Joint Real-time Object Detection and Pose Estimation Using Probabilistic Boosting Network

Jingdan Zhang[‡], Shaohua Kevin Zhou[†], Leonard McMillan[‡], and Dorin Comaniciu[†]
[†]Integrated Data Systems Department, Siemens Corporate Research, Princeton, NJ 08540
[‡]Department of Computer Science, UNC Chapel Hill, Chapel Hill, NC 27599

Abstract

In this paper, we present a learning procedure called probabilistic boosting network (PBN) for joint real-time object detection and pose estimation. Grounded on the law of total probability, PBN integrates evidence from two building blocks, namely a multiclass boosting classifier for pose estimation and a boosted detection cascade for object detection. By inferring the pose parameter, we avoid the exhaustive scanning for the pose, which hampers real time requirement. In addition, we only need one integral image/volume with no need of image/volume rotation. We implement PBN using a graph-structured network that alternates the two tasks of foreground/background discrimination and pose estimation for rejecting negatives as quickly as possible. Compared with previous approaches, we gain accuracy in object localization and pose estimation while noticeably reducing the computation. We invoke PBN to detect the left ventricle from a 3D ultrasound volume, processing about 10 volumes per second, and the left atrium from 2D images in real time.

1. Introduction

Real-time object detection and pose estimation are important to many computer vision applications ranging from face detection to segmentation of anatomical structures in medical images. The state-of-the-art of object detector is from Viola and Jones [12]: They off-line learn a binary classifier that differentiates the object of interest from the background and then online exhaustively slide a scanning window on the input image for object instances. The key ideas in [12] include the use of boosting for feature selection, the use of the integral image for fast evaluation of local rectangle features via the means of integral image, and the use of cascade for quick negative elimination.

However, it is challenging to build a real-time detector that also performs accurate pose estimation by using the detector like [12] to exhaustively scan all possible translations and poses. Consider detecting the left ventricle (LV) in a 3D echocardiogram (echo), which is an ultrasound volume of the human heart (Figure 1). Discovering the LV configuration is helpful to navigate the 3D volume; for example, from a known LV configuration, one can meet an emerging clinical need to automatically display canonical 2D slices. Since the LV can occur at arbitrary location and orientation, one needs six parameters[1] to fully align the LV rigidly: (t_x, t_y, t_z) for translation and $(\theta_x, \theta_y, \theta_z)$ for rotation. When extending [12] for LV detection in 3D volume, the computational cost increases exponentially with the dimensionality of the configuration space. Further, volume rotation and integral volume computation are time consuming since their computation is linearly dependent on the number of voxels. For example, it takes more than 1s to rotate a volume of size $128 \times 128 \times 128$ on a PC with a 2.4GHz CPU and 2GB memory!

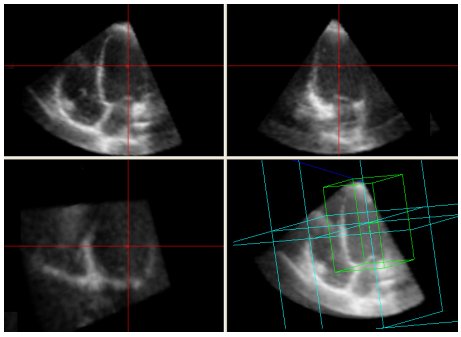


Figure 1. *Detecting the LV and its pose in a 3D ultrasound volume is important to automatically navigate multiple canonical planes for clinical practices.*

In this paper, we present an algorithm called probabilistic boosting network (PBN), which is able to jointly detect the object quickly and estimate the pose accurately. PBN estimates the pose β via a multiclass approximation to the posterior probability $p(\beta|I)$, learns the object detector $p(O|I, \beta)$ for each pose, and integrates the evidence from the pose classifier and object detectors using the total probability law as an integrating rule:

[1]Ideally, one also needs three additional parameter (s_x, s_y, s_z) for compensating the different scale/size varying with patients.

1-4244-1180-7/07/$25.00 ©2007 IEEE

$$p(O|I) = \int_{\beta} p(O|I,\beta)p(\beta|I)\, d\beta. \quad (1)$$

The above integrating rule is an instance of combining the generative mixture model with discriminative classification models [11]. To train the multiclass classifier and binary object detectors, PBN invokes boosting algorithm to perform feature selection based on a common feature pool. In addition, PBN endows a network structure to support efficient evaluation. Finally, PBN computes only one integral volume/image; no volume/image rotation is needed.

In summary, we make the following contributions: (i) We propose in Section 3 a multiclass approximation for pose estimation and derive an accurate estimate for even two continuous pose parameters. We learn the multiclass classifier via the LogitBoost algorithm that embeds feature selection and allows fast evaluation; (ii) We ground the process of evidence integration on the total probability law, which lays a solid theoretic foundation. As shown in Section 4, we analytically compute the posterior probability of being an object with no ad-hoc treatment; (iii) We implement in Section 4 two network structures (tree and graph) to estimate pose and discriminate foreground from background efficiently. The implemented structure enables fast rejection of patches both on the background and of wrong poses with the aid of only one integral volume/image; and (iv) We present in Section 5 a real-time algorithm that performs object detection and pose estimation from a 3D volume. To the best of our knowledge, this is the first kind in the literature. We also achieve real time performances on object detection from 2D images.

We proceed to address the real time challenges in achieving joint fast object detection and pose estimation and then motivate our approach.

2. Joint object detection and pose estimation

2.1. Previous approaches

Viola and Jones [12] proposed the first real time frontal-view face detector by exhaustively scanning the location (t_x, t_y) and scale s. However, only a sparse set of scales is scanned and the same scale is used for both x and y directions. There is no rotation variation involved.

While the fast evaluation of local rectangle features is applicable when dealing with scale variation, it is not as straightforward when applying it to the rotation variation. A simple approach is to train one classifier by pooling together data from all orientations as positives: The variation introduced by rotation is treated as intra-class. However, as previously pointed out [5, 7, 8], this treatment causes a significant increase in appearance variation, rendering a complex detector and inaccurate detection. Also it is impossible to recover rotation information. Another possibility [4] is to rotate the image for each sampled orientation and compute its according integral image; but, as mentioned earlier, it is computationally prohibitive for 3D volume.

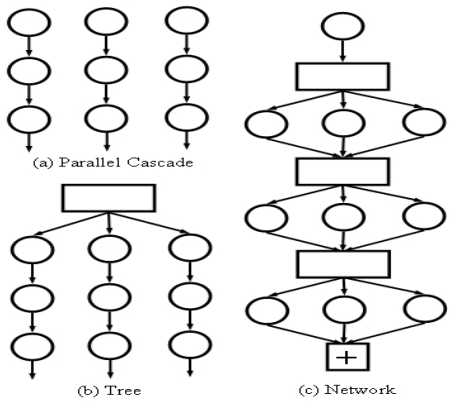


Figure 2. *Different structures for multi-pose detection. The circle represents a binary foreground/background classifier. The rectangle represents a multiclass pose classifier.*

A promising solution that requires only one integral volume/image is to train a collection of binary classifiers to handle different poses. A variety of structures are proposed to combine these classifiers. The most straightforward way is a parallel structure (Figure 2(a)) that trains a classifier for each discretized pose [13]. In detection, all classifiers are tested for every scanning window. The computation linearly depends on the number of poses. To accelerate the detection speed, the pyramid structure is proposed by Li *et al*. [8]. This approach adopts a coarse-to-fine strategy to train classifiers with increasingly narrower pose ranges. For the parallel structure, several classifiers might fire up at the same place when the actual pose is in-between the discretized poses. To estimate accurate pose needs additional work due to the difficulty in comparing responses among these classifiers. To discriminate different poses explicitly, tree structure (Figure 2(b)) that uses multiclass classifier as a differentiator is applied. In [7], a decision-tree is used to determine the pose of a face, followed by the binary classifier trained for that pose only. Most recently, Huang *et al*. [5] proposed a tree structure to divide the pose space into increasingly smaller subspaces. The foreground/background discrimination and pose estimation are performed at the same node using the VectorBoost algorithm.

In the above-mentioned detection approaches [5, 7, 8, 13], only a sparse set of orientations and scales are tested to meet real time requirement. Unfortunately, the anatomical structure in medical image (such as the LV) can possess arbitrary orientation and scale and the detection task needs to accurately determine them. Under these circumstances, it is challenging to build a rapid detector using the above approaches. In general, the speed is inversely proportional to the number of tested poses; in order to give an accurate account of the pose, the speed has to be sacrificed.

2.2. Our approach

We explore along the promising line of the tree structure. To avoid searching the whole configuration space, we break the whole configuration space into two halves: For the first half (such as translation), we still use exhaustive scanning; for the second half (such as rotation and even scale), we directly estimate the parameters from the image appearance. We refer to the parameters in the second half as *pose*. We couple the exhaustive scanning with pose estimation; this way we successfully remove the linear computational dependency on the number of poses.

For the tree structure, usually only one branch is evaluated based on the decision of the multiclass classifier [7] and hence, the error made by the multiclass classifier has great influence on the detection result. In [5], several branches may be chosen, based on the hard decision provided by the VectorBoost algorithm, for the purpose of reducing the risk of possible errors in pose estimation. We handle the uncertainty of pose estimation using probabilities as soft decisions. The multiclass classifier is trained using the LogitBoost algorithm [3], which provides a sound theory of computing the probability of the pose. This probability is used to choose branches to evaluate. The final probability of being the object is computed by applying the total probability law. In addition, we obtain a better estimation of the pose by using the conditional mean of the pose distribution.

To further increase the efficiency of the tree structure, a graph-structured network (Figure 2(c)) is proposed to reject background as early as possible. The multiclass classifier is decomposed into several sub-classifiers by taking advantage of the additive native of the boosted classifier. The sub-classifiers and binary detectors are coupled to form a graph structure that alternates the two task of pose estimation and object detection. Furthermore, we add a binary classifier as a pre-filter of the graph to reject the background that can be easily differentiated from the foreground.

3. Pose Estimation

Given a window containing the object of interest, we want to estimate the pose parameter(s) of the object based on the image appearance I in the window. We first discuss the algorithm for one-parameter estimation and then use the one-parameter estimation algorithm as a building block to deal with two-parameter estimation.

3.1. One-Parameter Estimation

Because the object appearance variation is caused not only by the rigid transformation we want to estimate, but also by noise, intensity variations, and nonrigid shape deformation, etc., a robust algorithm is needed to guarantee the accuracy of pose estimation. We handle the uncertainty of the pose estimation by learning a probability $p(\beta|I)$. This probability can be used to estimate β accurately and to avoid the early hard decisions that can cause errors in subsequent tasks.

In practice, $p(\beta|I)$ is approximated by the discretized probability $p(\beta_j|I)$ using a discrete set of parameters $\{\beta_1, \beta_2, \ldots, \beta_J\}$. We implemented the image-based multiclass boosting algorithm proposed by Zhou *et al*. [14]. This algorithm is grounded on the multiclass version of the influential boosting algorithm proposed by Friedman *et al*. [3], the so-called LogitBoost algorithm, which fits an additive symmetric logistic model by the maximum likelihood principle. This fitting proceeds iteratively to select weak learners and combines them into a strong classifier. The output of the LogitBoost algorithm is a set of J response functions $\{F_j(x);\ j = 1, \ldots, J\}$; each $F_j(x)$ is a linear combination of a set of weak learners:

$$F_j^n(x) = \sum_{i=1}^{n} f_{j,i}(x), \tag{2}$$

where $f_{j,i}(x)$ is a weak learner and n is the number of weak learners. LogitBoost provides a natural way to calculate the posterior distribution of class label:

$$p_j^n(x) = \frac{\exp(F_j(x))}{\sum_{k=1}^{J} \exp(F_k(x))}. \tag{3}$$

Refer to [14] for implementational details.

One merit of the LogitBoost algorithm is that the computed posterior probability approximates the true one asymptotically [3]. This means that we can tradeoff the approximation accuracy with the computational cost by adjusting the number of weaker learners in the response functions. This property is used to build the network structure to reject background more efficiently at early stages.

We infer the parameter β by using the minimum mean square error (MMSE) estimator, which is the conditional mean.

$$\hat{\beta}_{MMSE} = \int_{\beta} \beta p(\beta|I)\, d\beta \approx \sum_{j} \beta_j p(\beta_j|I). \tag{4}$$

This gives a better estimate than the maximum a posterior (MAP) estimate from a discrete set of possible values of β because the MMSE estimate can interpolate in between the values in the discrete set. The MAP estimator is defined as

$$\hat{\beta}_{MAP} = \max_{\beta} p(\beta|I) \approx \max_{\{\beta_1,\ldots,\beta_J\}} p(\beta_j|I). \tag{5}$$

Figure 3 compares the MMSE and MAP estimates for the rotation parameter in the second experiment (section 5.2). We learned the multiclass classifier using the 299 aligned training images rotated from -30 to 30 degree with 10 degrees apart. During testing, we synthetically rotated 72 aligned testing data with one degree apart. Figure 3(a) displays the rotation estimation result of a testing data. As the MMSE estimate can interpolate between angles, it gives

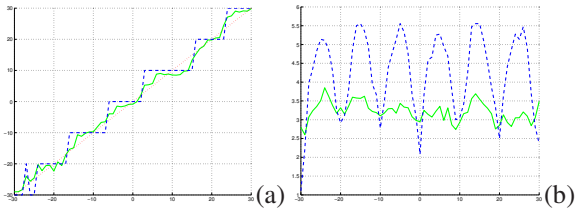


Figure 3. *The estimation of the rotation parameter. (a) a single testing data. (b) the MAE of all testing data. The red line is the ground truth. The blue line is the MAP estimate. The green line is the MMSE estimate.*

a smoother curve. Figure 3(b) shows the mean absolute error (MAE) in angle estimation of all testing data. The MMSE estimate obviously outperforms the MAP estimate except when the angle is close to the discretized values.

3.2. Two-Parameter Estimation

The LogitBoost algorithm that estimates one parameter can be used as a building block to construct a graph structure for estimating two or more parameters. In this section, we will focus on two-parameter estimation; the same principle can be applied to situations with more than two parameters. Suppose that the two unknown parameters are γ and β, we want to estimate $p(\beta, \gamma|I)$ from the image I. Figure 4 given a graphical illustration of three possible structures that can be used to fulfill the estimation task.

For the type-a structure, we treat the combination of β and γ as a single variable and train $p(\beta, \gamma|I)$ directly. This approach is structurally simple and has good performance when the number of the combination states is small. However, when the number of combination states is large or the appearance variations caused by both parameters are too complex to learn using a single classifier, this approach will give a poor performance. In this case, the divide-and-conquer strategy is appropriate to estimate parameters sequentially by using multiple multiclass classifiers.

For the type-b structure, we assume β and γ are independent. To train $p(\beta|I)$ (or $p(\gamma|I)$), we treat the variation in γ (or β) as intra-class. The joint distribution is approximated as

$$p(\beta, \gamma|I) \approx p(\beta|I) * p(\gamma|I). \quad (6)$$

The drawback of this approach is that the assumption of independency of β and γ is usually not valid.

For the type-c structure, we apply the exact conditional probability law:

$$p(\beta, \gamma|I) = p(\gamma|\beta, I) * p(\beta|I) \quad (7)$$

This can be represented as a tree structure. A root multiclass classifier is trained to learn $p(\beta|I)$ by treating the variation in γ as intra-class. Each child node corresponds to the conditional probability $p(\gamma|\beta_j, I)$ for a discrete state β_j. To compute $p(\beta, \gamma|I)$ efficiently, we omit branches whose probability $p(\beta|I)$ is below a predetermined threshold.

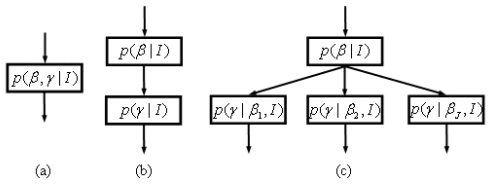


Figure 4. *Different structures for estimating probability of two-parameter pose.*

The choice of the root parameter for the type-c structure influences the overall performance, because the amount of the image appearance variations caused by the two parameters is not the same. Usually, the parameter that causes larger appearance variation should be the root node. This enables learning $p(\beta|I)$ easier and makes a better division of the pose space.

How to choose these three types is determined by the data properties. Our principle is that if the number of the poses is small and the appearance variation can be sufficiently captured by one classifier, we use the type-a structure. Otherwise, we choose the type-c structure.

4. Probabilistic boosting network (PBN)

In previous section, we discussed how to estimate the pose parameter in the scanning window. We now present the probabilistic boosting network (PBN) that integrates evidence from pose estimator and binary detectors. PBN characterizes three basic features.

1. It is *probabilistic*. PBN leverages the fundamental total probability law to compute the probability of being object O. Assuming that the pose parameter β is discretized into $\{\beta_1, \beta_2, \ldots, \beta_J\}$,

$$p(O|I) = \sum_{j=1}^{J} p(O|I, \beta_j) p(\beta_j|I), \quad (8)$$

where $p(O|I, \beta_j)$ is the binary classifier specific to a particular parameter β_j. To compute (8) efficiently, we ignore branches whose pose probability is smaller than a pre-specified threshold p_0.

$$p(O|I) \approx \sum_{j:\ p(\beta_j|I) \geq p_0} p(O|I, \beta_j) p(\beta_j|I). \quad (9)$$

2. It uses *boosting*. As discussed in section 3, the probability $p(\beta_j|I)$ is implemented using the multiclass LogintBoost algorithm [3]. The classifier $p(O|I, \beta_j)$ is implemented using the cascade of boosted binary classifiers [12], which is able to deal with numerous negatives and eliminate them as early as possible during testing. To implement the binary classifier in the cascade, one can use AdaBoost [2], binary LogitBoost [3], or other variants. Suppose that the cascade has C_j stages, then $p(O|I, \beta_j)$ is computed as

$$p(O|I,\beta_j) = \prod_{c=1}^{C_j} p_c(O|I,\beta_j), \qquad (10)$$

where $p_c(O|I,\beta_j)$ is the binary classifier for the c^{th} cascade. The complexity of the classifier $p_c(O|I,\beta_j)$ increases as the stage proceeds deeper (*i.e.*, c becomes larger). Without loss of generality, we assume that $C_1 = C_2 = ... = C_J = C$. If say $C_j < C$, we simply set $p_c(O|I,\beta_j) = 1$ for $c > C_j$.

3. It endows a *network* structure in implementation. Since the total probability law turns into

$$p(O|I) = \sum_{j=1}^{J} \prod_{c=1}^{C} p_c(O|I,\beta_j) p(\beta_j|I), \qquad (11)$$

it is implemented as a tree-structured network shown in Figure 2(b). Using this structure, the negatives will be rejected quickly when they flow through the network while the positives traverse through several branches. By combining the evidence from these branches, we are able to compute $p(O|I)$ exactly using (11).

In [9], a learning procedure called probabilistic boosting tree (PBT) is presented. Both PBN and PBT are able to provide exact object detection probability, use boosting, and have a tree structure, but they also differ a lot. In the tree-structured PBN, each node corresponds to a specified nuisance parameter, while in PBT there is no specific nuisance parameter. PBN also estimates the pose parameter explicitly. Finally, PBN manifests an efficient graph structure as shown next, which is not available in PBT. In [1, 6], Boosting is combined with (dynamic) Bayesian network for parameter learning, structure selection, and audio-video speaker detection.

4.1. Efficient graph structure

The tree-structured PBN is not yet optimal in terms of computation because the overhead computation of the probability $p(\beta|I)$ has to been done for all background windows. These candidate windows are randomly sent to several branches of cascades and rejected by these branches. This creates a dilemma for the tree-structure: the purpose of multiclass classifier is to select proper binary classifiers to reject background; however, determining the pose of these background patches wastes computation.

One way to solve this problem is to discard as much background windows as possible via a focus-of-attention mechanism. This can be achieved by pooling together all data from different poses as positives to train a pose-invariant classifier as a pre-filter. We tune the pose-invariant detector to have a 100% detection rate by adjusting the threshold. This detector cannot offer a precise detection, but it is useful to reject a large percentage of the background candidates.

To compute the probability $p(O|I)$

1. Let e_j be the number of the binary classifiers already evaluated in the j^{th} branch. Start with $e_j = 0$, response functions $F_j(x) = 0$, and probabilities $p(O|I,\beta_j) = 1$, $j = 1, \ldots, J$.
2. Use the pose-invariant classifier to pre-filter. If I is background, set $p(O|I) = 0$ and exit.
3. Repeat for $c = 1, 2, ..., C$:
   - Add n_c weak learners to $F_j(x)$ and update $p(\beta_j|I)$.
   - For j^{th} branch, $j = 1, \ldots, J$, if $p(\beta_j|I) \geq p_0$ and $p(O|I,\beta_j) > 0$:
     - Compute the probabilities $p_k(O|I,\beta_j)$, $k = e_j + 1, \ldots, c$. If it is background, set $p(O|I,\beta_j) = 0$.
     - Update
     $$p(O|I,\beta_j) \leftarrow p(O|I,\beta_j) \prod_{k=e_j+1}^{c} p_k(O|I,\beta_j).$$
     - Update $e_j = c$.
   - If $p(O|I,\beta_j) = 0$ for all branches, set $p(O|I) = 0$ and exit.
4. Compute $p(O|I)$ based on the total probability law.

Figure 5. *The PBN detection algorithm.*

Even though the pre-filter classifier is used, there are still quite some non-object windows passing the pre-filter and causing overhead computation of $p(\beta|I)$. Just as the cascade structure for binary detector $p(O|I,\beta_j)$, which breaks its computation into several stages with increasing complexity, we also decompose the computation of $p(\beta|I)$ into several stages by taking the advantage of the additive model arising from the LogitBoost algorithm. The response functions at the c^{th} stage is:

$$F_j^c(x) = F_j^{c-1}(x) + \sum_{i=1}^{c_n} f_{j,i}(x), \qquad (12)$$

where c_n is the number of weak learners at the c^{th} stage. For the type-b and type-c structures, the computation of the probability $p(\beta,\gamma|I)$ can also be decomposed by distributing the weak learners of the multiclass classifiers to several stages.

We organize the whole detector as a graph-structured network shown in Figure 2(c), which alternates the two tasks of pose estimation and background rejection by hierarchically distributing the overhead computation of $p(\beta|I)$. Figure 5 shows the detection algorithm of PBN with a single pose parameter. PBN detection with multiple pose parameters can be implemented in a similar way. The network is evaluated in a top-to-bottom fashion. At each new stage, more weak learners are added to update $p(\beta|I)$, which approximates the true posterior probability more accurately due to its asymptotic property. Based on the newly esti-

mated $p(\beta|I)$, the binary classifiers corresponding to large $p(\beta|I)$ at this stage are evaluated. If a new binary branch never explored by early stages is selected, we trace back to the beginning and re-evaluate the whole branch. If the candidate fails all selected binary classifiers, it is considered as a background window and the computation stops; otherwise, it proceeds to the next stage.

More accurate pose estimation helps to decide whether the candidate window belongs to the foreground, while the binary classifiers help to determine whether it is necessary to continue evaluating $p(\beta|I)$. This way, the positives are evaluated at a minimum number of branches and the negatives are quickly rejected by either the pre-filter or early detector cascades.

5. Experimental results

We applied the PBN classifier in two experiments. In the first experiment, we detected the LV in a 3D echo and estimated the LV orientation using the one-parameter estimation. In the second experiment, we detected the left atrium (LA) in a 2D echo and estimated two pose parameters.

5.1. Left ventricle detection in 3D echocardiogram

We used 50 3D echocardiographic volumes in training and 22 volumes in testing. The LV configurations for all 72 volumes are annotated by an expert. As a preprocessing step, we down-sampled every volume into a quarter size of the original one, giving a volume size of $54 \times 62 \times 67$.

We built a local coordinate system for each volume, with its center at the LV center and its z-axis along the long axis of the LV. Because the LV is roughly symmetric along its long axis, we treated the variation caused by the rotation along the z-axis as intra-class. We concentrated on detecting object position (t_x, t_y, t_z) and estimating two rotation parameters (θ_x, θ_y).

We divided the rotation along the x-axis into three angles $\{-20, 0, 20\}$ and the rotation along the y-axis also into three angles $\{-20, 0, 20\}$. Because the number of discrete poses is small, we used one multiclass classifier to estimate $p(\theta_x, \theta_y|I)$ directly, which corresponds to the type-a structure discussed in Section 3.2. For each angle, we randomly perturbed by a small amount (less than 3 voxels) around the LV center to collect more positives. Also the images caused by pure rotation along the z-axis are added as positive data in order to increase the robustness of the LV detection.

We implemented the following seven approaches: 1) a pose-invariant classifier. This is the same as using only the pre-filter but without adjusting the threshold; 2) a parallel-structured detector which makes decision based on the magnitude of response functions trained independently; 3) a tree-1 detector exploring only one branch which has the largest pose probability; 4) a tree-structured PBN detector exploring several most promising branches but without the pre-filter; 5) a graph-structured PBN detector without the pre-filter; 6) a tree-structured PBN detector with the pre-filter; and 7) a graph-structured PBN detector with the pre-filter. We constructed a common feature pool, based on the 3D local rectangle features [10], from which we selected via boosting relevant features to form the multiclass and binary classifiers. Because the ultrasound volume has a fan-like support region, it is likely that the ground truth LV window contains voxels outside the fan. To deal with this, we followed [4] to use the so-called mask volume which indicates one if a voxel is inside the fan and zero otherwise. To efficiently count the number of within-the-fan voxels in a specific window, we computed the integral volume of the mask volume. We empirically found that using the additional mask treatment introduces 30% extra computation when compared with [12]. Also, The area outside the known fan is not scanned. To have fair comparisons, we used the same discretization of the pose parameter for all seven approaches.

The testing results are presented in Table 1. Denote the estimated parameter by $(\hat{t}_x, \hat{t}_y, \hat{t}_x, \hat{\theta}_x, \hat{\theta}_y)$ and the ground truth by $(t_x, t_y, t_z, \theta_x, \theta_y)$, we calculated the error in translation as $\sqrt{\sum_{a=x,y,z}(\hat{t}_a - t_a)^2}$ and the errors in rotation as $|\hat{\theta}_x - \theta_x|$ and $|\hat{\theta}_y - \theta_y|$. Reported in Table 1 are the median values of the testing errors. The computational speed was measured on a PC with a 2.4GHz CPU and 2GB memory.

From the results, we observe that the rotation-invariant detector is the fastest one, but it lacks accuracy in detection due to the appearance complexity introduced by all rotations. Also, it does not estimate the orientation. The parallel and tree-1 structure are comparable in terms of speed and translation, but the tree-1 structure has an edge over the parallel one in terms of orientation accuracy. The reason why the parallel structure that evaluates all branches is slightly faster than the tree-1 structure that evaluates only one branch is due to the overhead computation of pose estimation, the small number of branches, relative easiness to reject the background patches. However, both their pose estimation is rough due to inability to interpolate. The tree-structured PBN gives better accuracy, benefiting from the pose probability computation that enables proper branch selection and the MMSE estimation of pose parameter, but it is slower than the parallel and tree-1 structures due to the above-mentioned reasons as well as more branches to be evaluated. The graph-structured PBN achieves similar accuracy to the tree-structured PBN but significantly improves the speed as it distributes the overhead computation of pose estimation hierarchically. The pose-invariant pre-filter is effective to reject the background candidates at early stages with no sacrifice of the final estimation. When it is coupled with the tree- and graph-structured PBNs, we achieved

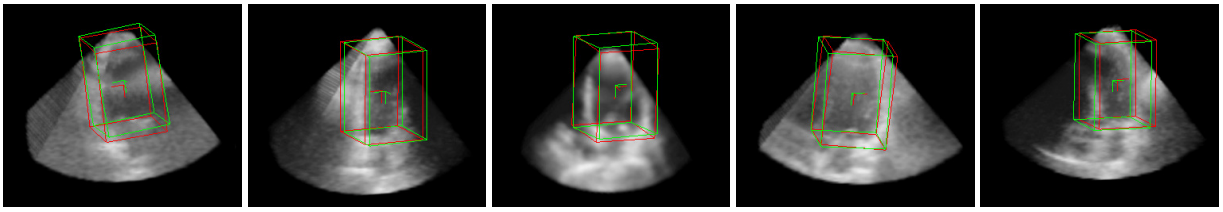


Figure 6. *3D LV detection results obtained by the PBN classifier. The green box is the PBN estimation and the red box is the ground truth.*

the real-time performance for 3D LV detection: processing about 10 volumes per second. To the best of our knowledge, this is the first algorithm that runs in real time to detect object in a 3D volume and estimate its pose as well. Figure 6 shows several detection and pose estimation results obtained by the PBN classifier.

	time(ms)	T(voxel)	θ_x (degree)	θ_y (degree)
pose inv.	83	3.593	n/a	n/a
parallel	223	2.549	9.109	6.976
tree-1 [7]	251	2.828	6.683	5.746
tree	295	2.236	5.481	3.967
graph	170	2.314	5.592	4.40
tree+pre	142	2.236	5.481	3.967
graph+pre	102	2.314	5.592	4.408

Table 1. *Parameter estimation results of 3D LV detection.*

5.2. Left atrium detection in 2D echocardiogram

We used the 2D echo of apical two chamber view (A2C) in the experiment. The A2C view captures only the LV and LA. LA detection in 2D echo is more difficult than LV detection [4] because the LA possesses more shape variations and the LA, being in the far field of the ultrasound probe, is more difficult to image. Refer to Figure 8 for some A2C example images.

Our data set contains 371 2D echo images with expert annotations of the LA, out of which 299 randomly selected images are for training and the remaining 72 for testing. In training, we followed the procedure of Procrustes shape alignment. The rigid transformation between an original contour $\mathbf{x}$ and the mean contour $\mathbf{x_0}$ is defined as:

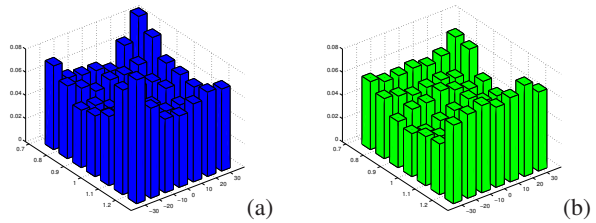
$$\mathbf{x} = SRA\mathbf{x_0} + T, \tag{13}$$

where $S = \text{diag}[s, s]$ is a uniform scaling matrix, R is a rotation matrix defined by rotation angle θ, $A = \text{diag}[1, \alpha]$ defines the aspect ratio of the contour, $T = [t_x, t_y]^T$ is a translation vector.

Among these five parameters $(s, \theta, \alpha, t_x, t_y)$, we treated the rotation θ and aspect ratio α as pose parameters. Based on the prior range and the precision requirement, we divided the rotation into seven angles $\{-30 + j * 10; j = 0, 1, \ldots, 6\}$ and the aspect ratio into seven discrete value $\{-.76, -.84, -.92, 1, 1.08, 1.16, 1.24\}$. Because it is difficult to fit 49 classes into the memory to learn $p(\theta, \alpha|I)$ with one multiclass classifier, we used the type-c structure discussed in section 3.2. The root node computes $p(\theta|I)$ because the rotation causes more distinct variations. The training data are obtained by synthetically transforming aligned data with different rotation and aspect ratio parameters.

Other three parameters are exhaustively searched. Totally 49 binary detector cascades are trained for all poses. In order to allow objects with an in-between pose to pass these detectors, we augmented the positive training data by a small perturbation of the corresponding pose.

We first compared the performance of the type-b and type-c structures for the two-parameter estimation. For each testing data, we synthesized 49 examples with the discretized rotation and aspect ratio parameters. The probability $p(\theta, \alpha|I)$ is computed for the type-b and type-c structures, respectively. The estimate of the rotation parameter θ is the same for the two structures; the MAE is 3.4 degree. We compared the error of the aspect ratio parameter α, as shown in Figure 7: The overall MAE error is 0.057 for type-b and 0.050 for type-c, a 14% improvement over the type-b. The reason is that the variation caused by the rotation, which is quite large, confuses the aspect ratio classifier, making the independence assumption of the type-b structure invalid.



(a) (b)

Figure 7. *The MAE of the aspect ratio in the two-parameter setting: (a) Type-b structure and (c) Type-c structure.*

	time(ms)	T(pix)	θ(degree)	s	α
exhaustive [4]	619.5	5.018	7.082	0.086	0.099
parallel	339.3	5.849	7.351	0.096	0.104
tree-1	145.2	7.925	7.225	0.158	0.147
tree	207.9	4.695	5.482	0.083	0.082
graph	141.3	4.695	5.482	0.083	0.082
tree + pre	104.4	5.027	5.767	0.083	0.083
graph + pre	76.1	5.027	5.767	0.083	0.083

Table 2. *Parameter estimation results of 2D LA detection.*

We then compared the detection and pose estimation results of seven approaches. Again, we used the mask image treatment. We implemented the exhaustive scanning algorithm in [4] for the LA[2], which trains one detector but rotates the image in testing, to take the place of the pose-invariant detector that is unable to estimate the pose. All

[2]Only the LV is originally addressed in [4].

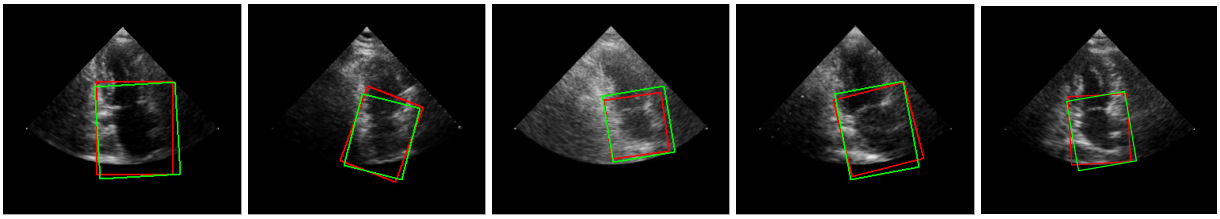


Figure 8. *2D LA detection results obtained by the PBN classifier. The green box is the PBN estimation and the red box is the ground truth.*

other six methods are the same. The detection is on half-sized images, which varies around the size 300×270. The testing results are presented in Table 2. The error in translation is measured by the Euclidean distance. The errors for other parameters are measured by the absolute value. Because there are a few outlier cases, the median error is reported. In each testing image, we selected the scanning candidate with the largest overall detection probability as the detection result and determined the pose using the MMSE estimate from PBN.

From Table 2, we observe that the exhaustive approach is slowest due to computation of integral images for all possible rotations and exhaustive search in the whole parameter space. The parallel structure saves the time for extra computation on integral images. However, comparing response functions of independently trained classifiers is problematic and hence the accuracy is worse than the exhaustive approach. Unlike the 3D LV detection, the tree-1 structure [7], which evaluates only one branches even with the overhead computation of pose estimation, is faster than the parallel structure, which evaluates 49 binary detectors. But, it performs worse than the parallel structure, which is consistent to the observation made in [7]. The tree-structured PBN is slower than the tree-1 structure, but it significantly improves the accuracy: On the average, the error in translation is reduced by 3.23 pixels (40.8% improvement), the error in rotation by 1.743 degree (24.1% improvement), the error in scale by 0.075 (47.5% improvement), and the error in aspect ratio by 0.065 (44.2% improvement). The graph-structured PBN not only matches the same performance improvements induced by the tree-structured PBN over the tree-1 structure but also matches the speed of the tree-1 structure. Using the pose-invariant classifier as a pre-filter of PBN, which rejects about 65% of the candidates, we effectively saved the computation with a negligible performance degradation. The most efficient graph-structured PBN coupled with the pre-filter is able to process about 13 frames per second, more than eight times faster than the exhaustive approach [4].

6. Conclusion

We have described a learning algorithm called probabilistic boosting network (PBN) for joint fast object detection and pose estimation. We have first depicted a boosting-based multiclass procedure that estimates the pose (one/multiple parameters) more accurately and then presented a graph-structured PBN for detecting objects. Coupled with a pose-invariant pre-filter, the graph-structured PBN becomes an effective tool that quickly rejects negatives. Finally, we have implemented a real time algorithm that detects objects from 3D volumes and 2D images.

References

[1] T. Choudhury, J. Rehg, V. Pavlovic, and A. Pentland. Boosting and structure learning in dynamic bayesian networks for audio-visual speaker detection. In *Proc. ICPR*, 2002.

[2] Y. Freund and R. Schapire. A decision-theoretic generalization of online leaning and an application to boosting. *J. Computer and System Sciences*, 55(1):119–139, 1997.

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

[4] B. Georgescu, X. S. Zhou, D. Comaniciu, and A. Gupta. Database-guided segmentation of anatomical structures with complex appearance. In *Proc. CVPR*, 2005.

[5] C. Huang, H. Ai, Y. Li, and S. Lao. Vector boosting for rotation invariant multi-view face detection. In *Proc. ICCV*, 2005.

[6] Y. Jing, V. Pavlovic, and J. Rehg. Efficient discriminative learning of bayesian network classifier via boosted augmented naive bayes. In *Proc. ICML*, 2005.

[7] M. Jones and P. Viola. Fast multi-view face detection. *MERL-TR2003-96*, July 2003.

[8] S. Li and Z. Zhang. FloatBoost learning and statistical face detection. *PAMI*, 26:1112–1123, 2004.

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

[10] Z. Tu, X. S. Zhou, A. Barbu, L. Bogoni, and D. Comaniciu. Probabilistic 3d polyp detection in ct images: The role of sample alignment. In *Proc. CVPR*, 2006.

[11] I. Ulusoy and C. M. Bishop. Comparison of generative and discriminative techniques for object detection and classification. In *Proc. Sicily Workshop on Object Recognition*, 2006.

[12] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proc. CVPR*, 2001.

[13] B. Wu, H. AI, C. Huang, and S. Lao. Fast rotation invariant multi-view face detection based on real adaboost. In *Proc. Auto. Face and Gesture Recognition*, 2004.

[14] S. K. Zhou, J. H. Park, B. Georgescu, C. Simopoulos, J. Otsuki, and D. Comaniciu. Image-based multiclass boosting and echocardiographic view classification. In *Proc. CVPR*, 2006.