Computer Vision and Image Understanding 131 (2015) 88–99

Contents lists available at ScienceDirect

Computer Vision and Image Understanding

journal homepage: www.elsevier.com/locate/cviu

An interactive tool for manual, semi-automatic and automatic video annotation

Simone Bianco [1], Gianluigi Ciocca [1], Paolo Napoletano [*,1], Raimondo Schettini [1]

DISCo (Dipartimento di Informatica, Sistemistica e Comunicazione), Università degli Studi di Milano-Bicocca, Viale Sarca 336, 20126 Milano, Italy

ARTICLE INFO

Article history:
Received 2 December 2013
Accepted 4 June 2014

Keywords:
Interactive video annotation
Automatic annotation
Semi-automatic annotation
Incremental learning
Object detection

ABSTRACT

The annotation of image and video data of large datasets is a fundamental task in multimedia information retrieval and computer vision applications. The aim of annotation tools is to relieve the user from the burden of the *manual* annotation as much as possible. To achieve this ideal goal, many different functionalities are required in order to make the annotations process as automatic as possible. Motivated by the limitations of existing tools, we have developed the *i*VAT: an *interactive* Video Annotation Tool. It supports manual, semi-automatic and automatic annotations through the interaction of the user with various detection algorithms. To the best of our knowledge, it is the first tool that integrates several computer vision algorithms working in an *interactive* and *incremental learning* framework. This makes the tool flexible and suitable to be used in different application domains. A quantitative and qualitative evaluation of the proposed tool on a challenging case study domain is presented and discussed. Results demonstrate that the use of the semi-automatic, as well as the automatic, modality drastically reduces the human effort while preserving the quality of the annotations.

1. Introduction

In recent years several video annotation tools have been developed with a twofold aim of reducing the *human effort* necessary to generate ground truth of large scale visual datasets and improving the *annotations quality*. Most of the tools proposed in the literature include computer vision and machine learning methods that support humans to annotate more efficiently [1–8], while some promote the use of crowd-sourcing based platform to improve the quality of the annotations [9–11].

The different tools can be characterized depending on the functionalities they support. All the annotation tools allow the user to locate in a frame an object of interest by drawing a boundary around it. The most basic, and easily drawn, boundary shape is a rectangle but different tools support other shapes as well such as ellipses and polygons. The most advanced tools also allow the drawing of the boundaries with the aid of semi automatic algorithms such in the case of [4,10]. Although these boundaries should be, theoretically, drawn on every frame in the video sequence, it is often useful, in order to reduce the human efforts, to annotate only few frames (i.e. key frames) and then propagate the annotation by meaning of dedicated algorithms. Almost all the tools considered here incorporate a form of basic annotation propagation exploiting the visual coherence of neighbor frames. The most efficient (in terms of computation time) propagation strategy is based on a simple linear interpolation of the boundaries of an object between a starting position and ending one. More advanced strategies exploit tracking algorithms to explicitly locate instances of the same object across different frames (e.g. [4,5]). Tools that support the annotator with algorithms that accomplish these elementary computer vision tasks have demonstrated to be quite effective in terms of the number of user interactions, user experience, usability, accuracy and annotation time [9]. Since speed-up and simplify the annotation process is of paramount importance in these tools, they often include mechanisms to easily browse the video frames, shots, and make available different modes with which users can interact with the tool (i.e. a graphical user interface, short-cuts, mouse actions, etc.).

Notwithstanding these basic functionalities, the final objective of the annotation tools is to relieve the user from the burden of the *manual* annotation as much as possible. To achieve this ideal goal, computer vision methods are often included in existing tools to support automatic or semi-automatic video annotation. The most recent trend is the integration of algorithms that accomplish more complex computer vision tasks, such as supervised object

* Corresponding author.
E-mail addresses: bianco@disco.unimib.it (S. Bianco), ciocca@disco.unimib.it (G. Ciocca), napoletano@disco.unimib.it (P. Napoletano), schettini@disco.unimib.it (R. Schettini).

[1] The authors contributed equally to this work.

http://dx.doi.org/10.1016/j.cviu.2014.06.015
1077-3142/

detection, template matching, action recognition, event detection, and advanced object tracking [1,2,4,12]. In particular, the use of supervised object detection algorithms allows the automatic annotation the different objects of interest. However, one of the main drawbacks of these algorithms is that they are often domain specific, and must be heavily trained to have a robust detection. Template matching algorithms on the other hand, can be readily used to detect specific instances of the objects but may lack the robustness necessary to detect objects that often change their appearance. For all these reasons, a more efficient approach could be the integration of different annotation modalities to give the user a flexible tool able to work efficiently well across different application domains. At the same time the tools incorporating object detection algorithms, should provide a mechanism for expanding their knowledge of the domain (such as incremental learning algorithms), in order to iteratively increase their efficacy.

Table 1 summarizes and compares recent video annotation tools found in the literature. We have identified some properties that we consider very important in a video annotation tool. The properties refer to the tool's design, user interactions, and basic and advanced annotation functionalities. These properties are:

- Platform: the tool is a Web-based or Desktop application?
- Programming language: what programming languages have been used in the development of the tool?
- Cross-platform: can the tool be used on different platforms?
- Object's boundary shape: what kind of shapes can have a boundary?
- List of objects: the tool allows the annotation of a given list of objects?
- Object's attributes: other information are associated to an object's identity?
- Time-line of objects: does the tool support time-line visualization?
- Temporal reference: are the annotations temporally referenced in the visualization?
- Frames navigation: does the tool supports a navigation within frames?
- Shots navigation: does the tool supports the extraction and navigation of video shots?
- Range-based operations: does the tool support annotation operations on a range of frames/objects?
- Key Frames: the tool supports the annotation of key frames?
- Template matching: does the tool support semi-automatic annotation through the use of template matching algorithms?
- Annotation propagation: what kind of annotation propagation mechanism is included in the tool?
- Supervised object detection: does the tool support automatic annotation through supervised object detection algorithms?
- Incremental learning: does the tool support an incremental learning mechanism?
- Cooperative annotation: is cooperative/crowd sourcing annotation supported?
- Cross domain: can the tool be extended to work on different domains?
- Evaluation tool: does the tool includes an evaluation module to assess annotation quality?

As it can be seen, each tool possesses a set of important functionalities and properties but lacks others also important for the annotation task. For this reason, we developed *i*VAT, an interactive annotation tool that supports the user during the annotation of videos, and that integrates different computer vision modules for object detection and tracking. Among its main features there is the support of three different annotation modalities: manual, semi-automatic and automatic. It also integrates an incremental learning mechanism. To the best of our knowledge, it is the first tool that integrates several computer vision algorithms working in an *interactive* and *incremental learning* framework. This makes the tool flexible and suitable to be used in different application domains.

Previous versions of this tool have been presented in [7,13]. With respect to previous works the main contributions of this paper are:

- an in-depth state of the art analysis is presented and discussed;

Table 1
Existing video annotation tools comparison. With the bullet we indicate that the given tool owns the specific functionality, with the circle we indicate the opposite, while with the minus we indicate that information on that functionality is not provided. The half filled circle stands for not completely integrated functionalities (see Section 4.3).

	VATIC [9]	ViPER-GT [1]	FLOWBOOST [2]	LabeLME V [3]	GTTOOL [4]	GTVT [5]	GTTOOL-W [10]	Inter OD [6]	iVAT (current version)
Platform	Web based	Desktop	–	Web based	Desktop	Desktop	Web based	Desktop	Desktop
Programming language	Html/JS/ Python	Java	–	–	–	VS.NET C#	VS.NET C#	–	C/C++/Qt
Cross-platform	•	•	–	•	–	○	•	–	•
Object's boundary shape	Rect	Rect/Ellip/ Poly	Rect	Poly	Poly/Active Cont.	Rect	Poly/Active Cont.	Rect	Rect/Ellip/ Poly
List of objects	•	○	–	○	○	•	○	○	•
Object's attributes	•	•	○	•	•	○	•	○	•
Time-line of objects	○	•	○	○	○	○	○	○	•
Temporal reference	○	•	•	•	•	○	○	○	•
Frames navigation	•	•	–	•	•	○	•	•	•
Shots navigation	○	○	–	○	○	○	○	○	•
Range-based operations	○	•	–	○	○	○	○	○	•
Key Frames	•	•	•	•	○	•	○	•	•
Template matching	○	○	○	○	•	○	○	○	•
Annotation propagation	Lin. Interp.	Lin. Interp.	Time based reg.	Homogr. Pres.	Tracking	Tracking	○	○	Lin. Interp.
Supervised object detection	○	•	•	○	○	○	○	•	•
Incremental learning	○	○	•	○	○	○	○	•	•
Cooperative annotation	•	○	○	○	○	○	•	○	◐
Cross-domain	•	•	–	•	•	○	•	•	•
Evaluation tool	○	•	○	○	○	•	•	•	•

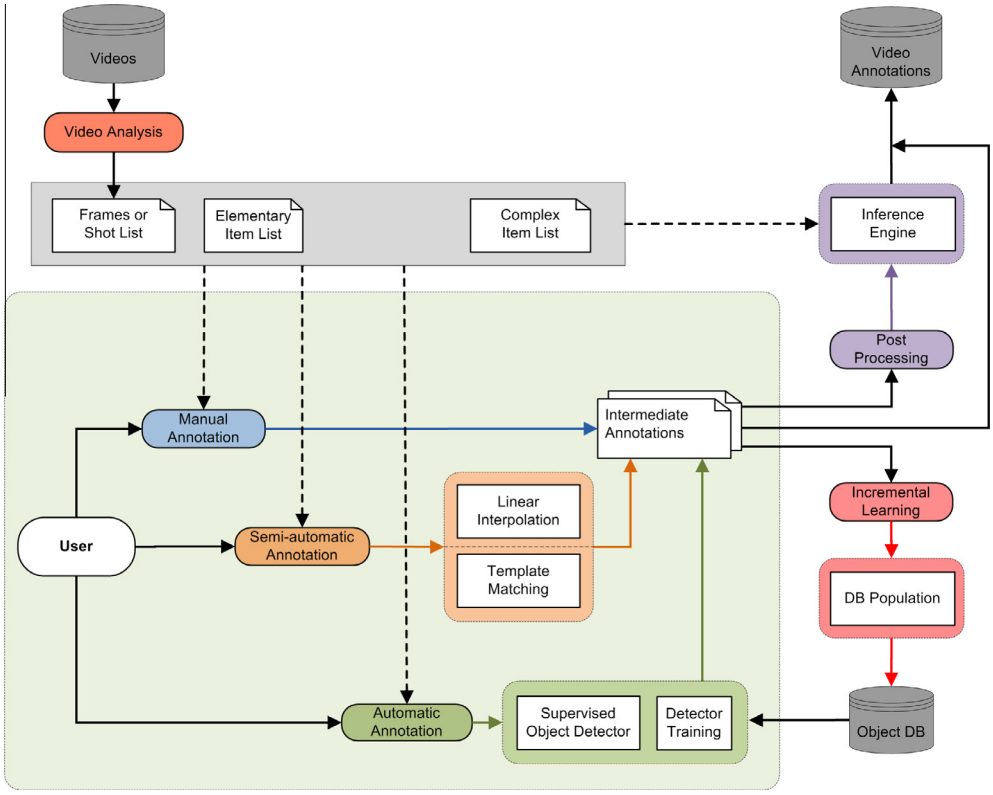

Fig. 1. The *i*VAT overview.

– object's boundary shape now includes polygon and ellipse;
– several supervised object detectors have been trained through the incremental learning paradigm;
– a quantitative evaluation module is now included to assess the annotation quality and the human effort;
– qualitative and quantitative evaluations and comparisons in a particularly challenging domain are presented and discussed;
– highlights on how *i*VAT enables the creation of large scale annotated datasets.

We organized the paper as follows. First, in Section 2, we describe the rationale behind the development of the *i*VAT tool, the modules that compose it, and the different annotation modalities and their interactions. In Section 3, we present the use of the tool in a particularly challenging domain: the annotation of cooking videos. A quantitative (i.e. accuracy of the annotation) and qualitative (i.e. user's usability satisfaction) evaluations of the tool are presented in Section 4. Finally, Section 5 concludes the paper.

2. *i*VAT overview

The scheme of the *i*VAT annotation tool is presented in Fig. 1. It has been developed using C/C++, Qt libraries [14] for the GUI and Open Computer Vision libraries [15] for computer vision algorithms, and is available on-line on the project web page.[2] Qt libraries have been chosen to make the tool platform independent and available also as a mobile application.

*i*VAT handles a video annotation session as a project. A *video analysis* module divides the input video into shots either by automatically detecting them from an Edit Decision List file (EDL) provided as input, or through a shot detection algorithm (e.g. [16] or [17]). Each annotation session must be associated to one or more list of items (i.e. objects of interest) to be annotated that are provided as text files during the project creation procedure. Since items can be of different nature, to facilitate the annotation process, the tool allows to group items into categories. Moreover, the tool allows to handle:

- *elementary item*: an item whose identity is a unique name and recognizable independently of other items (e.g. "Pedestrian", "African Elephant", "Chair", etc.);
– *complex item*: an item whose name and identity depend on the relation between two or more elementary items (e.g. if in a frame has been annotated a horse and above the horse has been annotated a person this may hint to a "horse-jockey" or "riding" annotations).

As it can be seen in Fig. 1, the tool supports different annotation modalities: *manual*, *semi-automatic* and *automatic*. The user can perform a fully manual annotation by providing, frame by frame, the annotations of the items of interest. To minimize user's interaction, the tool provides a semi-automatic annotation modality. With this modality, an initial annotation of the desired item is required at a given frame. Automatic algorithms will then provide the remaining annotations. Currently the tool includes a *linear interpolation* and a *template matching* algorithm. Finally, a fully automatic annotation can be obtained by activating annotation algorithms that provide, frame by frame, the annotations of selected items. Semi-automatic annotation can be virtually applied to any elementary item independently of the application domain. On the contrary, fully automatic annotation with *supervised object detectors* can be used to annotate only those elementary items for which a *learned* model have been created in advance and are domain dependent. Currently, *i*VAT supports a set of different supervised object detector algorithms that can be used at choice

[2] www.ivl.disco.unimib.it/research/ivat/.

by the user. The labeled examples needed to train the object detectors can be obtained from previous annotations. This allows the tool to exploit an *incremental learning* strategy: object instances can be added to the *object database* in order to either increase the robustness of existing object detectors or train new detectors. It is very important to point out that the object detectors perform real-time annotations. This makes supervised algorithms very compatible with the interactive paradigm of this tool.

A previous version of the tool incorporated a supervised action recognition module [13]. Results obtained on several simple actions were very encouraging. However, we decided to exclude this module from the current version, because of the high computational cost required for both the training and the prediction phases, that makes, clearly, this module incompatible with the interactive paradigm of the tool.

The resulting annotations of the elementary items (that we call *intermediate annotations*) can be either retained as is or *post-precessed* to automatically derive annotations for the complex items. This phase may be achieved by submitting all the annotations to an *inference engine* (e.g. a word predictor [18]) that augment the intermediate annotations with new ones. Inference rules specifically designed for the given application domain are required and should be provided in advance.

2.1. User Interface

The whole 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 (at the left side), list of items (at the right side), and a video browser (at the center) which allows the user to browse the shots and seek through their frames. The currently annotated items are shown in the displayed video frame.

The list of shots allows to jump at a specific time in the video sequence by simply clicking on the shot's description. The list of items is organized in a three-level hierarchy. At the top level, the items are divided into *annotated* and available (*list*) sets. The available set contains all the items that can be annotated for the specific domain (i.e. the ones contained in the item's list associated to the current project), while the annotated one contains all the already annotated items within the whole video sequence. Each set is further divided into item's categories, and finally, each category contains all the items belonging to it. The hierarchy structure speeds up the selection of the item of interest while clearly separating the different items from each other.

An annotated item is enclosed by a colored (dashed or solid) boundary drawn on the current frame shown in the central panel of the GUI. Although three different kinds of boundaries can be drawn (rectangular, circular/ellipsoid, and polygonal) to best fit the item's true boundary as shown in Fig. 2, the most widely supported boundary shape, is the rectangle (namely a bounding box, *bbox*). However, we plan to further expand out tool by also including semi-automatic boundary extraction algorithms such as [19]. The different colors represent the different item's categories. Solid boundaries stand for annotations that have been manually obtained, while dashed boundaries stand for annotations obtained by semi- or automatic algorithms.

While the video sequence can be browsed frame by frame, a video player is also embedded into *i*VAT (commands are positioned at the bottom of the video frame) which allows to display the frames sequentially with the annotations superimposed to them. This is especially helpful when dealing with moving objects in order to check if they have been annotated correctly following the video dynamics.

The lower half of the window is dedicated to the annotation time-lines. Every time a new item is annotated, a new time-line

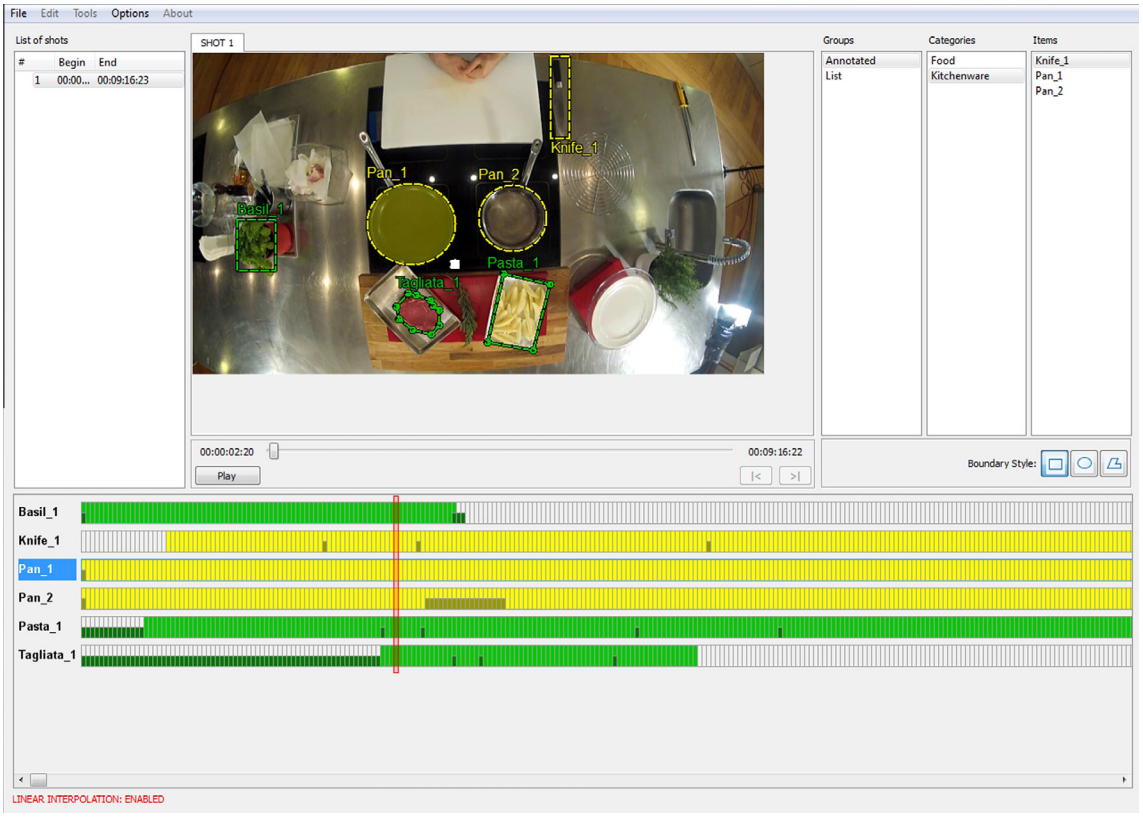

Fig. 2. The *i*VAT GUI.

is added. Each time-line allows the user to visually keep track of where the corresponding item has been annotated. Each time-line is divided into cells of the color of the item's category. Each cell in the time-line corresponds 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 by an algorithm, and if the annotation cannot be changed. Fig. 3a illustrates an example of time-line. A filled cell (green this case) indicates the presence of the item in the corresponding frame, while an empty cell (gray) corresponds to its absence. A marker superimposed on a cell (dark green in the example) means that the annotation can only be modified by the user (see next section for more details).

The vertical red bar is synchronized with the video browser, and can be used to easily navigate to a specific position within the video frames. By selecting a time-line, the bounding box corresponding to the item is highlighted and vice versa. This allows to easily locate an item if the annotations on the frame became cluttered. Another way to deal with a densely annotated frame, is to temporarily hide unwanted annotations. To facilitate the annotation process, the user can interact with the tool in different ways: using *click-able buttons*, *drag* & *drop* operations, *context menus* and keyboard *short-cuts*.

In order to illustrate all the functionalities of the proposed tool, a video demonstration is provided at the *i*VAT project's web page [20].

2.2. Annotation modalities

Since the annotation of an item can be either manually or automatically obtained, 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) annotations. This concept is related to a specific item at frame *f*. If the annotation at frame *f* has been provided or modified by the user, then the state of the annotation, independently of the presence at frame *f* of the item, is locked. On the contrary if the annotation at frame *f* has been provided or modified by an algorithm, then the state of the annotation, independently of the presence at time *f* of the item, is unlocked. As a safeguard, algorithms can changes annotations only if they are not locked. Only the user can modify the state of an annotations changing it from locked to unlocked and vice versa. Fig. 3b shows the finite state machine that describes all the possible interactions between manual and automatic change of annotations. The appearance of the states visually reflect the states of the cells in the time-lines.

2.2.1. Manual annotations

A manual annotation of an item 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 boundary around the object. Manually annotated items are identified by boundaries with a solid outline. The size and position of the boundary can be changed at any time as well as deleting it. The color of the boundary depends on the item's category. By design all the manual annotations are locked.

2.2.2. 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 boundaries. These annotations are also not locked. Linear interpolation currently supports only rectangular boundaries.

2.2.3. Automatic annotations

Automatic annotations can be obtained by different supervised algorithms embedded in the tool. 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

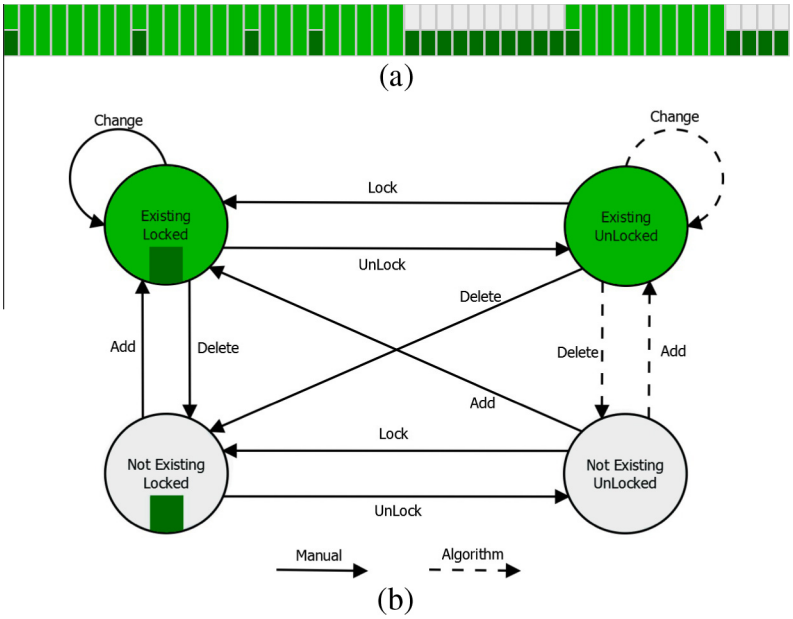

Fig. 3. An excerpt of an item's annotation time-line (a). Finite state machine describing the interactive annotation of an item (b).

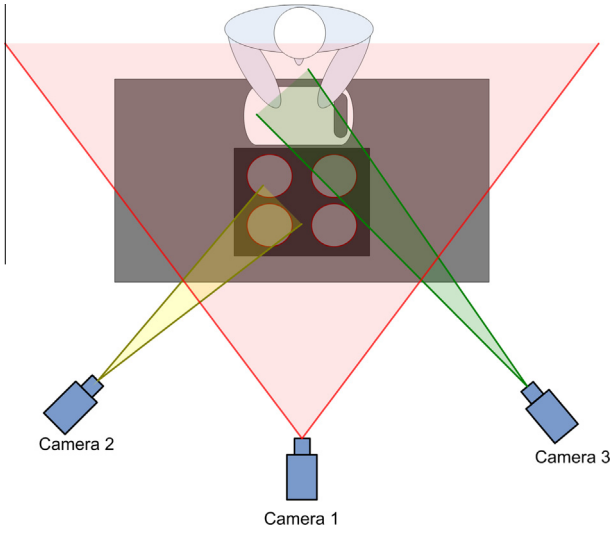

Fig. 4. Disposition of the digital cameras with respect to the kitchen worktop.

later for training the supervised object detector. Different supervised object detection algorithms are currently implemented employing a cascade of boosted classifiers working with Haar-like features [21,22], Histograms of Oriented Gradients (HOG) features [23], and Local Binary Patterns (LBP) features [24]. In this modality the annotations are shown with dashed bounding boxes since only rectangular boundaries are supported, and these annotations are not locked.

3. A case study domain: cooking video

As a case study, we show the usage of *i*VAT in the context of the cooking video domain. 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. Our tool have been used within the Feed For Good project[3] which aims at promoting a better nutrition awareness. The role of *i*VAT was to provide video recipes annotations exploiting computer-vision techniques in recognizing ingredients, and kitchen wares.

The recipes videos have been acquired by the Feed For Good team 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 setup is drawn in Fig. 4.

With respect to other domains, the cooking domain presents particular difficulties. For instance, during the recipe preparation, foods may heavily 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. 6. Not only the cucumber undergoes an appearance change but also the scale in which it is recorded changes as well. This makes the design and selection of computer vision algorithms for the cooking domain particularly challenging [25–27]. In this domain, automatic algorithms are expected to fail or give wrong results in some cases. However, the integrated functionalities of *i*VAT can help the user to overcome them by providing tools to correct these errors.

The video recipes are HD quality videos with a vertical resolution of 720 pixels (1280 × 720) at 25 frames per second and compressed in MPEG4. The videos were acquired with the aim of being esthetically pleasing and useful for the final user. The shooted videos were video edited to obtain the final videos. The edited videos are a sequence of shots suitably chosen from those captured by the three cameras in order to clearly illustrate the steps in the recipe. Fig. 5 shows a visual summary of the "Tegame di Verdure" recipe (the summary has been extracted using the algorithm in [17]).

4. Tool evaluation

To assess the effectiveness of the *i*VAT tool, we performed three analysis (both qualitative and quantitative). The first analysis is quantitative and aims to compare the *human effort* needed to annotate items contained in the videos by using all the three modalities available in the *i*VAT tool: manual, semi-automatic, and automatic. The second analysis, that is both qualitative and quantitative, compares the usability of the *i*VAT and the ViPER-GT [1] tools on a number of video annotation sessions performed by different users. The third analysis highlights how the *i*VAT tool enables the annotation of large scale datasets.

4.1. Quantitative evaluation

For the quantitative evaluation of the *i*VAT we considered a subset of videos extracted from the cooking domain described in the previous section. It has been done using an evaluation tool that is part of the *i*VAT and is illustrated in Fig. 7. The tool takes as inputs the user and the ground truth annotations. Once the annotations have been parsed, the tool computes statistics about annotation quality [28] and about user interaction. More in detail, we used 18 video recipes. Each recipe is about 10 min long and thus contains about 15,000 frames. The total number of frames to be annotated is about 270,000. The subset of videos has been chosen by considering just those videos that includes a suitable number of items in common. In fact, to use the automatic modality we needed to train a detector for each item considered and thus we needed object samples. For this reason, the set of videos containing a given object have been split in two equal parts. One part has been used to learn the corresponding object detector and the remaining for the tool evaluation. The object detectors used are the cascade classifiers on LBP features, since they give better results for a wider range of items of the domain considered.

For each video recipe a *reference annotation* has been obtained through the full manual (M) modality. The same videos has been annotated through the semi-automatic (SA) and automatic (A) modality. The quantitative analysis is done by comparing the number of user interventions needed in SA and A modality to obtain annotations that are identical to those obtained in full M modality.

Fig. 8a represents a possible item time-line generated in full M modality. Full M annotation mode requires the user to annotate the item on each frame in which the item is present for a total of 35 interventions out of 49 total frames, which corresponds to a 0.71 ratio. A possible time-line of an item annotated in SA mode is reported in Fig. 8b. In SA mode a total of 7 user interventions (corresponding to a 0.14 ratio) are needed in order to obtain an annotation equivalent to that obtained in M mode: the first five are aimed to lock as existing the item in frames 1, 9, 16, and 20; the last two are respectively aimed to lock as not existing the item in the frames from 26 to 35 and from 46 to 49. A possible time-line of an item annotated in full A mode is reported in Fig. 8c. In A mode a total of 4 user interventions (corresponding to a 0.08 ratio) are

[3] http://www.eservices4life.org/projects?view=project&task=show&id=9.

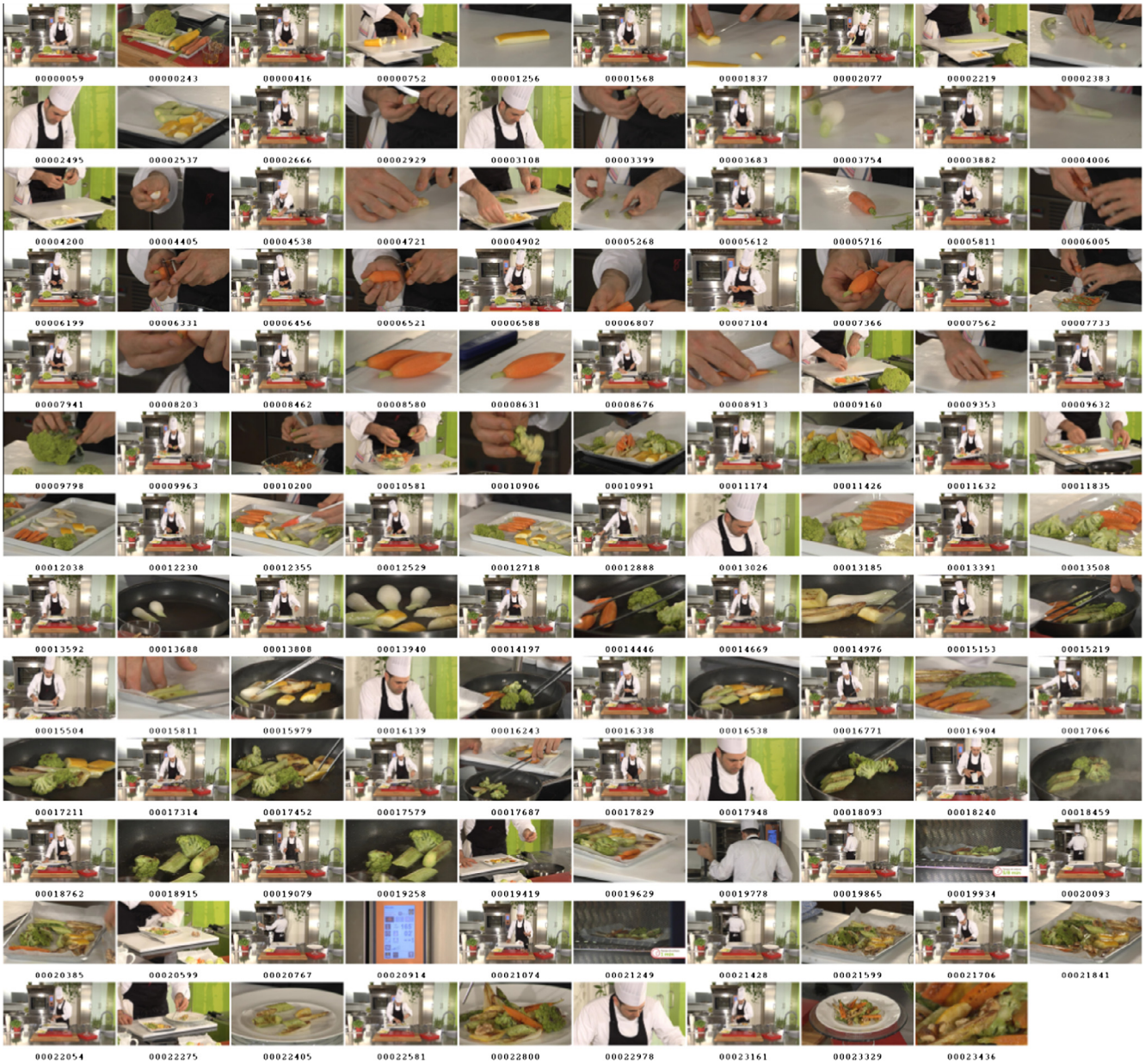

Fig. 5. Visual summary of the video sequence "Tegame di Verdure".

Fig. 6. How food changes appearance during cooking. In this sequence a cucumber is being finely chopped.

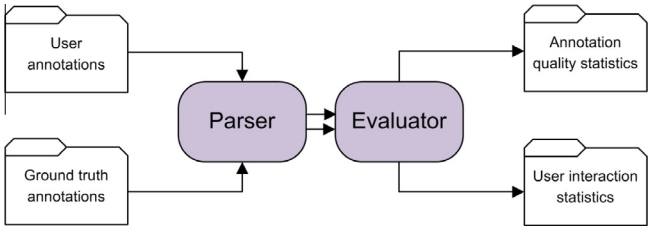

Fig. 7. Graphical representation of the evaluation tool.

needed in order to obtain an annotation equivalent to that obtained in M mode. The first three are aimed to set the item in frames 1, 16, and 20 as existing, while the fourth one to set the item in the frames from 26 to 35 as not existing.

For the A annotation mode, the experiments have been done using cascade classifiers on LBP features. Two important parameters have to be set for the cascade classifier: the former is the scale factor s_f, which specifies how much the image size is reduced at each image scale; the latter is the number of neighbors n_n each candidate rectangle should have to retain it. The best choice for the parameters makes the A annotations more reliable and thus less probable the human intervention to correct them. For this reason, for each parameter a grid search has been done to determine the best parameters combination: ten different values have been considered for the scale factor in the range $[1.05, 1.50]$ with 0.05 steps, and ten for the minimum neighbors number, i.e. $n_n = [1, \ldots, 10]$.

Classification performance are evaluated in terms of F1 score [29]. Averaging the F1 score over the different items and videos considered for each parameter combination, produces the surface drawn in Fig. 9a (this approach is similar to [30]). It is possible to notice that combinations with both higher s_f and n_n tend to give less accurate detections, resulting in a lower F1 score. The combination $[s_f, n_n] = [1.05, 4]$ results on average the best combination. Furthermore the slope of the surface with respect to s_f is lower than that with respect to n_n, indicating a lower dependence of the detector performance from the choice of s_f than from the choice of n_n. In order to have an idea of what is the worst-case performance for each parameter combination, the surface obtained plotting the lowest F1 across the items considered is drawn in Fig. 9b. Again, the best performances are obtained for combinations with both higher s_f and n_n. As additional information in Fig. 9c the surface of the ratio standard deviations is reported.

In Fig. 10 the ratio of human interventions in SA mode is compared to that in full A mode. The results are individually reported for a subset of annotated items. For the full A mode two different bars are reported: the former using the global best parameter combination (i.e. $[s_f, n_n] = [1.05, 4]$), the latter using the best parameter combination for each item, i.e. the combination giving the highest F1 score for each item. The corresponding numerical values are reported in Table 2, where the ratio of human interventions in full manual mode are also reported.

Analyzing Table 2 we can see that both the SA and the A annotation modes are able to decrease the average ratio of human interventions of at least one order of magnitude with respect to the full M annotation mode. The A mode with the best choice of parameters for each item improves the ratio by 2.67% on average with respect to the global best parameter choice on the items reported. On average the SA annotation mode requires 4.80% less human interventions with respect to A mode with global best parameters, while 2.12% with respect to automatic mode with best parameters for each item. A more detailed analysis reveals that there are items on which the difference between A and SA modes is low, and there are also cases in which the A mode remode requires less interventions than the SA one. Since the SA annotation mode employs a linear interpolation algorithm to propagate the item positions across the frames, it results particularly convenient when the item moves along a piece-wise linear path (see for example the green track in Fig. 11). For items with more complex motion, the full A annotation mode results more convenient (see for example the red track in Fig. 11). This means that the use of both the SA and A modality makes the tool flexible and suitable to be used in different application domain.

4.2. System usability: comparison with the ViPER-GT tool

Fourteen users tested both our tool and the ViPER-GT tool annotating 10 different video sequences, each about 1 min long. After having trained the users in the use of both tools, we asked them to process a set of video sequences by annotating items from a given list. Each user annotated half of the video sequences using the *i*VAT tool and the other half using the ViPER-GT tool, so that a video sequence is annotated by the same user exactly once. We ensured that each video sequence is equally annotated by both tools. We recorded each annotation session using both a screen-capture software and a keyboard/mouse capture software. The recorded videos are used to analyze the user's behavior during the annotation process, while the keyboard and mouse actions are used to evaluate the interactions. Before starting the annotation session, the tool's window is maximized to full screen. After the annotation sessions, we administered to the users a questionnaire in two parts in order to collect their impression about the usability of each tool and its functionalities. The first part was inspired by the System usability Scale (SUS) questionnaire developed by John Brooke at DEC (Digital Equipment Corporation) [31]. It is composed of statements related to different aspects of the experience, and the subjects were asked to express their agreement or disagreement with a score taken from a Likert scale of five numerical values: 1 expressing strong disagreement with the

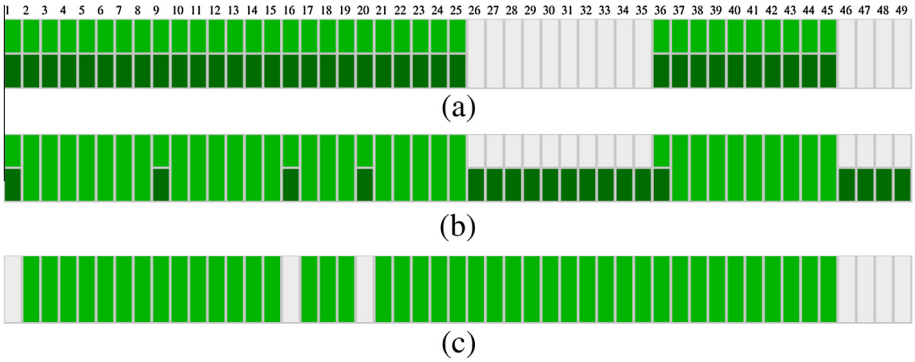

Fig. 8. Time-lines of an annotated item in manual mode (a), semi-automatic mode (b), and automatic mode (c).

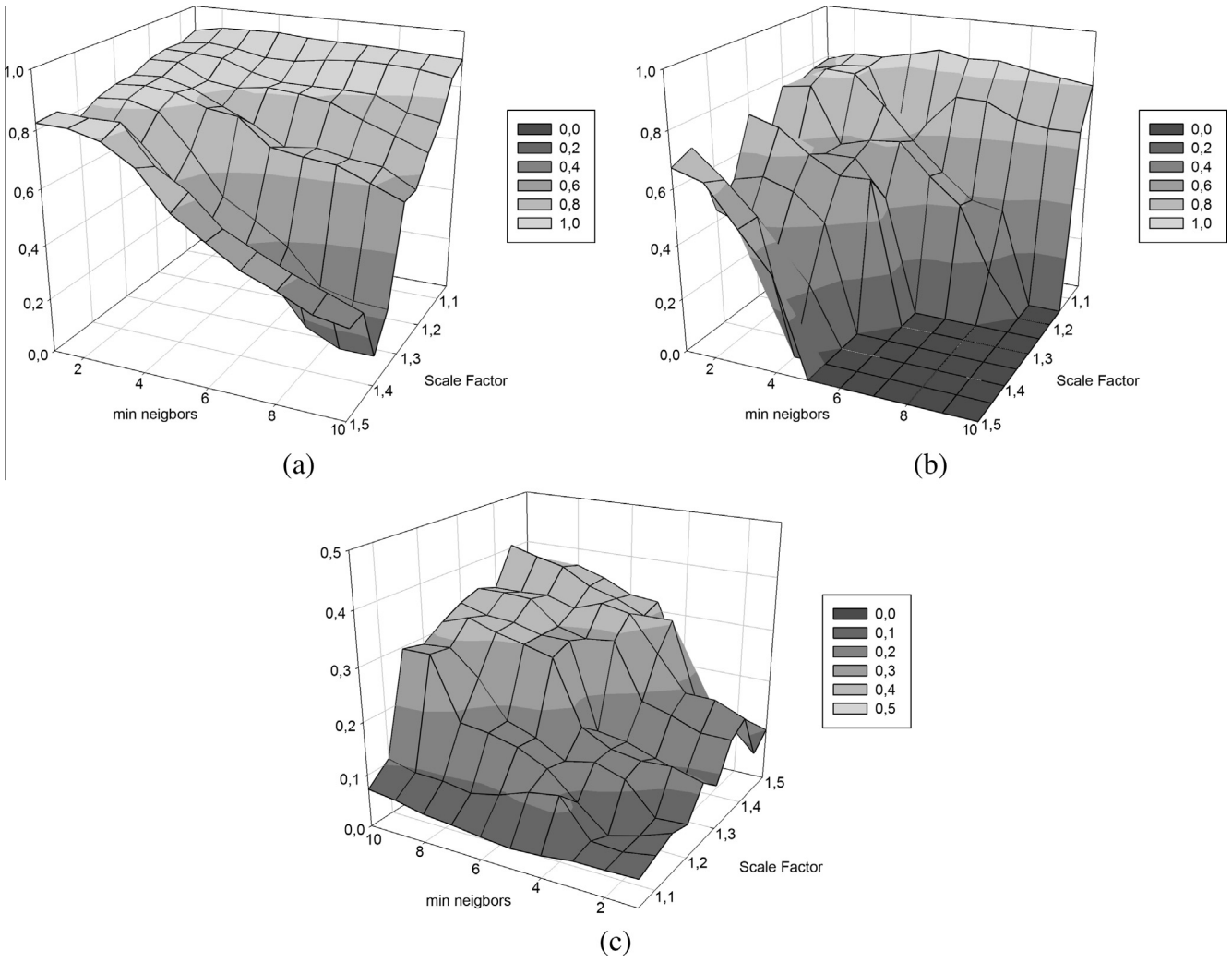

Fig. 9. F1 scores in full automatic mode: average (a), minimum (b), and standard deviation values (c) over the different items and videos considered for each (s_f, n_n) parameter combination.

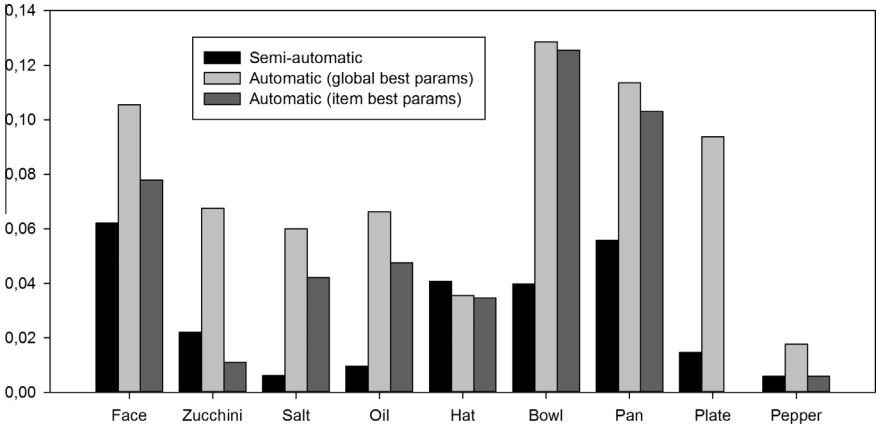

Fig. 10. Ratios of user interventions in semi-automatic and full automatic annotation modes with respect to the number of frames. The average results are individually plotted for a subset of annotated items (lower is better).

statement, 5 expressing strong agreement and 3 expressing a neutral answer. The second part of the questionnaire focuses more on the functionalities of the annotation tool and was administered with the same modalities.

The results of the questionnaire for both tools are reported in Table 3. The score given by the users are summarized by taking the median of all the votes. The best results are reported in bold. With respect to the usability, it can be seen that the *i*VAT tool

Table 2
Ratios of user interventions with respect to the number of frames. The average results are individually reported for a subset of annotated items (lower is better).

Item	Manual	Semi-automatic	Automatic[a]	Automatic[b]
Face	0.3697	0.0621	0.1055	0.0779
Zucchini	0.6966	0.0221	0.0676	0.0110
Salt	0.7427	0.0062	0.0600	0.0421
Oil	0.7903	0.0097	0.0662	0.0476
Hat	0.5178	0.0408	0.0356	0.0347
Bowl	0.8161	0.0398	0.1286	0.1255
Pan	0.6730	0.0557	0.1136	0.1030
Plate	0.5029	0.0146	0.0938	0.0000
Yellow Pepper	0.7744	0.0059	0.0176	0.0059
Average	0.6537	0.0285	0.0765	0.0497

[a] Using the global best parameters.
[b] Using the best parameter combination for each item.

has been rated positively for all the first ten statements. On the overall, the system has been considered easier to use than ViPER-GT (a score of 4 for *i*VAT against 2.5 for ViPER-GT). Moreover, the system has been considered less cumbersome to use than ViPER-GT (a score of 2 for *i*VAT against 4.5 for ViPER-GT).

With respect to the tool functionalities, the scores show that the *i*VAT tool is again judged positively. From the users' responses, it can be seen that the interactive mechanism can efficiently support the annotation of the videos. The semi-automatic algorithms, although not very precise in the annotation, can give a boost in the annotation time and require only few corrections to obtain the desired results. The best rated functionalities of the *i*VAT tool are the keyboard short-cuts and the graphical user interface. The short-cuts allow the users to interact with the system with mouse and keyboard simultaneously increasing the annotation efficiency. The graphical interface is easy to understand and allows to keep track on the annotated items with clear, visual hints. While designing the interface we were worried that the amount of displayed information would have been too much. An interview with the users proved the contrary. However, as suggested by the users, the time-line panel although useful (as is in the ViPER-GT) should be further improved. Although intuitive and easy to understand, the user considered to be useful to add the capability to zoom-in and zoom-out the time-line as is in the ViPER-GT. When the number of frames is very large, they felt very tedious to scroll the panel back and forth in order to seek the desired frame interval. A resizable time-line that could be made to fit the size of the panel would be a welcome addition to the tool. With respect to the time-line, the worst result of the ViPER-GT tool is obtained in the interactions required to edit the annotations. User found the use of the markers cumbersome. A comparative video showing side by side an annotation session performed with *i*VAT and ViPER-GT is available online [20].

Table 4 shows on average the interactions performed by the users to annotate the videos. As a measure of the manual efforts, we compute the percentage of manually drawn bounding boxes. As an indication of the user interactions with the graphical interface of the tool, we computed also the number of mouse clicks,

Fig. 11. Starting and ending frames of a video sequence with two tracks overlayed: an item on which it is more convenient to use the semi-automatic annotation mode (i.e. Bowl, green track); an item on which it is more convenient to use the full automatic annotation mode (i.e. Face, red track). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 3
Median scores given to the statements of the questionnaire administered to the users after the annotation sessions. The best scores are reported in bold (1 corresponds to 'strongly disagree', 2 to 'disagree', 3 to 'neutral', 4 to 'agree', and 5 to 'strongly agree').

#.	Statement	iVAT	ViPER-GT
1.	I think that I would like to use this system frequently	**4**	1.5
2.	I found the system unnecessarily complex	**2.5**	4
3.	I thought the system was easy to use	**4**	2.5
4.	I would need the support of a technical person to be able to use this system	**2**	**2**
5.	I found the various functions in this system were well integrated	**4**	2.5
6.	I thought there was too much inconsistency in this system	**1.5**	3.5
7.	I would imagine that most people would learn to use the system very quickly	**3.5**	2.5
8.	I found the system very cumbersome to use	**2**	4.5
9.	I felt very confident using the system	**3.5**	1.5
10.	I needed to learn a lot of things before I could get going with this system	**2.5**	**2.5**
11.	The time-line is not very useful	**1**	2
12.	Too many input/interactions are required to obtain acceptable results	**1**	5
13.	The keyboard short-cuts are useful	**4.5**	1.5
14.	It is difficult to keep track of the already annotated items	**1**	2.5
15.	The new item's bounding boxes can be drawn quickly	**4**	3
16.	I think there is too much information displayed in too many panels	**1.5**	2
17.	The system user interface is easy to understand	**4**	3
18.	Semi-automatic annotation algorithms are too slow	**2**	**2**
19.	I prefer using only manual/basic annotation functionalities	**2**	**2**
20.	I needed to correct many errors made by the semi-automatic annotation algorithms	**2.5**	3

Table 4
Interactions performed by the users to annotate the videos on average. The number in parenthesis is the standard deviation.

	iVAT	ViPER-GT
Percentage of manually drawn bounding boxes	2.53% (±1.70)	5.01% (±4.73)
Number of clicks	61	427
Number of keys pressed	138	133
Distance covered by mouse movements	11,663,178	36,793,244
Annotation time	00:08:21 (±00:04:28)	00:22:18 (±00:17:30)
Bounding-box per second	2.32	0.87

the number of keys pressed and the distance covered by mouse movements.

4.3. Large scale annotation with iVAT

In this section we highlight how the tool enables the generation of large scale annotated datasets. The *i*VAT has been used in the context of the cooking video domain where the aim was to label 54 recipes, each almost 20 min long and edited in shots of about 30 s. Each recipe contains an average of 6 items to be annotated in each frame, for a total of almost 9.72 millions of bounding-boxes. This task has been done by 9 users in parallel, each operating on a independent subset of 6 recipes. The total annotation time was of about 125 h.

A generalization of the task subdivision across users is reported in Fig. 12: the task is parallelized over users by sending to each of them a different part of the video sequence to be annotated, together with the list of the items to be annotated, and the supervised object detectors. Once the users complete the task, they send the annotations back to the server, where they are merged to generate the annotation for each entire video sequence. User annotation could then be exploited to learn supervised object detectors in an incremental learning strategy (as described in Section 2).

Using the subdivision schema reported in Fig. 12, the annotation task can be parallelized at video-level or even at shot-level. The expected time to annotate 9.72 millions of bounding boxes by parallelizing the annotation task at video-level (i.e. using 56 users, one for each video) and at shot-level (i.e. using 2160 users, one for each shot) are reported in Table 5. They are computed as linear predictions from the measured time needed by 9 users, which is reported in bold. Expected annotation times are also reported for smaller and larger datasets, ranging from 1 million up to 100 millions of bounding-boxes.

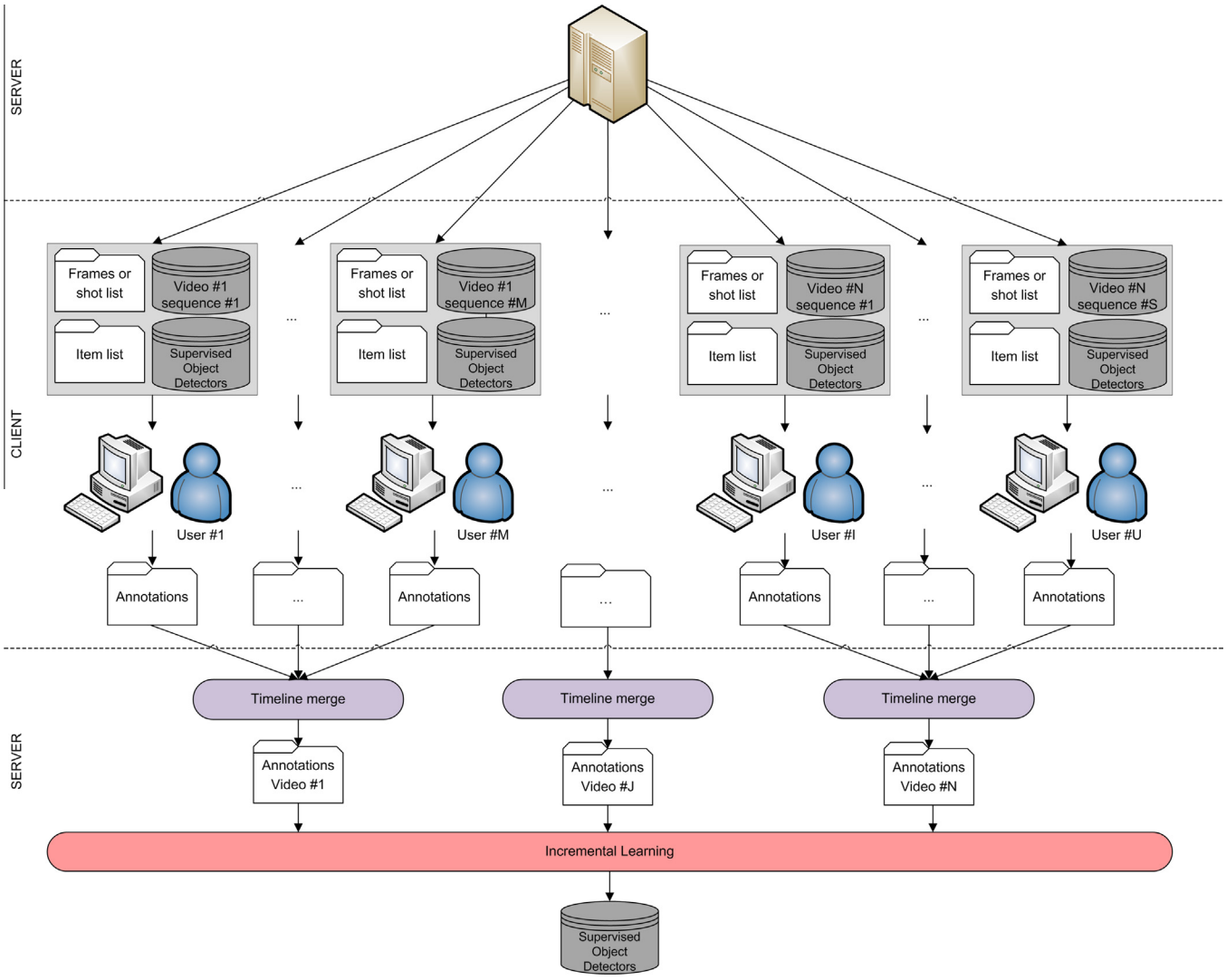

Fig. 12. Schema for the subdivision of the annotation task across users within the *i*VAT.

Table 5
Prediction of the expected times to annotate different dataset sizes with respect to the number of users. The entry in bold corresponds to the time needed to annotate the whole cooking domain in the experimental setup adopted (i.e. task parallalized over 9 users).

# Users	# Bounding boxes to be annotated							
	1 M	2 M	5 M	9.72 M	10 M	20 M	50 M	100 M
1	115:55:40	231:51:20	579:38:20	1126:49:05	1159:16:40	2318:33:20	5796:23:20	11592:46:40
9	12:52:51	25:45:42	64:24:16	**125:12:07**	128:48:31	257:37:02	644:02:36	1288:05:11
54[a]	2:08:49	4:17:37	10:44:03	20:52:01	21:28:05	42:56:10	107:20:26	214:40:52
2160[b]	0:03:13	0:06:26	0:16:06	0:31:18	0:32:12	1:04:24	2:41:01	5:22:01

[a] Task parallelized at video-level.
[b] Task parallelized at shot-level.

5. Conclusion

In this paper, motivated by the limitations of existing annotation tools, we developed *i*VAT, a tool for interactive, semi-automatic and automatic video annotation that integrates computer vision algorithms specifically designed to work in an *interactive* and *incremental* framework. The integration of semi-automatic and automatic annotation modality along with the incremental learning makes the tool flexible and suitable for different application domains.

Quantitative and qualitative evaluations of the proposed tool have been presented and discussed. We compared the *human effort* needed to annotate items contained in the videos of the domain by using all the three modalities available in the tool: manual, semi-automatic, and automatic. The tool is able to decrease the average ratio of human interventions of at least one order of magnitude with respect to the full manual annotation mode. A quantitative and qualitative comparison with ViPER-GT, a state of the art annotation tool, has been carried out. Results demonstrate that *i*VAT outperforms ViPER-GT in terms of both system usability and number of interactions required. Finally, we have highlighted how the tool enables the generation of large scale annotated datasets.

The tool has been developed using C/C++, Qt libraries for the GUI and Open CV libraries for computer vision algorithms. Qt libraries make the tool platform independent and available also as a mobile application. In addition, the modularity of the tool makes the integration of new computer vision modules very easy thus encouraging its expansion. The tool is available on-line at the project web page [20].

As future work we plan to extend the annotation tool to include the automatic and semi-automatic detection of actions and we plan to test the effectiveness of the tool on other application domains.

References

[1] D. Mihalcik, D. Doermann, The design and implementation of viper (2003).
[2] K. Ali, D. Hasler, F. Fleuret, Flowboost: appearance learning from sparsely annotated video, in: 2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2011, pp. 1433–1440.
[3] J. Yuen, B. Russell, C. Liu, A. Torralba, Labelme video: building a video database with human annotations, in: 2009 IEEE 12th International Conference on Computer Vision, 2009, pp. 1451–1458.
[4] I. Kavasidis, S. Palazzo, R. Di Salvo, D. Giordano, C. Spampinato, 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, ACM, 2012, pp. 6:1–6:5.
[5] A. Ambardekar, M. Nicolescu, S. Dascalu, Ground truth verification tool (GTVT) for video surveillance systems, in: ACHI'09, Second International Conferences on Advances in Computer–Human Interactions, 2009, IEEE, 2009, pp. 354–359.
[6] A. Yao, J. Gall, C. Leistner, L. Van Gool, Interactive object detection, in: 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2012, pp. 3242–3249.
[7] S. Bianco, G. Ciocca, P. Napoletano, R. Schettini, R. Margherita, G. Marini, G. Gianforme, G. Pantaleo, A semi-automatic annotation tool for cooking video, Image Processing: Machine Vision Applications VI, vol. 8661, SPIE, 2013, p. 866112.
[8] T. D'Orazio, M. Leo, N. Mosca, P. Spagnolo, P.L. Mazzeo, A semi-automatic system for ground truth generation of soccer video sequences, in: AVSS'09, Sixth IEEE International Conference on Advanced Video and Signal Based Surveillance, 2009, IEEE, 2009, pp. 559–564.
[9] C. Vondrick, D. Patterson, D. Ramanan, Efficiently scaling up crowdsourced video annotation, Int. J. Comput. Vision 101 (1) (2013) 184–204.
[10] I. Kavasidis, S. Palazzo, R. Salvo, D. Giordano, C. Spampinato, An innovative web-based collaborative platform for video annotation, Multimedia Tools Appl. (2013) 1–20.
[11] I. Kavasidis, C. Spampinato, D. Giordano, Generation of ground truth for object detection while playing an online game: productive gaming or recreational working?, in: 2013 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), IEEE, 2013, pp 694–699.
[12] A. Yao, J. Gall, C. Leistner, L. Van Gool, Interactive object detection, in: 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2012, pp. 3242–3249.
[13] S. Bianco, G. Ciocca, P. Napoletano, R. Schettini, R. Margherita, G. Marini, G. Pantaleo, Cooking action recognition with iVAT: an interactive video annotation tool, in: 17th International Conference on Image Analysis and Processing (ICIAP 2013), Lecture Notes in Computer Science, vol. 8157, Springer, Berlin/Heidelberg, 2013, pp. 631–641.
[14] Qt framework. <http://qt-project.org>.
[15] Open computer vision libraries – opencv. <http://opencv.org>.
[16] A.F. Smeaton, P. Over, A.R. Doherty, Video shot boundary detection: Seven years of {TRECVid} activity, Comput. Vis. Image Understan. 114 (4) (2010) 411–418.
[17] G. Ciocca, R. Schettini, An innovative algorithm for key frame extraction in video summarization, J. Real-Time Image Process. 1 (2006) 69–88.
[18] C.D. Manning, H. Schütze, Foundations of Statistical Natural Language Processing, vol. 999, MIT Press, 1999.
[19] N. Xu, N. Ahuja, R. Bansal, Object segmentation using graph cuts based active contours, Comput. Vis. Image Understan. 107 (3) (2007) 210–224.
[20] iVAT Project Web Page. <www.ivl.disco.unimib.it/research/ivat/>.
[21] P. Viola, M.J. Jones, Rapid object detection using a boosted cascade of simple features, in: 2001 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2001, pp. 511–518.
[22] R. Lienhart, J. Maydt, An extended set of haar-like features for rapid object detection, in: Int. Conference on Image Processing (ICIP), 2002, pp. 900–903.
[23] N. Dalal, Histograms of oriented gradients for human detection, in: Computer Vision and Pattern Recognition (CVPR), 2005, pp. 886–893.
[24] S. Liao, X. Zhu, Z. Lei, L. Zhang, S.Z. Li, Learning multi-scale block local binary patterns for face recognition, in: International Conference on Biometrics (ICB), 2007, pp. 828–837.
[25] A. Hashimoto, N. Mori, T. Funatomi, M. Mukunoki, K. Kakusho, M. Minoh, Tracking food materials with changing their appearance in food preparing, in: 2010 IEEE International Symposium on Multimedia (ISM), 2010, pp. 248–253.
[26] B. Schiele, A database for fine grained activity detection of cooking activities, in: CVPR '12, Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2012, pp. 1194–1201.
[27] Y. Ji, Y. Ko, A. Shimada, H. Nagahara, R.-i. Taniguchi, Cooking gesture recognition using local feature and depth image, in: Proc. of the ACM multimedia 2012 workshop on Multimedia for cooking and eating activities, 2012, pp. 37–42.
[28] R. Kasturi, D. Goldgof, P. Soundararajan, V. Manohar, J. Garofolo, R. Bowers, M. Boonstra, V. Korzhova, J. Zhang, Framework for performance evaluation of face, text, and vehicle detection and tracking in video: Data, metrics, and protocol, IEEE Trans. Pattern Anal. Mach. Intell. 31 (2) (2009) 319–336.
[29] C. van Rijsbergen, Information Retrieval, Butterworth, 1979.
[30] N. Lazarevic-McManus, J. Renno, D. Makris, G. Jones, An object-based comparative methodology for motion detection based on the *f*-measure, Comput. Vis. Image Understan. 111 (1) (2008) 74–85.
[31] J. Brooke, SUS: a quick and dirty usability scale, in: P.W. Jordan, B. Thomas, B.A. Weerdmeester, I.L. McClelland (Eds.), Usability Evaluation in Industry, Taylor & Francis, London, 1996. <http://www.usabilitynet.org/trump/documents/Suschapt>.