user to visually keep track of where the corresponding item has been annotated. Each cell in the time-line correspond to a frame position and it shows if the item is present in that frame (i.e. visible), if the annotation has been obtained manually

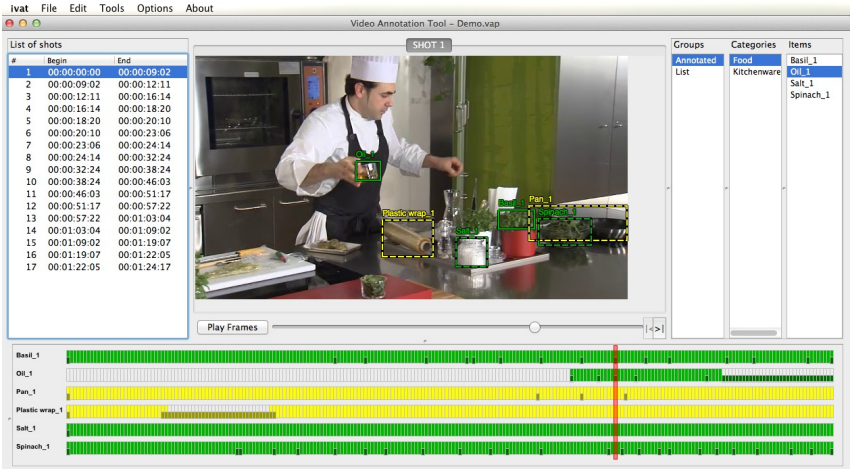


Fig. 2. The *i*VAT GUI

of by an algorithm and if the annotation cannot be changed (i.e. locked). A filled cell corresponds to the presence of the item in the corresponding frame, while an empty cell corresponds to its absence. A marker superimposed on a cell indicates an annotation that can be modified only by the user (i.e. locked annotation).

Manual Annotations. A manual annotation is achieved by firstly choosing it from the list on the right side of the GUI, and later by dragging and dropping it on the video frame. Once the item is dropped on the image the user can draw a bounding box around the object. Manually annotated items are identified by bounding boxes with a solid outline. The size and position of the bounding box can be changed at any time as well as deleting it. The color of the bounding box depends on the item's category. By default all the manual annotations are locked meaning that the automatic algorithms cannot modify or overwrite them. Only the user can change these annotations.

Semi Automatic Annotations. Once the user annotates a new item on the video frame, by default a linear object tracking takes place. The position of the items is propagated on subsequent frame. If another annotation of the same item is present in a later frame, a linear interpolation algorithm is used to propagate the item positions. If the linear object tracking is disabled, the user can activate an instance based object detector algorithm that, by using the first annotation as an example, try to automatically detect the position of the same item in the subsequent frames. The detection is done by a spatially-constrained template matching using a normalized correlation coefficient as the similarity measure between the template and candidate items. With the exception of the first, manual, one, all the subsequent annotations obtained in this modality are shown with dashed bounding boxes. These annotations are also not locked.

Automatic Annotations. Automatic annotations can be provided by different supervised algorithms embedded in the system. The automatic annotation can be activated by selecting an item from the list on the right side of the GUI and then by selecting the preferred algorithm in the context menu. This class of algorithms needs a learned template to work, therefore the tool allows users to crop object templates to be used later for training a supervised object detector in the *incremental learning* stage. The different supervised object detection algorithms implemented employ a cascade of boosted classifiers working with Haar-like features [16,11], Histograms of Oriented Gradients (HOG) features [5], and Local Binary Patterns (LBP) features [10]. The action detector uses the informative spatio-temporal features [6]. This algorithm will be described in more details in Section 3. Again, in this modality the annotations are shown with dashed bounding boxes and these annotations are not locked.

Interaction between Manual and Automatic Annotations. Since the annotation of an item can be manually or automatically provided, to handle the interactions of the annotations provided by the user with ones provided by the algorithms, we have introduced the concept of *locked* and *unlocked* objects. This concept is related to a specific object at time instant t. If the annotation at time t has been provided or modified by the user, then the state of the annotated item, independently of its presence at time t, is locked. On the contrary if the annotation at time t has been provided or modified by an algorithm, then the state of the annotated item, independently of its presence at time t, is unlocked. Only the user can modify the state of annotated items changing it from locked to unlocked and vice-versa. In Fig. 3 we show the finite state machine that describes all the possible interactions between manual and automatic change of annotations.

3 Cooking Action Recognition

The design and selection of computer vision algorithms for the cooking domain is particularly challenging for both object detection and action recognition task [7,14,8]. For instance, during the recipe preparation, foods may change their visual appearance and be occluded by cook's hands or kitchenware tools. An example showing a typical case where a cucumber is being chopped is reported in Fig. 4. In the case of cooking action, there are several ways in which a given action (e.g. cutting) can be performed, depending on the intended result (e.g. "Julienne", "Brunoise", "Chiffonade", etc...). The actions may be distinguishable only by detecting very small movements and can be very difficult to recognize in first person shots. Moreover, a cooking action (e.g. peeling) can be performed differently depending on the food it is applied to either in the movements required as well as in the use of kitchenwares. For example, a banana is peeled differently from an apple.

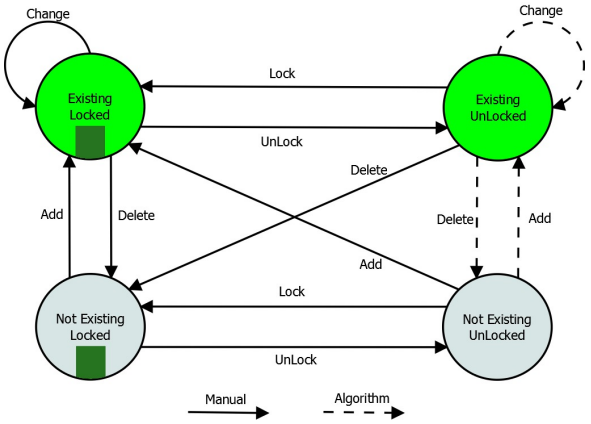


Fig. 3. Finite state machine describing the interactive annotation of an item

Fig. 4. How food changes appearance during cooking. In this sequence a cucumber is being finely chopped.

In *i*VAT, recognition of cooking actions is obtained by observing informative spatio-temporal features [6], such as: periodic motions and spatio-temporal corners. Periodic motions usually occur in simple actions such as body movement, facial expressions etc. . . , while spatio-temporal corners occur when a moving object inverts its motion direction, such as in the case of cutting a carrot, peeling a potato etc. . .

We considered actions that occur in a stack of subsequent images $I(x, y, t)$ of spatial dimensions $X \times Y$ and temporal dimension T. The response function used to detect spatio-temporal region of interest is defined as follows [6]:

$$R = (I * g * h_{ev})^2 + (I * g * h_{od})^2, \tag{1}$$

where $g(x, y; \sigma)$ is the 2D Gaussian smoothing kernel, applied only along the spatial dimensions, and $h_{ev}(t; \tau, \omega)$ and $h_{od}(t; \tau, \omega)$ are a quadrature pair of 1D Gabor filters applied temporally with $\omega = 4/\tau$:

$$h_{ev}(t;\tau,\omega) = -\cos(2\pi t\omega)e^{-t^2/\tau^2},$$
$$h_{od}(t;\tau,\omega) = -\sin(2\pi t\omega)e^{-t^2/\tau^2}. \quad (2)$$

Parameters τ and σ correspond roughly to the spatial and temporal scale of the detector. The response function is thought to give more importance to periodic motions as well as to moving corners associated to complex movements, and less importance to translational motions. For each interest point, that is the local maxima of the response function, a 3-dimensional region of interest is extracted. Such a region, also called cuboid, has two dimensions proportional to the parameter σ, and the third proportional to τ. Inside the cuboid is where the interesting motion is happening. The values of the two parameters must be such that most of the interesting motion is inside the cuboid.

Given a cuboid, the brightness gradient is calculated at each spatio-temporal location (x, y, t) so obtaining three channels (G_x, G_y, G_t) each having the same size as the cuboid. Finally, the feature vector is obtained by dividing the cuboid in sub regions and computing, for each of them, a local histogram of gradients, so following the same strategy as described by Lowe for 2D SIFT descriptors [12]. The final descriptor is obtained after a PCA dimensionality reduction procedure.

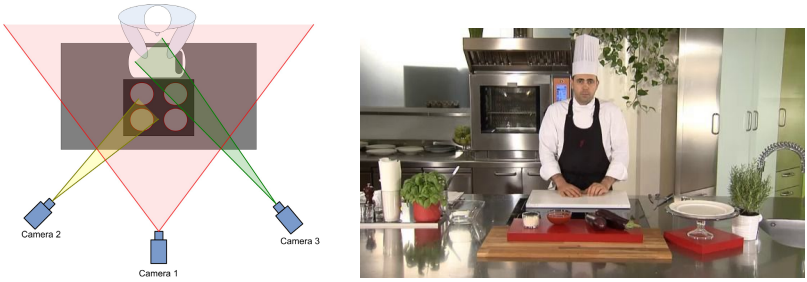
The training data consists of several type of actions and multiple instances for each type. The method described above extracts a large number of cuboids for each action instance, so the total number of cuboids is virtually unlimited. To reduce such a number, the notion of cuboid prototype has been introduced. This notion is based on the idea that although two instances of the same action may have a different visual appearance, such instances may have many of their interest points similar. A library of cuboid prototypes is obtained for the whole training set by using the k-means algorithm.

In the testing phase, each cuboid extracted from the test set is compared with prototypes measuring the similarity, based on the Euclidean distance, and so it is assumed to be one of the known types or rejected as an outlier.

3.1 Experimental Results

The cooking videos have been acquired in a professional kitchen with stainless steel worktop. The videos have been recorded by professional operators using three cameras: one central camera which recorded the whole scene with wide shots, and two side cameras for mid shots, medium close ups, close ups, and cut-ins. A schematic representation of the acquisition set-up is shown in Fig. 5.

In the video, close-ups and wide shots are intermixed. Wide shots are used during explanations of the cooking steps while close up are used to illustrate the current step (e.g. carrot peeling and cutting, vegetables boiling, etc. . .). In wide shots, the items that must be annotated are of a small size, which makes them difficult to recognize. On close-up shots the items are more recognizable but due to their handling they are often occluded by kitchenwares or by the hands of the cook. All these facts make it very difficult to correctly annotate the items throughout the entire video shots. Moreover, differently from other domains, the



(a) (b)

Fig. 5. Shooting set-up. a) Disposition of the digital cameras with respect to the kitchen worktop. b) An example of frontal shot captured by the Camera 1.

Table 1. Action sets used in the experiment

Action category	**#Sequences**
Oiling	30
Mincing	21
Cutting	64
Salting	24
Peppering	22
Peeling with a knife	14
Peeling with a potato peeler	14
Brushing	14
Coring	11
Total	214

cooking domain presents particular challenges such as frequent occlusions and food appearance changes.

After having analysed the different cooking videos, we have identified nine elementary actions that we are interested in recognizing: *Oiling*,*Mincing*, *Cutting*, *Salting*, *Peppering*, *Peeling with a knife*, *Peeling with a potato peeler*, *Brushing*, and *Coring*. These actions have been selected since they are present in different videos and thus a suitable training set can be constructed for each action. We have limited the selection and recognition of the actions in the shots captured by the camera 1 (i.e. the wide angled shots) since in these shots the actions are clearly visible. We found that, in our dataset, close-up shots are captured with very narrow angles and thus part of the actions often occur out of the frame.

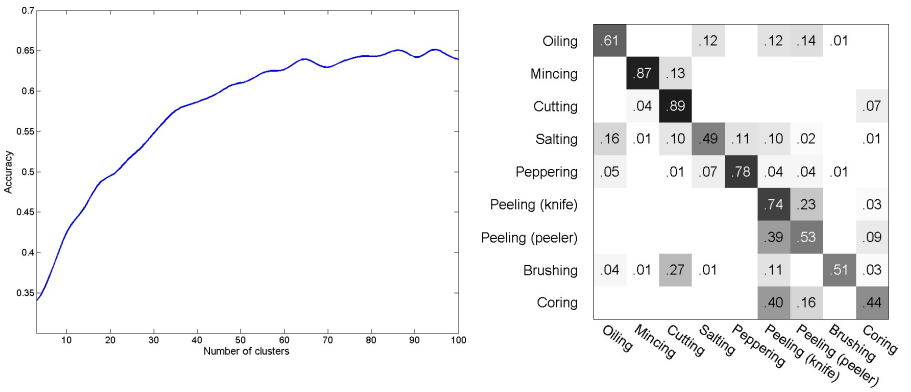
Table 1 summarizes the number of sequences belonging to each action's category. These sets are used for training and evaluation of the action recognition algorithm. Each action sequence is a crop of the original video which includes the hands of the cook. An example of action sequence is shown in Fig 6.

The best scale parameters σ and τ are found by a grid search on the parameter space $\sigma \in [4.0, 6.5]$ and $\tau \in [2.0, 4.0]$. The parameter space has been sampled with a 0.5 step in each dimension, for a total of 50 sample points. The grid search

Fig. 6. An example of a Peppering action sequence

has been done once and off-line and the total time required was about 2 hours on a i5 processor. Ten independent runs are made by randomly extracting for each action category an equal number of videos for training and testing with a maximum of 10 videos each. For each run, all the parameters configurations are tested varying the number of clusters K ($K \in [3, 200]$). The parameter combination (σ, τ, K) giving the higher mean accuracy over the 10 runs is then selected.

The plot of the mean accuracy over the 10 independent runs for the best scale parameters $(\sigma, \tau) = (6.5, 2)$ varying the number of clusters K is reported in Fig. 7(a). The plot is cropped at $K = 100$, when the accuracy starts to decrease. The mean confusion matrix over the independent runs for the best parameters configuration $(\sigma, \tau, K) = (6.5, 2, 95)$ is reported in Fig. 7(b).



(a) (b)

Fig. 7. Plot of the mean accuracy over the 10 independent runs for the best scale parameters varying the number of clusters K (a). Average confusion matrix over the 10 independent runs for the best scale parameters with the best number of clusters (b).

4 Conclusions

In this work we presented the results of cooking action recognition with *i*VAT. *i*VAT is a interactive video annotation which integrates different computer vision modules for object detection, tracking, and action recognition within an incremental learning framework that allows to increase the accuracy of the underlying algorithms over time. The cooking domain is particularly challenging for both object detection and action recognition task, since during the recipe foods change their visual appearance and actions may be distinguishable only by detecting very small movements. The preliminary results obtained for the task of cooking action recognition are encouraging. More results for both the action recognition and for the object detection task will be presented in next publications, together with a detailed description of the object detectors included and the incremental learning approach used in *i*VAT. Although we have developed *i*VAT for the annotation of cooking video, the underlying framework can be easily adapted and used to annotate videos from different domains as well. We plan to experimentally show this in future works.

Acknowledgements. *i*VAT has been developed as part of the R&D project "Feed for good" coordinated by Almaviva and partially supported by Regione Lombardia. The University of Milano Bicocca is supported by Almaviva. The authors thank Almaviva for the permission to present this paper.

References

1. Ali, K., Hasler, D., Fleuret, F.: Flowboost: Appearance learning from sparsely annotated video. In: 2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1433–1440 (2011)
2. Bianco, S., Ciocca, G., Napoletano, P., Schettini, R., Margherita, R., Marini, G., Gianforme, G., Pantaleo, G.: A semi-automatic annotation tool for cooking video. In: Image Processing: Machine Vision Applications VI, vol. 8661, p. 866112. SPIE (2013)
3. Brooke, J.: SUS: A Quick and Dirty Usability Scale. In: Jordan, P.W., Thomas, B., Weerdmeester, B.A., McClelland, I.L. (eds.) Usability Evaluation in Industry. Taylor & Francis, London (1996)
4. Ciocca, G., Schettini, R.: An innovative algorithm for key frame extraction in video summarization. Journal of Real-Time Image Processing 1, 69–88 (2006)
5. Dalal, N.: Histograms of oriented gradients for human detection. In: Computer Vision and Pattern Recognition (CVPR), pp. 886–893 (2005)
6. Dollar, P., Rabaud, V., Cottrell, G., Belongie, S.: Behavior recognition via sparse spatio-temporal features. In: Proceedings of the 14th International Conference on Computer Communications and Networks, pp. 65–72 (2005)
7. Hashimoto, A., Mori, N., Funatomi, T., Mukunoki, M., Kakusho, K., Minoh, M.: Tracking food materials with changing their appearance in food preparing. In: 2010 IEEE International Symposium on Multimedia (ISM), pp. 248–253 (2010)
8. Ji, Y., Ko, Y., Shimada, A., Nagahara, H., Taniguchi, R.I.: Cooking gesture recognition using local feature and depth image. In: Proc. of the ACM Multimedia 2012 Workshop on Multimedia for Cooking and Eating Activities, pp. 37–42 (2012)

9. Kavasidis, I., Palazzo, S., Di Salvo, R., Giordano, D., Spampinato, C.: A semi-automatic tool for detection and tracking ground truth generation in videos. In: Proceedings of the 1st International Workshop on Visual Interfaces for Ground Truth Collection in Computer Vision Applications, pp. 6:1–6:5. ACM (2012)
10. Liao, S., Zhu, X., Lei, Z., Zhang, L., Li, S.Z.: Learning multi-scale block local binary patterns for face recognition. In: Lee, S.-W., Li, S.Z. (eds.) ICB 2007. LNCS, vol. 4642, pp. 828–837. Springer, Heidelberg (2007)
11. Lienhart, R., Maydt, J.: An extended set of haar-like features for rapid object detection. In: Int. Conference on Image Processing (ICIP), pp. 900–903 (2002)
12. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. Int. J. Comput. Vision 60(2), 91–110 (2004)
13. Mihalcik, D., Doermann, D.: The design and implementation of viper (2003)
14. Schiele, B.: A database for fine grained activity detection of cooking activities. In: Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), CVPR 2012, pp. 1194–1201 (2012)
15. Torralba, A., Russell, B., Yuen, J.: Labelme: Online image annotation and applications. Proceedings of the IEEE 98(8), 1467–1484 (2010)
16. Viola, P., Jones, M.J.: Rapid object detection using a boosted cascade of simple features. In: 2001 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 511–518 (2001)
17. Vondrick, C., Patterson, D., Ramanan, D.: Efficiently scaling up crowdsourced video annotation. Int. J. Comput. Vision 101(1), 184–204 (2013)
18. Yao, A., Gall, J., Leistner, C., Van Gool, L.: Interactive object detection. In: 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3242–3249 (2012)
19. Yuen, J., Russell, B., Liu, C., Torralba, A.: Labelme video: Building a video database with human annotations. In: 2009 IEEE 12th International Conference on Computer Vision, pp. 1451–1458 (2009)