Can our TV robustly understand human gestures?

Real-Time Gesture Localization in Range Data

Adolfo López-Méndez
Technical University of Catalonia (UPC)
Barcelona, Spain
adolf.lopez@upc.edu

Josep R. Casas
Technical University of Catalonia (UPC)
Barcelona, Spain
josep.ramon.casas@upc.edu

ABSTRACT
The 'old' remote falls short of requirements when confronted with digital convergence for living room displays. Enriched options to watch, manage and interact with content on large displays demand improved means of interaction. Concurrently, gesture recognition is increasingly present in human-computer interaction for gaming applications. In this paper we propose a gesture localization framework for interactive display of audio-visual content. The proposed framework works with range data captured from a single consumer depth camera. We focus on still gestures because they are generally user friendly (users do not have to make complex and tiring movements) and allow formulating the problem in terms of object localization. Our method is based on random forests, which have shown an excellent performance on classification and regression tasks. In this work, however, we aim at a specific class of localization problems involving highly unbalanced data: positive examples appear during a small fraction of space and time. We study the impact of this natural unbalance on the random forest learning and we propose a framework to robustly detect gestures on range images in real applications. Our experiments with offline data show the effectiveness of our approach. We also present a real-time application where users can control the TV display with a reduced set of still gestures.

Categories and Subject Descriptors
[**Interactive media and games**]

General Terms
Human Computer Interaction, Random forest, gesture recognition, object detection, range data

Keywords
Human Computer Interaction, Random forest, gesture recognition, object detection, range data

1. INTRODUCTION
Digital television is definitely out of production studios and into the consumer market. True digital convergence provides increased flexibility for accessing a large variety of content from the living room instead of from the office desktop. Emerging technologies and concepts such as format-agnostic production or layered scene description enlarge the range of options being offered to the viewer. With such increased offer and flexibility, the old remote, designed to command station and volume adjustments, fails to meet requirements of increased interaction. Tablet displays have been successful introducing a new paradigm in gesture based interaction (e.g. muti-touch [1]) to counter the absence of screen pointers, but living-room displays rather require touch-less interaction.

Gesture localization and recognition methods are increasingly taking an important role within human-computer interaction systems. In fact, these methods are becoming part of commercial systems for interactive creation and display of media content [12] or gaming applications [7]. One of the key elements towards their applicability in real world problems is the irruption of low-cost or consumer depth cameras, which have paved the way to cope with critical illumination problems of classical vision approaches.

In situations when motion does not play a semantic role, gesture detection and localization is a particular case of an object detection and localization problem. Take for instance hand digits; motion is present when changing from one digit to another, but it does not provide any meaning. Indeed, one is interested in retrieving the location of the hand poses contained in an alphabet of gestures. This kind of gestures turn out to be very convenient for interactive applications, since they can effectively reduce user fatigue, i.e., to avoid the *gorilla arm* syndrome. A challenge of such a gesture localization problem is that natural scenes present cluttered backgrounds and moving objects, and on top of that, still gestures may usually appear as objects of small size, and thus represented by a few pixels.

1.1 Related work
The state-of-the-art basically focuses on recognition rather than on localization. Kollorz et al. [6], recognize still hand gestures by using a Time-of-Flight camera. In their approach, they use projections of the image and optional depth features in combination with a nearest neighbor classifier. Similarly, Ren et al. [11] recognize hand gestures with a modified Earth Mover's Distance. Both approaches are limited by its strong dependence on the hand segmentation, thus being prone to errors under clutter or complex scenes.

Regarding object detection and localization methods, random forests and their variants have attracted the attention of the image processing and computer vision community [2, 14, 5] due to its excellent performance for classification and regression tasks. Demirdjian and

Chenna [3] proposed a temporal extension of random forests for the task of gesture recognition and audio-visual speech recognition. However, as before, their gesture recognition approach strongly relies on spatial and temporal segmentation. In fact, this approach is not shown to work properly in real applications, since experiments are performed on temporally and spatially segmented sequences. Shotton et al. [13] use random forests on range data to detect body parts. Their ultimate goal is to estimate human poses, which at the same time may pave the way towards pose-based gesture recognition. However, their approach relies on a dense labeling of the human body in order to detect each part. Gall et al. [5] propose a Hough forest framework for dealing with different tasks, including object detection and localization. Although they overcome the problem of the dense labeling, their Hough forest framework is tested on standard datasets where objects of interest (*positive*) have a relatively large size compared to background (*negative*).

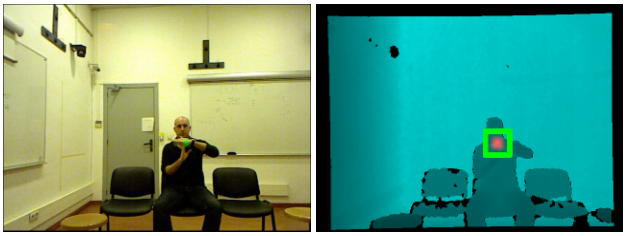
1.2 Proposal

In this paper, we present a gesture localization method based on random forests. Our algorithm relies on depth data captured from a single sensor and does not require a dense labeling of data for training. Furthermore, our method does not require neither segmenting the body parts performing the gesture nor temporal segmentation. The proposed approach admits a broad range of still gestures due to two main reasons. First, one can define gestures with different body parts or sets of body parts. For instance, still gestures can be performed with one hand, two hands or hand and head. Second, our approach can discriminate rather subtle differences between similar gestures. Our contributions are the following:

- Our approach implies a novel formulation of gestures as objects, allowing us to constrain the problem of localization while keeping a high user acceptance and suitable ergonomics.
- We present an efficient tree-wise boosting framework for random forests that addresses highly unbalanced localization problems, where *positive* classes are represented by a few pixels and appear during a few frames.
- We experimentally show that, compared to common training strategies, boosted learning of trees selects the best training samples, thus preventing from manually selecting the best training patches per class. Compared to other boosting strategies for random forests, our approach achieves higher generalization performance.
- We propose a depth-invariant gesture localization method that integrates votes casted by the random forest. Our method takes into account temporal consistency of the votes in a very efficient manner.
- We present an online implementation of our approach that serves users to control an interactive display by means of an easy-to-learn reduced set of still gestures. This interactive display is part of the system developed within the European project FascinatE [4].

2. CLASSIFICATION FORESTS

Classification forests are a specific class of random forests [2] designed for classification tasks. Classification forests are an ensemble of m classification trees, which are binary trees. Nodes n in each tree have a learned probability distribution $p_n(c|\mathbf{I}_t, \mathbf{x})$ that reflects how likely is a class c given a pixel $\mathbf{x}$ in the image $\mathbf{I}_t$. These



(a) (b)

Figure 1: Gesture Localization with Random Forests (best viewed in color). (a) A number of votes (green dots) are casted for the target gesture (b) votes are aggregated to estimate a probability density (overlaid in red on the input depth map) and a localization is estimated (green square).

probability distributions are learned by recursively branching left or right down the tree, according to some node-specific weak classifier, until some stopping criteria are met and thus a leaf node is reached. Weak classifiers associated to each node are binary functions of feature vectors obtained from images $\mathcal{I}$. The robustness of forests is based on the combination of several classification trees. Usually, one performs this combination by averaging the distributions over the leaf nodes $\{l_1, \cdots, l_M\}$ reached in all the M trees:

$$p(c|\mathbf{I}_t, \mathbf{x}) = \frac{1}{M} \sum_{m=1}^{M} p_{l_m}(c|\mathbf{I}_t, \mathbf{x}) \tag{1}$$

Each tree is trained separately with a small subset of the training data obtained by sampling with replacement. Learning is based on the recursive splitting of training data into left $\mathcal{L}$ and right $\mathcal{R}$ subsets, according to some binary test f and a threshold θ. The binary test is a function of the feature vector $\mathbf{v}$ obtained from each training example.

At each node, a test f and a threshold θ are randomly generated, and the one that maximizes some criteria is selected. We employ the information gain as a test selection criterion:

$$\Delta IG = -\frac{|\mathcal{L}|}{|\mathcal{L}| + |\mathcal{R}|} H(\mathcal{L}) - \frac{|\mathcal{R}|}{|\mathcal{L}| + |\mathcal{R}|} H(\mathcal{R}) \tag{2}$$

where $|\ |$ denotes the number of elements of the subset and $H()$ is the Shannon entropy of the classes in a subset. The process continues until a maximum depth D is reached or the information gain cannot be further maximized.

2.1 Tests and Features

To build our random tests, we employ the depth-based features proposed in [13]. Specifically, for a given pixel $\mathbf{x}$ the test f has the following expression:

$$f_\theta(\mathbf{I}, \mathbf{x}) = d_\mathbf{I}\left(\mathbf{x} + \frac{\mathbf{u}}{d_\mathbf{I}(\mathbf{x})}\right) - d_\mathbf{I}\left(\mathbf{x} + \frac{\mathbf{v}}{d_\mathbf{I}(\mathbf{x})}\right) \tag{3}$$

where $d_\mathbf{I}$ is the depth map associated to image $\mathbf{I}$ and $\mathbf{u}$ and $\mathbf{v}$ are two randomly generated pixel displacements that fall within a

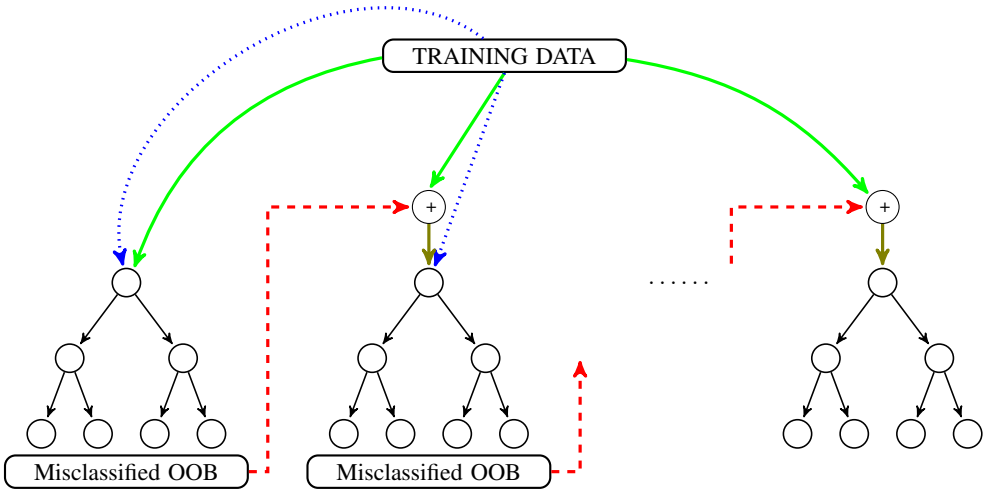


Figure 2: Tree-Wise Boosting approach for Random Forests (best viewed in color). Green thick arrows denote the samples used for training each tree. These samples are obtained by sampling with replacement from the whole training set. Red dashed arrows indicate the samples that have been misclassified by the decision tree. These samples come from the out-of-bag set, indicated by the blue dotted arrows.

patch size. Pixel displacements are normalized with the depth evaluated at pixel $\mathbf{x}$ in order to make the test features invariant to depth changes.

This approach helps in automatically learning the most significant low level configurations of the interest objects for detection.

3. TRAINING DETECTION FORESTS

In the following, we present detection forests as a particular approach of classification forests towards localization of gesture classes in range data. In this work, we study the performance of classification forests trained for all the C classes and also $C-1$ forests trained with each *positive* class against the rest. It has been reported [10] that class-specific forests outperform multiclass detectors, although their computational complexity grows linearly with the number of classes.

We focus on highly unbalanced class distributions, where the classes of interest, or *positive* classes, are a minority in comparison to the *negative* class, both spatially (*positive* classes occupy a few pixels) and temporally (*positive* classes appear in a few frames). These highly unbalanced distributions are found in real world gesture recognition and localization applications. On the one hand, users are not constantly performing gestures, hence, the distribution of gesture classes (*positive*) with respect to non-gesture (*negative*) is biased towards the latter. On the other hand, the actual appearance of a gesture may be represented by a relatively low number of pixels, thus increasing the bias towards *negative* classes. This unbalance makes that low false positive rates constitute actually a large number of false positive votes. Taking into account this phenomenon is highly important during training, since machine learning approaches will generally try to maximize the classification rate, rather than the *true positive* classification rate. This basically means that our objective is to minimize the false positive rate while keeping a high classification rate.

In order to train a random forest, one usually samples a subset of the training data for the construction of each tree. If one employs this method on highly unbalanced datasets, the forest learning is clearly biased. To address this problem, one can sample a small fraction of the data and ensure that each class obtains approximately the same number of training samples, i.e., the distribution at the root node is approximately uniform. We compared this approach with a weighted random forest scheme [14] based on weighting the *positive* examples with the inverse class frequency. Balancing the number of training samples presented a better performance since the unbalance is so large that weighted random forest overpowers the response on positive classes, thus increasing the false positive rate.

While balancing reduces the bias, it is still a common procedure for training the forest. Indeed, balancing the training samples generally presents the drawback of sampling highly correlated *positive* samples while loosely sampling the *negative* class. This drawback has a greater impact on class-specific learning schemes, where loose sampling has the effect of missing relevant samples of the *negative* class that may effectively reduce the false positive rate. In order to overcome this problem, we propose a boosted learning scheme. In [5], boosted learning is applied to Hough forests. In their approach, a forest of 15 trees is trained by first learning a sub-forest of 5 trees and then picking a fixed number of positive and negative samples, consisting in the samples that are harder to classify. These samples are used to train the next 5 trees, and the whole process is repeated in order to train the remaining 5 trees. In this paper, we propose a different approach to boost the trees. Our approach is designed in a tree-wise manner: in [5] base learners are *subforests*, while in our approach base learners are decision trees. Hence, in our training approach, each tree is tested with training data separately, while in [5] several trees are evaluated as a forest and hence, they can be tested with training data multiple times.

Our method performs training of detection forests as follows (see Fig. 2). We train the first tree with a balanced set of samples from

each class. Once the tree is trained, we evaluate it against the out-of-bag set [2]. The wrongly classified samples are added to the training set of the second tree (up to a maximum number of training samples). This new training set is completed by sampling with replacement from the full training set until balance is achieved. We train the second tree with this training subset and we repeat the process until the forest is fully trained.

In the proposed approach, we do not attempt to balance the wrongly classified samples. This is because the majority of wrongly classified samples are false positives that we want to incorporate into our training subsets, in order to have a more relevant set of *negative* examples. Another difference with respect to [5] is that we exclusively use the out-of-bag set. This increases the efficiency (we evaluate less samples) and, together with the tree-wise approach, allows trees to be more uncorrelated.

4. GESTURE LOCALIZATION

As introduced in [13], and made explicit by Eq. 3, working with depth data allows detection and classification algorithms to deal with world coordinates, avoiding problems related to apparent size due to the projection process in visual cameras. Therefore, analysis, detection and classification of test features is easily made depth-invariant.

For gesture detection and localization, a set of patches are provided to the detection forest, which casts a vote whenever a *positive* class has more probability than the *negative* class and other *positive* classes. Fig. 1 illustrates the casted votes for a positive class in a class-specific learning example. To detect a gesture, we first estimate a probability density using the votes within a frame and we take into account temporal consistency by recursively updating this distribution with votes aggregated from past time instants. In order to construct the probability density, we use a Parzen estimator with Gaussian kernel K:

$$p(c|\mathbf{I}_t) = \sum_i p(c|\mathbf{I}_t, \mathbf{x}_i) K(\mathbf{x} - \mathbf{x}_i) \tag{4}$$

Note that the resulting density is not a probability measure, since it does not integrate to 1. In any case, it allows us to evaluate (up to proportionality) how likely is an image region to be representing a gesture. In order to account for the time component of the approximated density, we sequentially update $p(c|\mathbf{I}_t)$ as follows:

$$p'(c|\mathbf{I}_t) = \alpha p(c|\mathbf{I}_t) + (1-\alpha) p'(c|\mathbf{I}_{t-1}) \tag{5}$$

This is a simple yet effective method to keep temporal consistency of the casted votes, as it requires storing a single probability map. An adaptation rate $\alpha = 0.8$ works well in practice, as it prevents several false positives while avoiding a delayed response.

Finally, we compute the pixel location $\mathbf{g}_c$ of a gesture class $c > 0$ as the pixel location with maximum probability:

$$\mathbf{g}_c = \underset{\mathbf{x}}{\operatorname{argmax}} \;\; p'(c|\mathbf{I}_t) \tag{6}$$

we ensure that such a maximum represents a target gesture by thresholding the probability volume V computed by locally integrating the estimated pseudo-probability measure :

$$V = \sum_{\mathbf{x} \in \mathcal{S}} p'(c|\mathbf{I}_t(\mathbf{x})) \tag{7}$$

where $\mathcal{S}$ is a circular surface element of radius inversely proportional to the depth, and centered at the global maximum, i.e:

$$\mathcal{S} = \{\mathbf{v} \;\; | \;\; \|\mathbf{v} - \mathbf{g}_c\| < r(d_{\mathbf{I}_t}(\mathbf{g}_c))\} \tag{8}$$

In this way, the localization is depth-invariant.

5. EXPERIMENTAL RESULTS

We conduct experiments on two different setups. Both setups, however, aim at a gesture-based control of a display, an increasingly targeted application for commercial audio-visual systems. The first setup consists of offline recordings. The main objective of the experiments in the first setup is to provide a quantitative performance of our approach in front of existing training methods. The second setup is an online demonstrator. For this second setup, we describe how easily we bring our approach to a real-life application. The qualitative results on this online setup aim to show the potential of our approach. Our method has been developed within the framework of European project FascinatE [4] and therefore, some of the specificities of our experiments aim to respond to the needs of the project. In spite of that, we believe that our still gesture approach towards interaction can be integrated in many different setups having similar objectives.

5.1 Offline setup

The first setup comprises offline processing on a range data dataset recorded with Kinect [7]. The target scenario simulates a gesture-based interface to control a display. The design of this interface considers some additional gestures such as scrolling or zooming the displayed content. For that matter, the data recorded in this setup contains a great variety of negative examples. Our objective is to detect a subset of events or gestures whose objective is to trigger different control modes. In this setup, we focus on three still gestures namely *FingerOnMouth*, *HandOnEar* and *Tee* (see Fig. 3). Their objective is to trigger events such as pause/play or to control the volume, and they involve one or two hands and/or the head. Although it is a small number of gestures, the main challenge is to correctly detect them in front of a potentially huge set of *negative* classes, i.e., distractors such as other gestures or poses that do not trigger events. Consequently, as we will be assessing precision and recall as in a detector, three gestures suffice to quantitatively test our approach.

We record 4 training sequences where a set of gestures and actions, including the 3 target gesture classes, are performed [1]. In each training sequence a single subject performs the gestures with the same hand (right or left depending on the gesture) while sitting or standing up in front of a display. Additionally, we record 2 challenging test sequences containing 4 subsequences. In each subsequence an actor enters the scene, i.e., in the first subsequence

[1]The authors will make the video data and annotations employed in this paper publicly available

Figure 3: Training examples for the offline experiments. Left: *FingerOnMouth* Middle: *HandOnEar* Right: *Tee*.

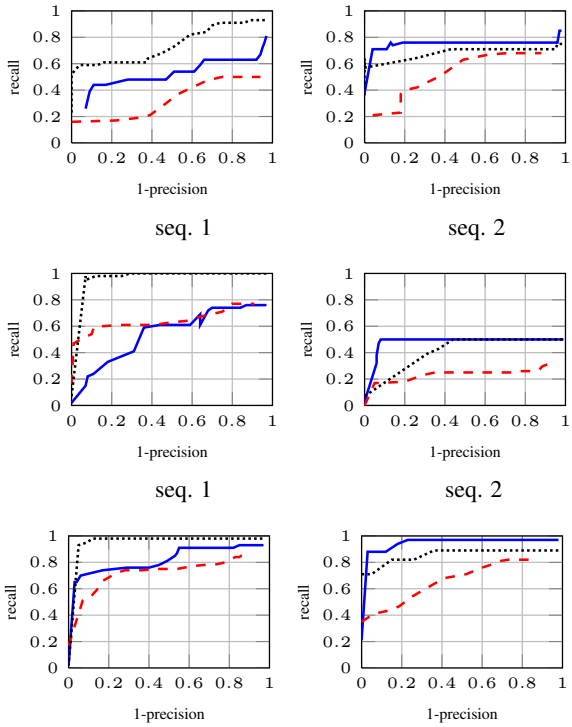


Figure 4: Performance for each gesture class in the class-specific scheme. *Top*: Balanced training. *Middle*: Boosting of forests, as proposed in [5]. *Bottom*: Our boosted training. Target gestures: *FingerOnMouth* (solid blue), *HandOnEar* (dashed red) and *Tee* (dotted black).

only one actor performs the actions, while in the 4-th subsequence 4 actors are present in the scene and performing gestures. In the test sequences, actors perform actions with both the right and left hand (see Fig. 6). Hence, testing data contains great changes in appearance and clutter in comparison to the training data. In overall, we use 11 minutes of range video data for training and 5 minutes of test data. To the best of our knowledge, there is no similar dataset in the literature. Existing datasets focusing on American Sign Language [16, 9] or one-shot learning [8] do not provide the testbed for evaluating the robustness of an approach against clutter and distractors. Also, considering the FascinatE project framework, there is a explicit need for collecting this data for testing purposes.

Using this data, we test our approach. The employed detection forests have 15 trees with maximum depth 20, and each tree is trained with 10000 examples per class. The pixel locations of both training and testing patches are collected in foreground regions.

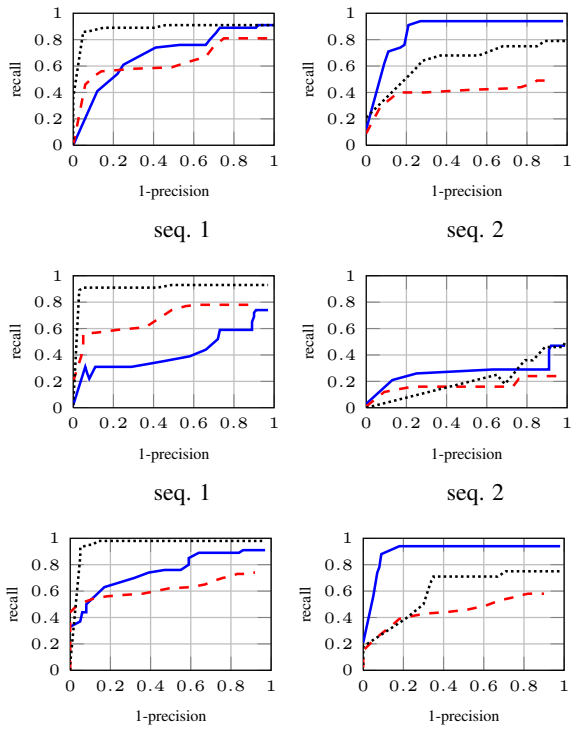


Figure 5: Performance for each gesture class in the multiclass scheme. *Top*: Balanced training. *Middle*: Boosting of forests, as proposed in [5]. *Bottom*: Our boosted training. Target gestures: *FingerOnMouth* (solid blue), *HandOnEar* (dashed red) and *Tee* (dotted black).

Foreground regions are computed by thresholding the depth with a single background image. For testing, foreground regions are densely sampled at 1/16th resolution (1 of every 4 pixels). Note that such a segmentation is in fact not required by our algorithm, whose binary tests work on unsegmented data; the foreground/background segmentation is used to constrain the test regions. In the training phase, we randomly pick a maximum number of patches per class out of this dense sampling. In order to detect gestures performed with the hand that was not in the training set, we simply flip the test patches horizontally.

We train class-specific and multiclass forests with the balanced sampling method and the boosted learning method proposed in this paper. Additionally, we implement a boosted approach based on [5]. Specifically, we employ 10000 patches per class in order to train a forest of 5 trees. Then we evaluate this forest with all the training data, and we collect the 1000 examples per class that were misclassified with more error. For each one of the next 5 trees, we randomly sample 9900 patches per class, in order to complete the training subsets of each decision tree. The process ends when the whole forest is fully trained. In all cases, we employ squared patches of size 85x85 pixels, as we concluded, after quantitative experiments, that this patch size presents an accurate pixel-wise classification.

To measure the accuracy of our method, we consider a correct localization if the estimated gesture and the actual gesture belong to the same class and if the estimated location is within a radius of 10 pixels. We test several thresholds for the pseudo-probability vol-

Figure 6: Examples of successful localization results for the offline experiments. Top row: *FingerOnMouth* Middle row: *HandOnEar* Bottom row: *Tee*.

	Seq. 1			Seq. 2		
Class-Specific	FingerOnMouth	HandOnEar	Tee	FingerOnMouth	HandOnEar	Tee
Balanced	0.49	0.31	0.73	0.74	0.46	0.69
Boosted Forests[5]	0.55	0.59	**0.97**	0.48	0.22	0.46
Our Tree-Wise Boosting	**0.81**	**0.65**	0.95	**0.94**	**0.57**	**0.85**
Multiclass	FingerOnMouth	HandOnEar	Tee	FingerOnMouth	HandOnEar	Tee
Balanced	0.70	0.59	0.87	0.87	0.38	0.68
Boosted Forests[5]	0.42	0.62	0.89	0.29	0.17	0.29
Our Tree-Wise Boosting	0.74	0.57	0.94	0.90	0.41	0.63

Figure 7: Area Under the Curve (AUC) for balanced and boosted learning strategies.

ume V (see Eq. 7) in order to compute the precision and recall per each individual gesture class in the two sequences.

Localization results in the test sequences[2] are shown in Figs. 4, 5 and the corresponding Areas Under the Curve (AUCs) are presented in Fig. 7.

The proposed boosted learning outperforms the balanced learning in the tested cases. Recall that, as mentioned in section 3, balanced learning outperforms the weighted learning scheme proposed in [14]. Surprisingly, we obtained poor results with the boosting scheme inspired by [5]. Despite obtaining a notable accuracy in the first sequence (and even obtaining the best localization accuracy for the *Tee* gesture), it completely fails to localize gestures in the more challenging sequence 2. These results indicate that using *subforests* as base learners is prone to over-fitting when dealing with the huge unbalance posed by real gesture localization applications.

We specifically observe that our boosted learning method notably outperforms other approaches for class-specific training. Such improvement is due to the fact that the number of *negative* samples is greater in the class-specific case (all the other gestures count as *negative*). These results indicate that our tree-wise boosted learning is able to select better training samples when a huge number of them are provided to the method, and thus allows working efficiently with small subsets of the training data.

The individual gesture class performance reveals that the detector can provide excellent results with gestures like *FingerOnMouth* or *Tee*. In contrast, the *HandOnEar* gesture is harder to localize due to variations of pose among subjects. This suggests that such a gesture requires more positive training samples capturing these variations. We also see that the horizontal flipping provides an excellent accuracy in detecting gestures that are performed with a hand that is not in the training set, thus allowing a reduction in the number of training examples.

The best performance is achieved with class-specific detection forests in combination with the proposed boosted learning method. Using this configuration, the average per frame localization accuracy for all the classes and test sequences is 96.8%.

The presented detection forests have been implemented in C++, with no optimization efforts, and the offline detection runs at an average rate of 10 fps with the multiclass scheme in a 2.40GHz CPU. Provided that we have employed 15 forests, with maximum depth 20 and dense sampling, this is a satisfactory computational performance, since one could achieve real-time performance for the class-specific scheme by using less trees, by pruning the trees in depth, by using a sparser sampling and/or by using regions of interest to constrain the test locations.

[2]Videos with sample results can be found in : *https://sites.google.com/site/adolfolopezmendezphd/gesture_localization_project*

5.2 Online Setup

We use our gesture localization approach to develop an online, real-time TV control (see Fig. 8). In this scenario, a user asks for the control by using a still gesture. Once he or she has the control, the user can employ additional gestures in order to control the displayed content. The same gesture employed for taking the control can be also used in order to release the control. For this online demonstrator we require 5 still gestures (see Fig. 9):

- **Tee** : The same gesture employed in the offline experiments (see Section 5.1). It is used by users in order to get/release the control.
- **HandOnEar**: Serves users to raise the volume. Same gesture employed in the offline experiments (see Section 5.1).
- **FingerOnMouth**: Users lower the volume with this gesture. Also employed in the offline experiments (see Section 5.1).
- **Cross**: Employed in order to mute/activate the audio. This gesture consists in crossing the index fingers of both hands.
- **ParallelHands**: This gesture is employed in order to pause or resume the reproduction/streaming of some video content. Users put their hands in parallel and approximately in front of their face in order to perform this gesture.

The diagram in Figure 10 depicts how the online demonstrator works. As abovementioned, the system starts in *idle* mode. In this mode, the audio-visual content is displayed/streamed but users do not have control over it. In order to control it, users have to perform gesture *Tee* in order to switch to command mode. In this mode, users can use the 5 available gestures to control the audio and the video streaming/reproduction, or to simply release the control, allowing any other subject to handle it.

We train class-specific forests for each one of the 5 available gestures. To this end, we record new data where 5 actors perform these gestures, as well as a number of distractors. These 5 actors are recorded from 2 different viewpoints (see Fig. 11). In all the recorded training sequences, actors are standing up.

In this online demonstrator, we do not rely on any kind of background learning. In order to approximately segment the scene into relevant and non-relevant pixels (for the sake of efficiency) we simply threshold the depth using two values z_{near} and z_{far}. Specifically, we set $z_{\text{near}} = 0.8m$ and $z_{\text{near}} = 3.5m$. In addition, to further increase the efficiency of our real-time demonstrator, we employ a head tracking algorithm [15] in order to track a user after he or she gets the control (i.e., in the command mode, see Fig. 10). The estimated head position is used in order to compute a bounding box that restricts the forest evaluation to a reduced number of pixels. In this way, we can run up to 5 class-specific detection forests in real-time without GPU implementation of our algorithms.

Note that the viewpoint of the Kinect sensor in the online demonstrator differs from the training data (see Figs. 8 and 11). Furthermore, in our demonstration, people are sitting most of the time, hence the lower body pose or even the torso inclination differ from the training data.

A video showing the performance of this demonstrator can be found in *http://vimeo.com/43476448*.

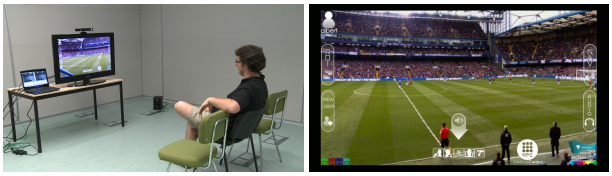


Figure 8: Online demonstrator. Left: Setup; the demonstrator runs in a single laptop (placed next to the TV). Right: Detail of the displayed content and the user feedback. Icons on the left and right margins of the screen are not used for our still gesture demonstrator. The icon on the bottom center (above the bar) is used to give feedback about still gesture commands (the icon shows that the user is lowering the volume). The bar on the bottom shows the 5 available still gestures.

.

Figure 9: Still gestures employed in the online demonstrator. From left to right: Tee, HandOnEar, FingerOnMouth, Cross and ParallelHands.

During our online tests with the presented demonstration, we observe that the system responds accurately and fast. This high performance is especially remarkable for gestures *Tee*, *ParallelHands* and *Cross*, despite *FingerOnMouth* and *HandOnEar* show also a high detection accuracy. In tests with novice users, we observe that these gestures are easy to perform, and thanks to the almost inexisting number of false positives, the application allows to efficiently learn to use them for controling the audio-visual content. Furthermore, we did not observed signs of fatigue, as most of the time users can be in a relaxed position. However, a more thorough usability study has to be undergone in order to assess our first observations on learning curves and fatigue.

Our online system presented some minor fail cases. We observed some sporadic confusion between *Cross* and *Tee* just after taking the control (entering the *Command Mode*). Specifically, some users tilted their *Tee* gesture right after entering the *Command Mode*, and due to this change in the inclination of the gesture, a *mute* event was triggered. This was not critical, since the action can be easily undone by performing an additional *Cross* gesture. Furthermore, by setting an appropriate timeout after entering in the *Command Mode*, this problem would be solved. We also observed some delay for some *HandOnEar* performances. However, all the users succeeded in raising the volume by using this gesture.

Our online system has been installed in several rooms with different screens and beamers, and in particular, it has been successfully tested and reviewed for the FascinatE project. During these project tests, new users were able to quickly learn the proposed gestures and to successfully control displayed audio-visual content.

We deployed the whole online demonstrator in an Intel i7 2.20Ghz laptop, with 8GB RAM. Despite the available cores, we are able to run 4 class-specific still gesture localizations on a single core in real-time, with no optimization efforts. Provided that random forests are highly parallelizable, a GPU implementation might yield super real-time performance.

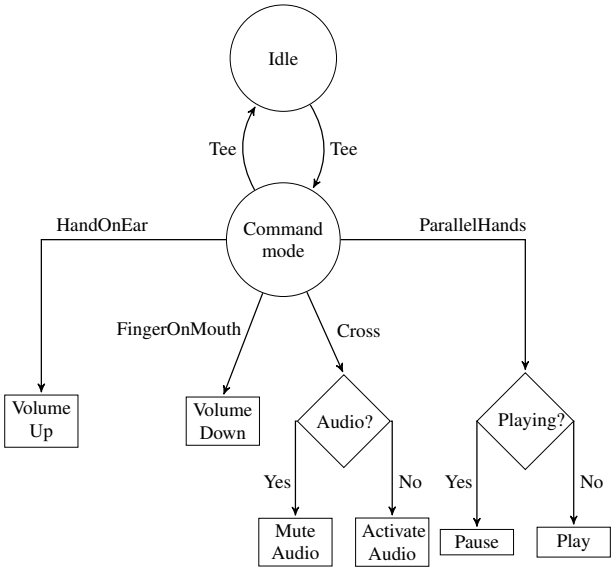


Figure 10: Operation diagram of the online demonstrator. Rounded nodes denote operation modes, while squared nodes denote events.

Figure 11: Viewpoints employed during the training towards the online demonstrator

.

6. CONCLUSIONS

We have presented a novel approach towards gesture localization for interactive displays. We propose a boosted learning framework for random forests, aiming at accurately localizing gesture and object classes in highly unbalanced problems, where *positive* classes barely appear in natural images, thus requiring a special attention with false positives. We have also proposed a depth-invariant method to robustly transform the votes casted by the forests into pixel locations. Our experiments, aiming at real gesture-based applications, show the effectiveness of our method and the robustness against clutter and moving objects. We have presented a real-time demo that has been successfully tested and reviewed for the european project FascinatE.

Provided that we obtained encouraging results on real scenarios, future work involves undergoing a thorough user study. The user study aims to find a set of easy-to-learn still gestures providing excellent detection accuracy and interactivity. We have to constrain this set to have a reduced number of gestures, such that users can easily remember the gesture-based controls.

7. ACKNOWLEDGEMENTS

This work has been partially supported by the Spanish Ministerio de Ciencia e Innovación, under project TEC2010-18094 and European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement n. 248138 FascinatE.

We would also like to thank Albert Gil and Xavier Suau for providing the demonstrator feedback and the implementation of the head tracker, and Technicolor for the rendering system.

8. REFERENCES

[1] Apple Inc. Magic Trackpad, 2012.

[2] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[3] D. Demirdjian and C. Varri. Recognizing events with temporal random forests. In *Proceedings of the 2009 international conference on Multimodal interfaces*, ICMI-MLMI '09, pages 293–296, New York, NY, USA, 2009. ACM.

[4] European FP7 project FascinatE: Format Agnostic SCript-based InterAcTive Experience. http://www.fascinate-project.com.

[5] J. Gall, A. Yao, N. Razavi, L. Van Gool, and V. Lempitsky. Hough forests for object detection, tracking, and action recognition. *TPAMI*, 33(11):2188 –2202, nov. 2011.

[6] E. Kollorz, J. Penne, J. Hornegger, and A. Barke. Gesture recognition with a Time-Of-Flight camera. *Int. J. Intell. Syst. Technol. Appl.*, 5:334–343, November 2008.

[7] Microsoft Kinect for the Xbox. http://www.xbox.com/kinect.

[8] One-shot learning Gesture Challenge. http://gesture.chalearn.org/.

[9] N. Pugeault and R. Bowden. Spelling It Out: Real-Time ASL Fingerspelling Recognition. In *ICCV-CDC4CV*, 2011.

[10] N. Razavi, J. Gall, and L. Van Gool. Scalable multi-class object detection. In *CVPR*, pages 1505 –1512, june 2011.

[11] Z. Ren, J. Yuan, and Z. Zhang. Robust hand gesture recognition based on finger-earth mover's distance with a commodity depth camera. In *ACM MM*, MM '11, pages 1093–1096, New York, NY, USA, 2011. ACM.

[12] Samsung SMART TV. http://www.samsung.com/us/2012-smart-tv/.

[13] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake. Real-time human pose recognition in parts from single depth images. In *CVPR*, pages 1297 –1304, june 2011.

[14] J. Shotton, M. Johnson, and R. Cipolla. Semantic texton forests for image categorization and segmentation. In *CVPR 2008*, pages 1 –8, june 2008.

[15] X. Suau, J. R. Casas, and J. Ruiz-Hidalgo. Real-time head and hand tracking based on 2.5d data. In *Multimedia and Expo (ICME), 2011 IEEE International Conference on*, pages 1 –6, july 2011.

[16] D. Uebersax, J. Gall, M. Van den Bergh, and L. Van Gool. Real-time Sign Language Letter and Word Recognition from Depth Data. In *ICCV-HCI*, pages 1–8, 2011.