Image-based multiclass boosting and echocardiographic view classification

S. Kevin Zhou[†], J.H. Park[†], B. Georgescu[†], C. Simopoulos[‡], J. Otsuki[‡], and D. Comaniciu[†]
[†]Integrated Data Systems Department, Siemens Corporate Research, Princeton, NJ 08540
[‡]Ultrasound Division, Siemens Medical Solutions, Mountain View, CA 94039

Abstract

We tackle the problem of automatically classifying cardiac view for an echocardiographic sequence as a multiclass object detection. As a solution, we present an image-based multiclass boosting procedure. In contrast with conventional approaches for multiple object detection that train multiple binary classifiers, one per object, we learn only one multiclass classifier using the LogitBoosting algorithm. To utilize the fact that, in the midst of boosting, one class is fully separated from the remaining classes, we propose to learn a tree structure that focuses on the remaining classes to improve learning efficiency. Further, we accommodate the large number of background images using a cascade of boosted multiclass classifiers, which is able to simultaneously detect and classify multiple objects while rejecting the background class quickly. Our experiments on echocardiographic view classification demonstrate promising performances of image-based multiclass boosting.

1. Introduction

Echocardiography (or echo in short) is the ultrasound image of the human heart. It is a commonly used imaging modality to visualize the structure of the heart. Because the echo is typically viewed as a 2D slice of the 3D human heart, standard views are captured to better visualize the cardiac structures. For example, in the apical four-chamber (A4C) view shown in Figure 1(a), all four cavities, namely left and right ventricles, and left and right atria, are present. In the apical two-chamber (A2C) view shown in Figure 1(b), only the left ventricle and the left atrium are present. However, within one standard cardiac view, there are a lot of appearance variations due to dependence on equipment and sonographer, patient difference, etc. To automate the clinical workflow and facilitate the subsequent processing tasks such as endocardial wall motion analysis, it is a must to have an automatic tool that classifies the input echo sequence into standard cardiac views. In this paper, we present a multiple object detection approach to solving the cardiac view classification problem.

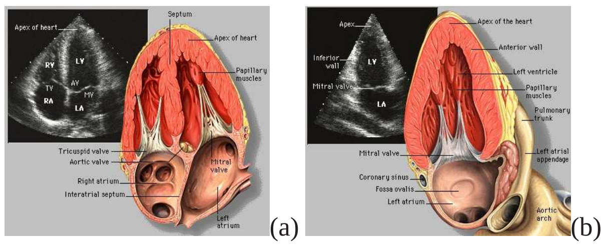


Figure 1. The illustration of (a) apical four-chamber (A4C) and (b) apical two-chamber (A2C) views. Courtesy of Yale Atlas of Echo.

Detection of multiple objects is challenging. Most of the existing approaches train a separate binary classifier for each object against the background and scan the input image for objects. Using different classifiers for different objects has inherent disadvantages. In training, the training complexity increases linearly with the number of classes. In testing, it is very likely that several classifiers fire up at the same location. Due to the difficulty in comparing responses among these classifiers, determining the actual object needs additional work, such as training pairwise classifiers between two object classes. Also, evaluating multiple binary classifiers online is a time-consuming process that thwarts a realtime requirement.

We present an image-based multiclass boosting procedure for object detection. At the core, we train a multiclass classifier using the LogitBoost algorithm [4] that combines weak learners into a strong classifier. The weak learners are associated with the Haar-like local rectangle features [12, 17] for fast computations. Instead of using the simple decision stump (or regression stump), we propose the use of piecewise constant function to enhance the modeling power of the weak learners.

Often in the midst of multiclass boosting, one class is already completely classified correctly. Further boosting does not help too much for that class. To take advantage of this fact, we propose to train a tree structure by focusing on the remaining classes to improve learning efficiency. We show that posterior probabilities can still be computed.

To handle the vexing background class that has numerous examples, we adopt the cascade training procedure [17].

As a result, we achieve a cascade of boosted multiclass strong classifiers, which is a unified algorithm able to simultaneously detect and classify multiple objects while rejecting the background class quickly.

The paper is structured as follows. Section 2 reviews the literature on multiclass classifier and learning-based object detection approaches. In section 3, we recapitulate the multiclass LogitBoost algorithm and discuss the image-based weak learners. In section 4, we present two useful extensions of the multiclass classifier: tree structure for efficient training and cascade structure for handling the background class. Section 5 reports the experiments on echocardiographic view classification. We conclude the paper in section 6.

2. Related literature

In this section, we first briefly review the literature on multiclass classifier and highlight the advantage of using boosting. We then review several object detection approaches that leverage boosting.

2.1. Multiclass classifier

Classifying multiple classes is a long-standing problem in the literature. Early multiclass classifiers [5] include CART, K-nearest neighbors, neural network (e.g., multilayer perceptron), mixture models, etc. Along another line, researchers first convert the multiclass classification problem to several binary classification problems and then combine the binary classifier outputs for a final decision. There are two common combining rules. The first rule is *one vs. the rest*, assuming that one class is separated from the remaining classes. The second rule is *one vs. one*, assuming that pairwise binary classifiers are learned.

The AdaBoost algorithm [3] proposed by Freund and Schapire is an influential algorithm that learns a binary classifier. In [13], Schapire and Singer extended the AdaBoost algorithm to handle a multiclass problem, the so-called AdaBoost.MH algorithm by reducing multiclass to binary. In the work of [4], Friedman *et al.* presented a multiclass boosting algorithm where no conversion from multiclass to binary is necessary.

One distinct approach is the error-correcting output code (EOCC) proposed by Dietterich and Bakiri [1]. In EOCC, because each class is assigned a unique binary string which is computed using binary classifiers, again the multiclass problem is converted to binary ones.

2.2. Learning-based object detection

Since there is a large body of literature on object detection, we only review approaches that are learning-based. Another active line of research on object recognition uses feature points [7, 11] and/or parts [6, 9].

Viola and Jones [17] proposed a real-time face detection algorithm. It invoked the AdaBoost algorithm to selectively combine into a strong committee weak learners based on Haar-like local rectangle features [12], whose rapid computation is enabled by the use of integral image. Another contribution is that they introduced the cascade structure to deal with a rare event detection. In [10], Li and Zhang proposed the FloatBoost, a variant of the AdaBoost, to address multiview face detection. We generalize the Viola and Jones algorithm [17] to deal with multiple objects by training a multiclass classifier with the cascade structure. To the best of our knowledge, it is not clear to us whether such a generalization to detection of multiple objects is feasible.

Because the strategy of learning different binary classifiers for different objects has the scalability issue as each classifier is trained and run independently, sharing features was proposed [15] to overcome this difficulty by enforcing that, during training, the different classifiers maximally share the same set of features. As a consequence, the performance is improved given the same modeling complexity. However, in [15], how to deal with the background class is not demonstrated. The final training outputs are again separate binary classifiers. In our approach, we directly train a multiclass classifier using boosting, which implies that features are always shared at each round of boosting.

Recently, Tu [16] proposed a probabilistic boosting tree (PBT), which unifies classification, recognition, clustering into one treatment. The PBT, which is a binary tree, naturally targets a two-class setting by iteratively learning a strong classifier at each tree node. The training data are subsequently parsed into the left and right node at the next tree level. To deal with a multiclass problem, the author artificially converted a multiclass problem to a two-class problem by first finding the optimal feature that divides all remaining training examples into two parts and then use the two-class PBT to learn the classifier. We also train a tree structure but our method is very different from the PBT. We will detail the differences in section 4.1.

Other than boosting, a general object detection frame based on support vector machine is presented in [12]. In [8], convolutional neural network is trained to deal with multiple object classification.

3. Multiclass boosting

Suppose that we have a $(J+1)$-class classification problem. Classes $\{C_1, C_2, \dots, C_J\}$ stand for J objects and class C_0 stands for the background class or none-of-the-above. The training set for the class C_j $(j = 0, 1, \dots, J)$ is denoted by

$$\mathcal{S}_j = \{x_1^{[j]}, x_2^{[j]}, \dots, x_{n_j}^{[j]}\}.$$

Below when describing the boosting algorithm, the training set for each classes are pooled together to form a big training set $\{x_1, x_2, \ldots, x_N\}$, where $N = \sum_{j=0}^{J} n_j$. The class label for each data point x_i is represented by a $J+1$ vector

$$\mathbf{y}_i^* = [y_{i0}^*, y_{i1}^*, \ldots, y_{iJ}^*]. \tag{1}$$

If the true label for x_i is c, then

$$y_{ij}^* = [j == c] = \{ \begin{array}{ll} 1 & if\ j = c \\ 0 & otherwise \end{array} . \tag{2}$$

where $[\pi]$ is an indicator function of the predicate π. If π holds, then $[\pi] = 1$; otherwise, $[\pi] = 0$.

3.1. LogitBoost algorithm

We adopt the multiclass version of the influential boosting algorithm proposed by Friedman *et al.* [4], the so-called LogitBoost algorithm. The output of the LogitBoost algorithm is a set of $(J+1)$ learned response functions $\{F_j(x);\ j = 0, 1, \ldots, J\}$; each $F_j(x)$ is a linear combination of the set of weak learners, thereby implying that the $F_j(x)$ functions automatically share features [15]. Figure 2 illustrates the LogitBoost algorithm, which fits an additive symmetric logistic model to achieve maximum likelihood using adaptive quasi-Newton steps, whereas the AdaBoost.MH algorithm essentially uses the one against the rest rule by fitting $(J+1)$ uncoupled additive logistic models. See [4] for detailed justification. The final classification result is determined as

$$j = \arg \max_{j=0,1,\ldots,J} F_j(x). \tag{3}$$

One advantage of using the LogitBoost algorithm is that it naturally provides a way to calculate the posterior distribution of class label:

$$p_j(x) = \frac{\exp(F_j(x))}{\sum_{k=0}^{J} \exp(F_k(x))} = \frac{\exp(F_j'(x))}{1 + \sum_{k=1}^{J} \exp(F_k'(x))}, \tag{4}$$

where $F_j'(x) = F_j(x) - F_0(x)$.

The key of the LogitBoost algorithm is the step (Δ) in Figure 2. Mathematically, it solves the following optimization problem:

$$f_{mj} = \arg \min_{f \in \mathcal{F}} \sum_{i=1}^{N} w_{ij} |z_{ij} - f(x_i)|^2, \tag{7}$$

where $\mathcal{F}$ is a dictionary set.

3.2. Dictionary set

The dictionary set is a structural component in boosting since it determines the space where the output classifier function resides. We use the piecewise constant functions as the elements of the dictionary set. A piecewise constant function that is associated with a feature function quantizes the feature response in a piecewise fashion,

LogitBoost ($J+1$ classes)

```
1. Start with weights $w_{ij} = 1/N$, $i = 1, 2, \ldots, N$, $j = 0, 1, \ldots, J$, $F_j(x) = 0$, and $p_j(x) = 1/(J+1)\ \forall j$.
2. Repeat for $m = 1, 2, ..., M$:
    • Repeat for $j = 0, 1, \ldots, J$:
        – Compute working responses and weights in the $j^{th}$ class
              $z_{ij} = \frac{y_{ij}^* - p_j(x_i)}{p_j(x_i)(1 - p_j(x_i))}$;   (5)
              $w_{ij} = p_j(x_i)(1 - p_j(x_i))$.   (6)
        – ($\Delta$) Fit the function $f_{mj}(x)$ by a weighted least-squares regression of $z_{ij}$ to $x_i$ with weights $w_{ij}$.
    • Set $f_{mj}(x) \leftarrow \frac{J}{J+1}(f_{mj}(x) - \frac{1}{J+1}\sum_{k=0}^{J} f_{mk}(x))$, and $F_j(x) \leftarrow F_j(x) + f_{mj}(x)$.
    • Update $p_j(x) \propto \exp(F_j(x))$.
3. Output the classifier $\arg\max_j F_j(x)$.
```

Figure 2. The LogitBoost algorithm [4].

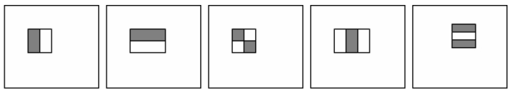


Figure 3. Five types of the Haar-like rectangle filters

$$f(x) = \sum_{k=1}^{K} a_k\ [h(x) \in R_k];\quad R_k = (\theta_{k-1}, \theta_k]. \tag{8}$$

where $[.]$ is an indicator function, $h(x)$ is the feature (response) function, and $\{R_k;\ k = 1, \ldots, K\}$ are the intervals that divide the entire real line with $\theta_0 = -\infty$ and $\theta_K = \infty$.

We use a linear feature $h(x) = g \otimes x$, where g defines the feature and $\otimes$ denotes a convolution. Since every function in the dictionary set is associated with a unique feature, this step allows boosting to operate as an oracle that selects the most relevant visual features. We follow [12, 17] to use the local rectangle features whose responses can be rapidly evaluated through the mean of integral image. Refer to [17] for details on how to rapidly compute the local rectangle features. The g function that defines the feature is parameterized as $g = g_{r,c,w,h,t}$ where (r, c) is the rectangle center, (w, h) is the rectangle size, and t is the feature type. Figure 3 shows five feature types. In sum, the piecewise constant function is in the following form $f(x) = f(x; r, c, w, h, t)$. The intervals R_k for each rectangle feature are empirically determined beforehand. One way is to find the minimum and maximum responses for the rectangle feature and then uniformly divide them into K intervals.

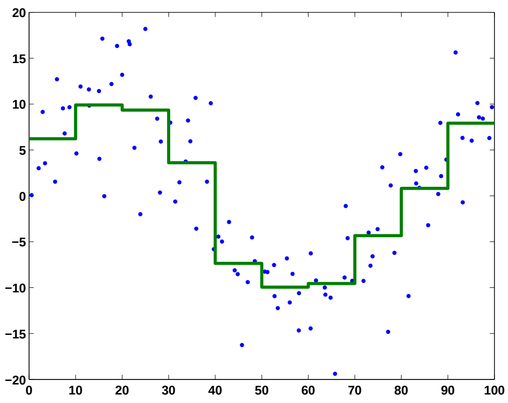


Figure 4. The fitting of a piecewise constant function.

Equation (7) now becomes

$$f_{mj} = \arg \min_{r,c,w,h,t} \sum_{i=1}^{N} w_{ij} |z_{ij} - f(x_i; r, c, w, h, t)|^2. \quad (9)$$

For a given configuration (r, c, w, h, t), the optimal values of a are computed as

$$a_k = \frac{\sum_{i=1}^{N} w_{ij} z_{ij} [(g \otimes x_i) \in R_k]}{\sum_{i=1}^{N} w_{ij} [(g \otimes x_i) \in R_k]}. \quad (10)$$

Figure 4 illustrates fitting the piecewise constant function to a set of unweighted data points. In the step (Δ) of Figure 2, the most relevant feature with the smallest error is singled out to maximally reduce the classification error.

In the literature, simple binary regression stumps are used [17, 15]. However, it is easy to show that the piecewise constant function combines multiple binary regression stumps, therefore enhancing the modeling capability of the weak learners and consequently improving training speed.

4. Extensions

In this section, we address two useful extensions of the multiclass boosting algorithm. In the first extension, we show how to learn a tree structure which improves efficiency in both training and testing. In the second extension, we propose to train a cascade structure to deal with the vexing background class that has numerous examples.

4.1. Tree structure

Empirical evidence tells that often, in the midst of boosting a multiclass classifier, one class (or several classes) has been completely separated from the remaining ones and further boosting yields no additional improvement in terms of the classification accuracy. This fact can be utilized for efficient training. To this end, we propose to train a tree structure.

Figure 5 gives a simple example to illustrate the tree training. Suppose we are given a 3-class problem (C_1, C_2, and C_3). After several boosting iterations, we find that the class C_1 has been classified correctly. We stop training and store the output functions as $\{F_{1,j}(x);\ j = 1, 2, 3\}$, which forms the first layer of the tree that merges the classes C_2 and C_3. Next, for the second layer of the tree, we continue to train a binary classifier that separates C_2 and C_3 and store the output functions as $\{F_{2,j}(x);\ j = 2, 3\}$.

To calculate the posterior probability of the class label, we first computer the posterior probability for each tree layer. For example, for the first layer, we compute

$$p_1(1|x) = \frac{\exp(F_{1,1}(x))}{\sum_{j=1}^{3} \exp(F_{1,j}(x))};\ p_1(2,3|x) = 1 - p_1(1|x). \quad (11)$$

For the second layer, we compute

$$p_2(2|x) = \frac{\exp(F_{2,2}(x))}{\sum_{j=2}^{3} \exp(F_{2,j}(x))};\ p_2(3|x) = 1 - p_2(2|x). \quad (12)$$

Finally, the overall posterior probabilities are

$$p(1|x) = p_1(1|x);\ p(2|x) = p_1(2,3|x) p_2(2|x); \quad (13)$$

$$p(3|x) = p_1(2,3|x) p_2(3|x). \quad (14)$$

It is easy to verify that $\sum_{j=1}^{3} (j|x) = 1$.

Generalizing the above 3-class example to an arbitrary multiclass problem is easy. The main idea is to repeat a ‘divide-and-merge’ strategy. For a merged class, we learn a multiclass classifier to divide them. After the classifier is learned, we separate those ‘easy’ classes from the remaining classes that are merged into one class. Similarly, the posterior calculation can be carried out.

The proposed tree structure is very different from the multiclass PBT algorithm [16]. First, when ‘dividing’ at each tree node, the PBT first converts a multiclass task to a binary one based on an optimal feature and then invokes boosting to learn a binary classifier; whereas we never perform such a conversion and always learn a truly multiclass classifier. Second, there is no ‘merging’ operation in the PBT. The samples are always divided till there is no further ambiguity, leading to a risk of overfitting. Since we operate at a class level (rather than a sample level as in the PBT), we seldom overfit.

4.2. Cascade structure

The cascade structure that corresponds to a degenerate decision tree is introduced to perform learning under a rare event scenario. Such a scenario presents an unbalanced nature of data samples. The background class has numerous samples because all data points not belonging to the object classes belong to it.

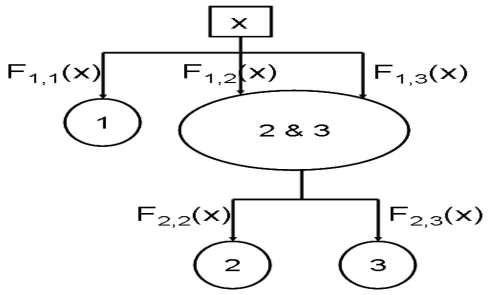


Figure 5. The tree structure of a boosted multiclass classifier.

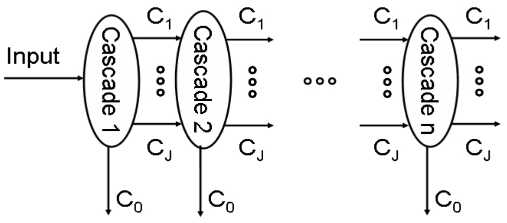


Figure 6. The cascade of boosted multiclass classifiers.

To effectively examine more background examples, only those background examples that pass the early stages of the cascade are used for training the current stage. This idea is the same as negative bootstrapping in the neural network literature. However, the learned strong classifier for each stage has to be adjusted in order to reach a desired detection rate. In [17], this is done by changing the strong classifier threshold.

We follow [17] to train a cascade of boosted multiclass classifiers as shown in Figure 6. The examples for the background class fed to the first stage are randomly cropped. To train a cascade of boosted multiclass classifiers, we must make sure that the training examples for class $\{C_1, C_2, \ldots, C_J\}$ can succeed the available stages. To achieve this, we decide to change the response function $F_j(x)$. One simple solution is to add a constant shift T_0 to the output function of the background class, $F_0(x)$. The value of T_0 is chosen to guarantee the desired detection rate. Note that here the concept of the detection rate is applied to all classes except the background class.

Suppose that the desired rate is $1-\alpha$, we choose T_0 such that

$$F_i(x) \geq F_0(x) + T_0;\ x \in \mathcal{S}_i = \{x_1^{[i]}, x_2^{[i]}, \ldots, x_{n_i}^{[i]}\} \quad (15)$$

must be valid for $\lfloor(\sum_{j=1}^{J} n_j) * (1-\alpha) + 0.5\rfloor$ times. In other words, T_0 is the α percentile of the values $\{F'_j(x) = F_j(x) - F_0(x); x \in \mathcal{S}_j,\ j = 1, 2, \ldots, J\}$.

It should be emphasized that the tree and cascade structures are two different concepts. The tree structure is a multiclass classifier, while the cascades consists of several stages of multiclass classifiers. In practice, the two structures can be combined to achieve better efficiency.

The use of cascade during online detection of object is illustrated as follows. Given an image, we want to find the instances of the objects of interest in it. This is achieved via an exhaustive scanning of the image. A window W that defines a region of interest slides in the space of (x, y, θ, s), where (x, y), θ, and s are the center, orientation, and scale of the window, respectively. Only the windows that pass all stages of the cascade are subject to final classification. Denote the set of all these windows by $\mathcal{W}$:

$$\mathcal{W} = \{W | p_0^{(n)}(W) < \max_{j=1,\ldots,J} p_j^{(n)}(W); n = 1, ..., N_{cas}\}. \quad (16)$$

where $p_j^{(n)}$ is the posterior probability of W being class j when the boosted multiclass classifier for the n^{th} cascade is used and N_{cas} is the number of stages in the cascade.

In the case where multiple objects are present in the scene, we can simply classify each window in the above set, assign it a class label and post-process those windows with severe overlapping. In the case when only one final decision is needed, such as cardiac view classification, *ad hoc* fusion strategies such as majority voting can be used.

5. Echocardiographic view classification

In this section, we present details of our automatic system of echocardiographic view classification (EVC), where an echocardiographic video in a DICOM format is classified into three classes, namely the A4C and A2C view along with the background class, using the learned cascade of multiclass strong classifiers. The reason why we chose A2C and A4C views for our experiments is that these two views are commonly used clinically, and they are very similar to each other and hard to discriminate. Therefore, it is enough to show a proof of concept. In the future, we plan to add more views and use the same procedure for classification.

The EVC system takes an echo video as input but infers the view label based on an image classification approach without considering temporal information to reduce computation time toward a real-time requirement. Only the end-diastolic (ED) frame is used for classification. The ED frame is characterized by the fact that the heart is mostly dilated. Also, the ED frame index is directly coded in the DICOM header.

The overview of the EVC system is depicted in Figure 7. It consists of three modules: 1) training data collection, 2) multiclass boosting training, and 3) online view classification. During online classification, the system only analyzes the ED frame given an echocardiographic video sequence, and classifies the ED frame into one of the trained views by combining the results of all scanned subwindows. The classification result of the ED frame represents the final view of the echocardiographic video sequence.

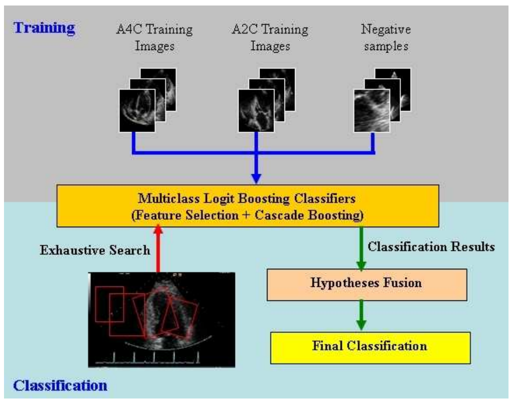


Figure 7. The overview of the EVC system

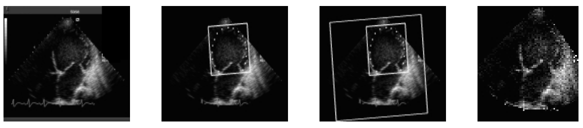


Figure 8. Procedure of template image cropping. (a) A4C ED frame. (b) LV contour and bounding box. (c) Template bounding box along with LV bounding box. (d) Cropped template image.

5.1. Training data collection

We used 391 A2C sequences and 466 A4C sequences for collecting training data. We first designed template layouts for the two views, which highlight their characteristic structures. The template is designed based on the left ventricle (LV) orientation and size. Specifically, for the A4C view, the template contains all four chambers; while for the A2C view, the template contains both chambers. We manually annotated the LV endocardium to align the data and reduce the appearance variation. Given a training video sequence and its LV annotation, a template image is cropped according to the template layout and rescaled to a canonical size. The template cropping procedure is shown in Figure 8.

Given a video sequence, training images are collected from five frames, i.e., the $ED - 2$ frame to $ED + 2$ frame Further, we collected additional positives by perturbing the template bounding box by a small amount around its center and in its width, height, and angle. Negative training images are collected outside the perturbing range for positives.

5.2. Multiclass boosting training

We employed Haar-like local rectangle features used in [12, 17]. The Haar-like local rectangle filters provide local high frequency components in various directions and resolutions. Figure 3 shows five types of the filters we used, where the output of the filters is a scalar value by subtracting the sum of the intensity values in the white rectangle(s) from that of the grey rectangle(s). The output value indicates the magnitude of a high frequency of the local region. We can generate a very large number of filters by varying the type, location, and size of the filter.

We trained a cascade of two multiclass classifiers, following the principles presented in section 3 and 4. For each stage of the cascade, we further trained a tree structure (as in 4.1) if applicable. To train the first cascade, we randomly selected twice as many negative samples as the positives. To train the second cascade, we followed the bootstrapping procedure in section 4.2.

	A2C	A4C	Missed
A2C	31/34=91.2%	3/34=8.8%	0/34=0%
A4C	3/48=6.2%	43/48=89.6%	2/48=4.2%

Table 1. Detection and Classification results for the test data

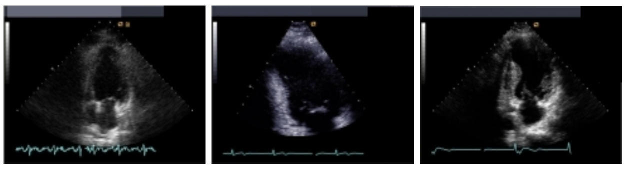
5.3. View classification and results

We used 34 A2C sequences and 48 A4C sequences for experiments of detection and classification of cardiac views. Given a sequence, we performed exhaustive search from the left-top corner to the right-bottom corner by changing the width, height, and angle. This exhaustive search approach may yield multiple results of detection and classification, especially around the correct view location. We employed a multiple hypotheses fusion approach based on majority voting. If the EVC system yields multiple hypotheses (classification results), the number of classification results of each view is computed and the final view is determined according to the maximum number of view classification results. If the numbers are the same, we employed a simple tie-break rule based on the mean of all the posterior probabilities the classifier provides.

The detection and classification results are shown in Table 1. There are 31 out of 34 A2C sequences correctly classified and 3 sequences misclassified as A4C. In the case of A4C, there are 43 out of 48 sequences detected and classified correctly, 3 sequences misclassified as A2C, and 2 sequences failed in classification as any of the two views. Figure 9 illustrates some of the misclassified examples. As shown in the figure, these misclassified ones are very challenging even for human discrimination if only the ED frame is presented. In terms of speed, on average, our system runs almost in real time at a PC with a 2GHz CPU and a 2GB memory, consuming about 1.5s to classify a sequence which typically contains a full cardiac cycle with about 30 frames.

5.4. Discussion

To the best of our knowledge, the only relevant work that attempted to recognize the cardiac views is [2] where the constellation model of the cavities is used. The authors of [2] used the gray-level symmetric axis transform [14] to detect a cardiac chamber, assuming that there exists a distinct


(a) Three A2C examples misclassified to A4C

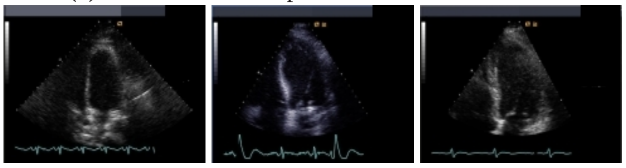

(b) Three A4C examples misclassified to A2C

Figure 9. Misclassified examples

cavity in the image for each chamber of the heart. While this assumption in general holds for the A4C view, it is invalid for the A2C view. After the chambers are detected, they modeled the constellation of the parts using the Markov random field to capture both appearances and spatial relationships of the chambers. The support vector machine classifier (SVM) is used for view classification, based on the energy vector extracted from the Markov model.

Our approach is very different from [2]. While a three-stage approach is presented in [2], we proposed a one-shot solution built on the machine learning and appearance-based object detection literature. One main drawback of [2] is that its performance heavily depends on the chamber detector, which has no guarantee of localizing the parts correctly. Our approach, instead, explicitly encodes the part structure into template design and searches the desired part structure in running time. In fact, one byproduct of our approach is accurate localization of the chambers.

In experiments, a small database was used in [2], with 15 echo videos of normal cases and six videos of abnormal cases. Four experimental settings were evaluated and accuracies ranging from 54% to 88% were reported using the leave-one-out scheme. It is hard to predict how their performance scales to a large database. We used a large database: training uses 391 A2C and 466 A4C videos and testing uses 34 A2C and 48 A4C sequences. We achieved an overall accuracy of 90.2%. Also there is no computational speed reported in [2]; while we run almost in real time. Presumably, computation in [2] cannot be close to real time because fitting a Markov random field model is slow.

6. Conclusions

We proposed a multiclass boosting procedure for simultaneous detection of multiple objects and successfully applied it to automatic view classification of echocardiographic sequences. The image-based multiclass boosting is a one-shot solution that possesses many promising properties such as the tree structure for efficient training, the cascade training to deal with rare event detection, and always sharing local feature across the classifier outputs. In future, we plan to (i) increase the number of views in the EVC system and (ii) evaluate the proposed multiclass boosting method to detect generic objects.

References

[1] T. Dietterich and G. Bakiri. Solving multiclass learning problem via error-correcting output codes. *Journal of Artificial Intelligence*, 2:263–286, 1995.

[2] S. Ebadollahi, S. Chang, and H. Wu. Automatic view recognition inechocardiogram videos using parts-based representation. In *CVPR*, pages 2–9, 2004.

[3] Y. Freund and R. Schapire. A decision-theoretic generalization of online leaning and an application to boosting. *Journal of Computer and System Sciences*, 5(1):119.

[4] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. *Ann. Statist.*, 28(2):337–407, 2000.

[5] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer, 2001.

[6] S. Krempp, D. Geman, and Y. Amit. Sequential learning of reusable parts for object detection. *Technical report, CS Johns Hopkins Univ.*, 2002.

[7] S. Lazebnik, C. Schmid, and J. Ponce. Affine-invariant local descriptors and neighborhood statistics for texture recognition. In *ICCV*, 2003.

[8] Y. LeCun, F. Huang, and L. Bottou. Learning methods for generic object recognition with invariance to pose and lighting. In *CVPR*, 2004.

[9] F. Li, R. Fergus, and P. Perona. A bayesian approach to unsupervised one-shot learning of object categories. In *ICCV*, 2003.

[10] S. Li and Z. Zhang. Floatboosting learning and statistical face detection. *PAMI*, 26(9), 2004.

[11] D. Lowe. Object recognition from local scale-invariant features. In *ICCV*, 1999.

[12] C. Papageorgiou, M. Oren, and T. Poggio. A general framework for object detection. In *ICCV*, 1998.

[13] R. Schapire and Y. Singer. Improved boosting using confidence-rated predictions. *Machine Learning*, 37(3):297–336, 1999.

[14] H. Tagare, F. Vos, C. Jaffe, and J. Duncan. Arrangement : A spatial relation between parts fo evaluating similarity of tomograhic section. *PAMI*, 17:880–893, 1995.

[15] A. Torralba, K. Murphy, and W. Freeman. Sharing features: efficient boosting procedures for multiclass object detection. In *CVPR*, 2004.

[16] Z. Tu. Probabilistic boosting-tree: Learning discriminative models for classification, recognition, and clustering. In *ICCV*, 2005.

[17] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *CVPR*, pages 511–518, 2001.