Bilayer Segmentation of Webcam Videos Using Tree-Based Classifiers

Pei Yin, *Student Member*, *IEEE*, Antonio Criminisi, John Winn, and Irfan Essa, *Senior Member*, *IEEE*

Abstract—This paper presents an automatic segmentation algorithm for video frames captured by a (monocular) webcam that closely approximates depth segmentation from a stereo camera. The frames are segmented into foreground and background layers that comprise a subject (participant) and other objects and individuals. The algorithm produces correct segmentations even in the presence of large background motion with a nearly stationary foreground. This research makes three key contributions: First, we introduce a novel motion representation, referred to as "motons," inspired by research in object recognition. Second, we propose estimating the segmentation likelihood from the spatial context of motion. The estimation is efficiently learned by random forests. Third, we introduce a general taxonomy of tree-based classifiers that facilitates both theoretical and experimental comparisons of several known classification algorithms and generates new ones. In our bilayer segmentation algorithm, diverse visual cues such as motion, motion context, color, contrast, and spatial priors are fused by means of a conditional random field (CRF) model. Segmentation is then achieved by binary min-cut. Experiments on many sequences of our videochat application demonstrate that our algorithm, which requires no initialization, is effective in a variety of scenes, and the segmentation results are comparable to those obtained by stereo systems.

Index Terms—Computer vision, image understanding, machine learning, decision tree, random forests, boosting, motion analysis.

1 Introduction and Related Work

This paper addresses the problem of extracting a foreground layer from videochat captured by a (monocular) webcam that closely approximates depth segmentation from a stereo camera. The foreground is intuitively defined as a subject of videochat, not necessarily frontal, while the background is literally anything else. Applications for the proposed technique include background substitution, compression, adaptive bit rate video transmission, and tracking. These applications have at least two requirements: 1) robust segmentation against strong distracting events, such as people moving in the background, camera shake, or illumination change, and 2) efficient separation for attaining live streaming speed. This paper focuses on the common scenario of the videochat application.

Image layer extraction has been an active research area [2], [3], [12], [29], [34]. Recent work in this area has produced compelling, real-time algorithms, based on either stereo [13] or motion [8]. Some previously used algorithms require initialization in the form of a "clean" image of the background [28].

Stereo-based segmentation [13] seems to achieve the most robust results, as background objects are correctly separated from the foreground independently from their motion-versus-stasis characteristics. This paper aims at achieving a similar behavior *monocularly* (cf. Fig. 1).

In some monocular systems [28], the static background assumption causes inaccurate segmentation in the presence of camera shake (e.g., for a webcam mounted on a laptop screen), illumination change, and large objects moving in the background. Although the algorithm in [8] precludes the need for a clean background image, the segmentation still suffers in the presence of large background motion. Moreover, initialization is sometimes necessary in the form of global color models.

The work in [35] has started an important line of research in using geometric models (e.g., planar motion) for the segmentation of optical flow fields. However, in the videochat application, such rigid models are not capable of accurately describing the foreground motion. Furthermore, we wish to avoid the complexities associated with optical flow computation.

The algorithm proposed in this paper exploits motion *and* its spatial context as a powerful cue for layer separation, and the correct level of geometric rigidity is automatically learned from training data (see the discussion in Section 7.1). The algorithm benefits from a novel, quantized motion representation (cluster centroids of the spatiotemporal derivatives of video frames), referred to as *motons*. Motons (related to *textons*), inspired by recent research in motion modeling [8] and object/material recognition [18], [22], [27], [32], [36], are combined with *shape filters* [27] to model long-range spatial correlations (shape). These new features prove useful at capturing the visual context and filling in missing, textureless, or motionless regions.

Fused motion-shape cues are discriminatively selected by supervised learning. Key to our technique is a classifier trained on *depth-defined* layer labels, such as those used in a

- *P. Yin is with Microsoft Corp., One Microsoft Way, Redmond, Washington 98052. E-mail: peiyin@microsoft.com.*
- *A. Criminisi and J. Winn are with Microsoft Research Cambridge, Cambridge CB3 0FB, UK. E-mail: {antcrim, jwinn}@microsoft.com.*
- *I. Essa is with the School of Interactive Computing, College of Computing, Georgia Institute of Technology, Atlanta, Georgia 30332. E-mail: irfan@cc.gatech.edu.*

Manuscript received 13 Dec. 2008; revised 5 Oct. 2009; accepted 4 Dec. 2009; published online 1 Mar. 2010.
Recommended for acceptance by B. Triggs.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number TPAMI-2008-12-0855.
Digital Object Identifier no. 10.1109/TPAMI.2010.65.

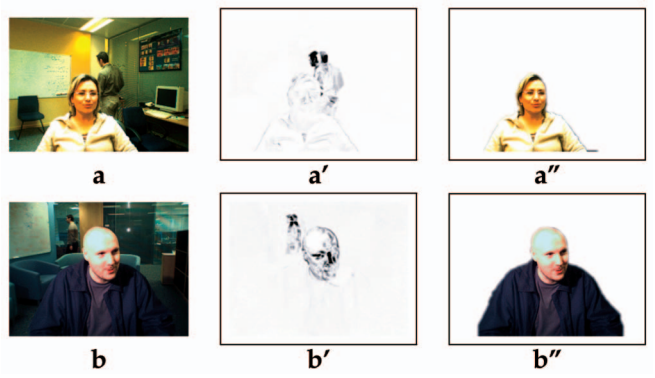

Fig. 1. Achieving robust layer extraction monocularly. (a) and (b) Original frames from the "IU" and "JM" sequences from [13], respectively. (a') and (b') Temporal derivatives (dark indicates large values). The foreground person is nearly stationary, while the background one is moving. In this case, background subtraction techniques or conventional motion-based algorithms would tend to classify the background person erroneously as foreground. Furthermore, inaccurate classification may be produced in the textureless and motionless areas of the foreground. (a") and (b") Segmentation obtained by the proposed algorithm. Correct foreground/background separation has been achieved (the extracted foreground is shown in the original colors).

stereo setting [13], as opposed to motion-defined layer labels, such as in [8] (compare Figs. 2b and 2c). Thus, to induce depth in the absence of stereo while maintaining generalization, the classifier is forced to combine other available cues accordingly.

Combining multiple cues addresses one of the two aforementioned requirements, robustness, for the bilayer segmentation. To meet the other requirement, efficiency, a straightforward way is to trade accuracy for speed if only one type of classifier is used. However, we may be able to achieve efficiency without sacrificing much accuracy if multiple types of classifiers, such as AdaBoost [10], decision trees [23], random forests [7], random ferns [21], and attention cascade [33], are available. That is, if we view these "strong" classifiers as a composition of "weak" learners (decision stumps), we may be able to control how the strong classifiers are constructed to fit the limitations in evaluation time. In this paper, we describe a general taxonomy of classifiers which interprets these common algorithms as variants of a single tree-based classifier. This taxonomy allows us to compare the different algorithms fairly in terms of evaluation complexity (time) and select the most efficient or accurate one for the application at hand.

By fusing motion shape, color, and contrast with local smoothness prior in a conditional random field model [15], [16], we achieve pixelwise binary segmentation through min-cut [5]. The result is a segmentation algorithm that is efficient and robust to distracting events and that requires no initialization.

2 Motons and Shape Filters

This section describes the motion-shape features used in our segmentation algorithm. We build upon the motion-versus-stasis model of [8] and combine it with concepts borrowed from recent studies in object recognition [27]. This work has resulted in a powerful set of features that simultaneously capture motion and its long-range spatial context.

Fig. 2. Ground truth layer labeling in video frames. (a) A frame from a monocular training sequence. In this video, both the closer and farther persons are moving. (b) *Depth*-based layer labeling (white for the foreground, black for the background, and gray for an uncertain region), as used in [13]. Here, only the closest person is labeled as being in the foreground. (c) *Motion*-based layer labeling, as used in [8]. Both moving objects are labeled as being in the foreground. In this paper, we use *only* depth-based labeling, which encourages our monocular system to learn to "imitate" stereo.

2.1 Notation

Given an input sequence of images, a frame is represented as an array $\mathbf{z} = (z_1, z_2, \ldots, z_n, \ldots, z_N)$ of pixels in the YUV color space, indexed by the pixel position n. A frame at time t is denoted $\mathbf{z}^t$. Temporal derivatives are denoted $\dot{\mathbf{z}} = (\dot{z}_1, \dot{z}_2, \ldots, \dot{z}_n, \ldots, \dot{z}_N)$ and computed as $\dot{z}_n^t = \mid G(z_n^t) - G(z_n^{t-1}) \mid$ at each time t with a Gaussian kernel $G(\cdot)$ at a scale of σ_t pixels. Spatial gradients $\mathbf{g} = (g_1, g_2, \ldots, g_n, \ldots, g_N)$, in which $g_n = \mid \nabla z_n \mid$, are computed by convolving the images with the derivatives of Gaussian (DoG) kernels of width σ_s. Here, we use $\sigma_s = \sigma_t = 0.8$, approximating a Nyquist sampling filter. Spatiotemporal derivatives are computed on the Y channel only. Given $\mathbf{O}_m = (\mathbf{g}, \dot{\mathbf{z}})$, the segmentation task is to infer a binary label $x_n \in \{Fg, Bg\}$.[1]

2.2 Motons

Here, we follow a procedure similar to that for constructing textons [18]. First, we compute the two-dimensional $\mathbf{O}_m$ for all training pixels and then cluster the two-dimensional $\mathbf{O}_m$ into M clusters using expectation maximization (EM). The M resulting cluster centroids are called *motons*. An example with $M = 10$ motons is shown in Fig. 3. This operation may be interpreted as building a vocabulary of motion-based visual words. Our visual words capture information about the motion and the "edgeness" of image pixels, rather than their texture content as in textons.

Clustering 1) enables efficient indexing of the joint $(g, \dot{z})$ space while maintaining a useful correlation between g and $\dot{z}$ and 2) reduces sensitivity to noise. We have empirically tested 6, 10, and 15 clusters with multiple random starts. A dictionary size of just 10 motons has proven sufficient, and the clusters are generally stable in multiple runs. Our moton representation also yields fewer segmentation errors than the use of $\mathbf{O}_m$ directly (see the discussion in Section 7.2).

The observation in [8] is that strong edges with low temporal derivatives usually correspond to background regions, while strong edges with high temporal derivatives are likely to be in the foreground. Textureless regions tend to have their Fg/Bg log-likelihood ratio (LLR) close to zero due to uncertainty. Such motion-versus-stasis discrimination properties are retained by our quantized representation; however, they cannot sufficiently separate the moving background from the moving foreground.

1. In this paper, we use Fg and Bg to denote the foreground and the background.

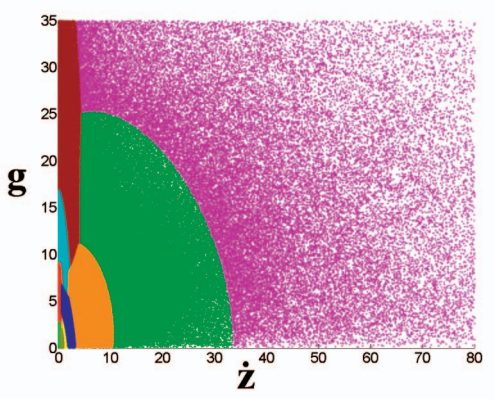

Fig. 3. Motons. Training spatiotemporal derivatives clustered into 10 cluster centroids (motons). Different colors for different clusters.

Given a dictionary of motons, each pixel in a new image can be assigned to its closest moton by maximum likelihood (ML). Therefore, each pixel can now be replaced by an index into our small visual dictionary [36]. An example of the resulting moton map is color coded in Fig. 4b. Then, a moton map can be decomposed into its M component bands, namely "moton bands." Thus, we have M moton bands I^k, $k = 1, \ldots, M$, for each video frame $\mathbf{z}$. Each I^k is a binary image, with $I^k(n)$ indicating whether the nth pixel has been assigned to the kth moton or not. Figs. 4c, 4d, and 4f show some example moton bands. The clusters used in this paper, similar to the textons [27] in vision or a codebook in speech processing [24], will provide efficient indexing and reduce noise as a preprocessing step. However, note that not all moton bands bear an intuitive meaning.

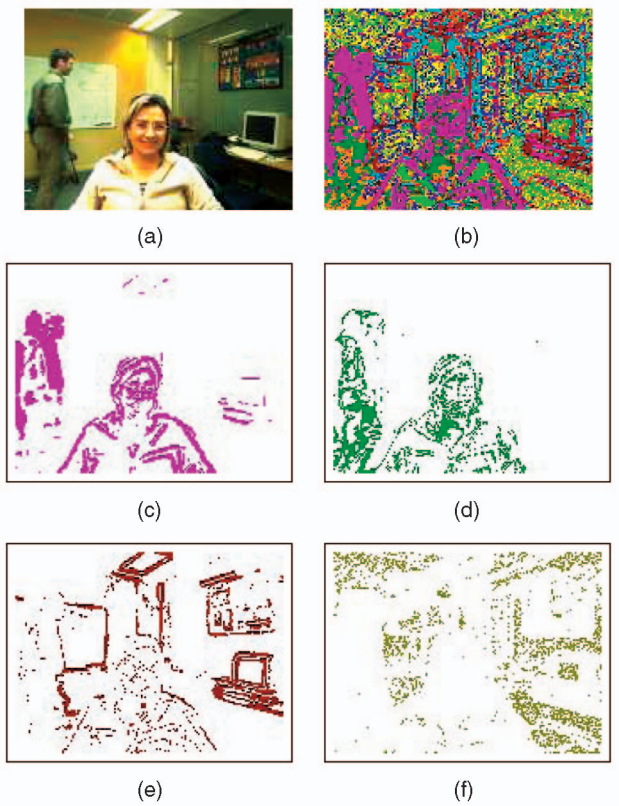

Fig. 4. Motons maps and moton bands. (a) The original frame from the "IU" sequence. (b) The corresponding moton map with $M = 10$ motons. The same color corresponds to the same moton. (c) A moton band showing all of the pixels associated with a "moving-edge" moton. (d) Pixels associated with a moving, "weak-texture" moton. (e) Pixels associated with a "stationary-edge" moton. (f) Pixels associated with a stationary, "weak-texture" moton.

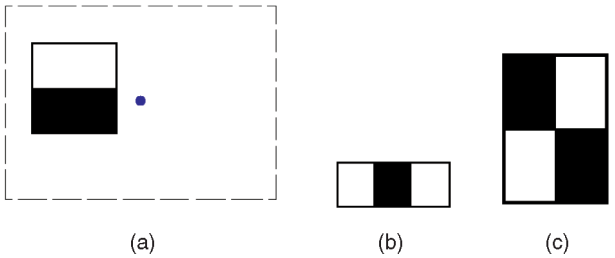

Fig. 5. Some shape filters used in [33], [39], and/or [27]. (a) A shape filter composed of two rectangles (shown in a detection window centered at a particular pixel). (b) and (c) Shape filters composed of more rectangles.

2.3 Shape Filters

In our videochat application, the foreground object (usually a person) moves nonrigidly yet in a structured fashion. This section first briefly explains the shape filters and then shows how to capture the spatial context of motion adaptively.

To detect faces using the spatial context of an image patch (detection window), [33] builds a overcomplete set of Harr-like shape filters, shown in Fig. 5. The sum of the pixel value in the white rectangles is subtracted by the sum of the pixel value in the black rectangles. The result value of the subtraction is the feature response used for classification. In [33], the shape filter is applied only to a gray-scale image, that is, the "pixel value" in that paper corresponds to image intensity. In speaker detection [39], similar shape filters are applied to a gray-scale image, a frame difference, and a frame running average. In object categorization, texton layout filters [27] generalize the shape filters by randomly generating the coordinates of the white and black rectangles and apply the filters to randomly selected texton channels (bands). Experiments in multiple applications [27], [33], [39] have shown that the shape filters produce simple but effective features for classification. In order to efficiently compute the feature value of the shape filters, integral image processing [33] can be used. An integral image $ii(x, y, k)$ is the sum of the pixel value in the rectangle from the top left corner $(0, 0)$ to the pixel (x, y) in image band k ($k = 1$ in [33]). Therefore, computing the sum of the pixel value for an arbitrary rectangle with the top left at (x_1, y_1) and the bottom right at (x_2, y_2) in image band k requires only three operations $ii(x_2, y_2, k) - ii(x_1, y_2, k) - ii(x_2, y_1, k) + ii(x_1, y_1, k)$, that is, constant time complexity for each rectangle regardless of its size in the shape filter and band index k.

To infer Fg/Bg for a pixel n in this paper, we also use the contextual information within a (sliding) detection window centered at n.[2] The size of the detection window is about the size of the video frame and fixed for all the pixels. Within a detection window, similar to [27], a shape filter is defined as a moton-rectangle pair (k, r), with k indexing in the dictionary of motons and r indexing a rectangular mask,

2. Regions outside the actual video frame are padded with zero.

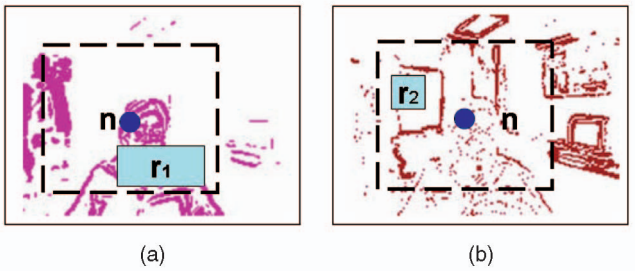

Fig. 6. Shape filters applied to moton bands. (a) and (b) Two moton bands with rectangular masks r_1 and r_2 (centered at the same pixel n) superimposed. Given n and the shape filter (k, r), $v_n(k, r)$ counts the number of pixels associated with the texton k within rectangle r (see text).

shown in Fig. 6. First, we denote all of the possible shape filters within a detection window as $\mathcal{S}^*$ and then define a whole set of d shape filters $\mathcal{S} = \{(k_i, r_i)\}$, $i = 1, \ldots, d$, by randomly selecting moton-rectangle pairs from $\mathcal{S}^*$. For each pixel position n, we compute the associated feature ψ_n as follows: Given the moton k, we center the detection window at n and count the number of pixels in I^k that fall in the offset rectangle mask r. This count is denoted $v_n(k, r)$. The feature value $\psi_n(i, j)$ is obtained by simply subtracting the moton counts collected for the two shape filters (k_i, r_i) and (k_j, r_j), i.e., $\psi_n(i, j) = v_n(k_i, r_i) - v_n(k_j, r_j)$. The moton counts v_n can be computed efficiently by one integral image for every moton band I^k. Therefore, given S, by randomly selecting i, j, and k ($1 \leq i, j \leq d$, $1 \leq k \leq M$), a total of $d^2 \cdot M^2$ features can be computed at every pixel n.

Next, our features are discriminatively selected and combined by a classifier for the estimation of our motion-shape unary potentials (U^{MS} in Section 5). The following section presents a taxonomy of tree-based classifiers and shows how common tree-based classifiers may be interpreted as instances of the same general algorithm. Such a taxonomy then helps us to select classifiers that perform best in our application.

3 The Tree-Cube Taxonomy

As described in Section 1, common classification algorithms, such as decision trees [23], boosting [11], and random forests [7], share the fact that they build "strong" classifiers from a combination of "weak" learners, often just decision stumps. The main difference among these algorithms is the way the weak learners are combined, and exploring the difference may lead to an accurate classifier that is also efficient enough to fit the limitations of the evaluation time. This section presents a useful framework for constructing strong classifiers by combining weak learners in different ways to facilitate the analysis of accuracy and efficiency.

The three most common ways to combine weak learners are 1) hierarchically (H), 2) by averaging (A), and 3) by boosting (B), or more generally adaptive reweighting and combining (ARCing) [6]. In Fig. 7, the origin represents the weak learner (e.g., a decision stump), and axes H, A, and B represent those three basic combination "moves."

1. The H-move hierarchically combines weak learners into decision trees. During training, a new weak learner is iteratively created and attached to a leaf node, if needed, based on information gain. Evaluation of one instance includes only the weak learners along the corresponding decision path in the tree. It can be shown that the H-move reduces classification bias [4].
2. The B-move, instead, linearly combines weak learners. After the insertion of each weak learner, the training data are reweighted or resampled [23] such that the last weak learner has 50 percent accuracy in the new data distribution. Evaluation of one instance includes the weighted sum of the outputs of *all* of the weak learners in the booster. Examples of the B-move include AdaBoost and gentle boost. Boosting reduces the empirical error bound by perturbing the training data [11].
3. The A-move creates strong classifiers by averaging the results of many weak learners. Note that all of the weak learners added by the A-move solve the same problem while those sequentially added by the H and B-moves solve problems with a different data distribution. Thus, the main computational advantage in training is that each weak learner can be learned independently and in parallel. When the weak learner is a random tree, A-move gives rise to random forests. The A-move also reduces classification variance [4].

Paths, not vertices, along the edges of the cube in Fig. 7 correspond to different combinations of weak learners and thus (unlimited) different strong classifiers. If we restrict each of the three basic moves to being used once at most, the tree taxonomy produces three order-1 algorithms (excluding the base learner itself), six order-2, and six order-3 algorithms, as listed in Table 1. Many known algorithms, such as boosting (B), decision trees (H), booster of trees (HB), and random forests (HA), are conveniently mapped into paths through the tree cube. Also note that the widely used attention cascade [33] can be interpreted as a one-sided tree of boosters (BH). The tree-cube taxonomy also enables us to explore new algorithms (e.g., HAB) and compare them to other algorithms of the *same* order (e.g., BHA).

Next, we explore which classifier performs best for our video segmentation application. Following the tree-cube taxonomy, we focus on comparing three common order-2 models: HA, HB, and BA, that is, we compare random forests (RFs) of trees, booster of trees (BT), and ensemble of boosters (EB) to evaluate the behavior of different moves.[3] As a sanity check, we also evaluate the performance of a common order-1 model B, namely, the booster of stumps (using gentle boost, denoted as GB). From this elaborate comparison, we gained a number of insights into the design of tree-based classifiers presented in Section 7. These insights, such as bias/variance reduction, accuracy/efficiency trade-off, the complexity of the problem, and the labeling quality of the data, have motivated the design of an order-3 classifier, HBA, in Section 7.4.

3. Projecting the bias-variance analysis in Section 7.3.1 onto the tree-cube taxonomy suggests that AB and AH may not be favorable in building a classifier with a low-test error. Evidence in Section 6.1.1 also shows that the attention cascade [33] version of BH is unlikely to achieve both accuracy and efficiency in this application at the same time.

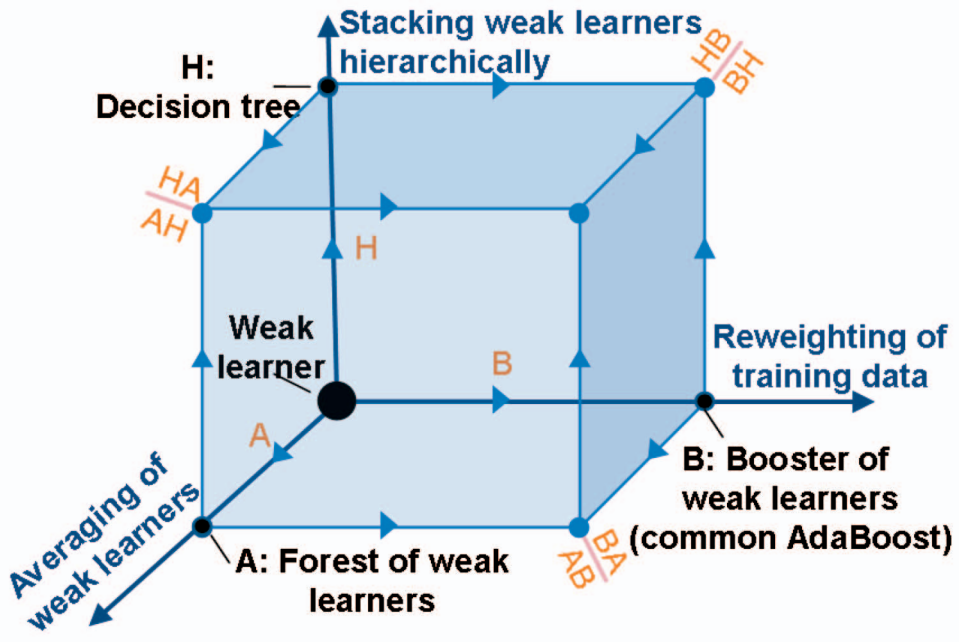

Fig. 7. The tree-cube taxonomy of classifiers captures many classification algorithms in a single structure (see text, for details).

4 Random Forests versus Booster of Trees versus Ensemble of Boosters

The base weak learner used in this paper is the widely used decision stump. A decision stump applied to the nth pixel takes the form $h(n) = a \cdot \mathbf{1}(\psi_{\mathbf{n}}(\mathbf{i},\mathbf{j}) > \theta) + \mathbf{b}$, in which $\mathbf{1}(\cdot)$ is a 0-1 indicator function and $\psi_n(i,j)$ is the shape filter response for the ith and jth shape filters (as described in Section 2). A positive value of the real-valued $h(n)$ output indicates that pixel n belongs to Fg and vice versa. Now, we look at different ways of combining stumps into strong classifiers. We begin with the H-move.

4.1 Decision Tree

When training a tree, we compute θ, a, and b of a new decision stump at each iteration for either the least-square error [27] or the maximum entropy gain, as described later. During testing, the output $F(n)$ of a tree classifier is the output of the leaf node.[4] Next, we analyze the details of the B-move.

4.2 Gentle Boost

Out of the many variants of boosting, here we focus on the Gentle Boost algorithm [11] because of its robustness properties [19], [30]. For the nth pixel, a strong classifier $F(n)$ is a linear combination of stumps $F(n) = \sum_{\ell=1}^{L} h_\ell(n)$. The details of the algorithm can be found in [11]. In this paper, we employ gentle boost as the B-move (in Fig. 7) algorithm for GB, BT, and EB.

4.3 Random Forests

A forest is made of many trees, and its output $F(n)$ is the average of the output of all the trees (the A-move). A random forest is an ensemble of decision trees trained with *random features*. In this case, each tree is trained by adding new stumps in the leaf nodes in order to achieve maximum information gain. However, unlike boosting, the training data of RF are not reweighted for different trees. In the more complicated order-3 classifier HBA, the training data are only weighted within each BT, but the BTs are constructed independently. RF has been applied to the recognition problems, such as OCR [1] and keypoint recognition [17] in vision.

4.4 Randomization

The tree-based classifiers are effectively trained by optimizing each stump on only a few (1,000 in our implementation) randomly selected shape filter features. This reduces the statistical dependence between weak learners [1] and provides increased efficiency without significantly affecting their accuracy [9], [27].

TABLE 1
Tree-Cube Classifiers

Path	Classification algorithm
A	voting by committee [4]
B	booster of stumps [33]
H	decision tree [23]
HA	forest of trees (decision forest) [7]
HB	booster of trees [11]
AH	tree of forests (of stumps)
AB	booster of forests (of stumps)
BA	ensemble of boosters
BH	tree of boosters [33]
HAB	booster of forests of trees
HBA	ensemble of boosters of trees
BAH	tree of ensembles of boosters (of stumps)
BHA	ensemble of trees of boosters
ABH	tree of boosters of forests (of stumps)
AHB	booster of trees of forests (of stumps)

Fifteen algorithms encoded in the taxonomy of Fig. 7, if we restrict each of the three basic moves to be used once at most.

4. Following the routine in [23] the output of the leaf node of a tree is the expected value of the class, which is determined empirically. However, gentle boost requires the output be a least-square fit of a weighted distribution of the training labels. We follow this requirement instead, which actually affects the results in a minor way, in our experiments in which gentle boost is used.

In all three algorithms, the classification confidence is computed by softmax transformation [11], [27] as follows:

$$\widetilde{P}(x_n = Fg|\mathbf{O}_m) = \frac{\exp(F(n))}{1+\exp(F(n))}. \quad (1)$$

Next, we describe how these motion-shape-based classifiers are combined with color, contrast, and spatial smoothness to obtain binary segmentation.

5 Layer Segmentation

Segmentation is cast as an energy minimization problem in which the energy to be minimized is similar to that in [13], the only difference being that the stereo-match unary potential U^M is replaced by our motion-shape unary potential U^{MS}:

$$U^{MS}(\mathbf{O}_m, x; \Theta) = \sum_{n=1}^{N} log(\widetilde{P}(x_n|\mathbf{O}_m)), \quad (2)$$

in which $\widetilde{P}$ is from (1). The CRF energy is as follows:

$$\begin{aligned} E(\mathbf{O}_m, z, x; \Theta) &= \gamma_{MS} U^{MS}(\mathbf{O}_m, x; \Theta) \\ &+ \gamma_C U^C(z, x; \Theta) + V(z, x; \Theta). \end{aligned} \quad (3)$$

Similarly to [13], U^C is the color potential (a combination of global and pixelwise contributions), and V is the widely used contrast-sensitive spatial smoothness term [8], [13], [25]. Model parameters are incorporated in Θ. Relative weights γ_{MS} and γ_C are optimized discriminatively from the training data. The final segmentation is inferred by a binary min-cut as in [8], [13]. No complex temporal models such as [8] are used here. Finally, because many segmentation errors are caused by strong background edges, background edge abating [28], which adaptively "attenuates" background edges, could also be exploited here if a pixelwise background model were learned on the fly.

6 Experimental Results

Our new motion-shape likelihood in (2) is validated in Section 6.1 while the accuracy of the segmentation computed by the complete CRF model in (3) is assessed in Section 6.2. We have collected a database of 28 monocular video sequences,[5] for which we have pixelwise labeled every fifth or tenth frame into the foreground, the background, and the uncertain regions (difficult, mixed-pixel regions) according to their distance from the camera (Fig. 2b). In our experiments, we chose seven clips randomly for training and two clips for validation. Then, we select 46 labeled frames from the training clips to train the tree-based classifiers and 10 labeled frames for validation, and all of the other 426 labeled frames in the remaining 19 testing clips are used for testing.

6.1 Comparing Unary Classifiers

GB, EB, and BT were trained by minimizing the empirical loss as required by boosting while RF was trained by maximizing the information gain as required by C4.5 [23]. All three algorithms share the same set of motons. Their testing errors are then measured and compared with one another. The ensemble size for GB, EB, BT, and RF is set to 195 stumps, 47 boosters, 19 trees, and 47 trees, respectively, to maximize the accuracy of the validation set. The trees in BT and RF have 50 nodes, *optimal for BT* on the validation set.

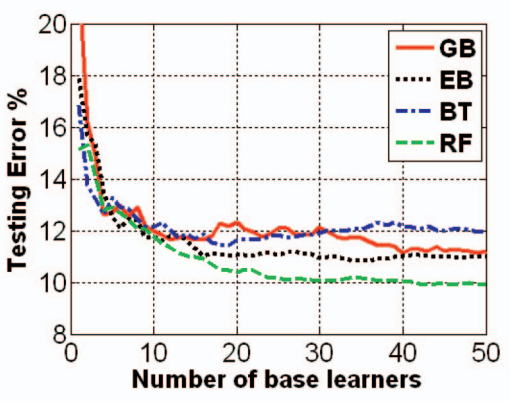

Fig. 8. Comparing accuracy of classifiers. Testing unary accuracy with respect to the complexity of the ensembles in one trial. Five trials have been taken and RF has consistently outperformed the GB, EB, and BT algorithms.

We then evaluate the unary classification accuracy with the 426 testing frames, classify the pixels[6] into the foreground and the background by thresholding at $F(n) = 0$, average the error rate over $426 * 160 * 120 = 8.18 * 10^6$ pixels, and measure the evaluation efficiency (frame per second) with our nonoptimized MATLAB code without parallelization.

6.1.1 Accuracy of Unary Potentials

Fig. 8 compares the classification accuracy of GB, BT, and RF when changing the number of base learners. Assuming balanced trees, evaluating one binary decision tree with 50 nodes roughly equals evaluating $\log_2 50 \approx 6$ stumps (depending on the balance of the tree). Thus, we have scaled down the curve of GB along the x-axis by a factor 6 so that the expected number of stumps evaluated is the same for GB, BT, and RF. Of the three, random forests consistently yield the lowest testing errors.

From the GB curve in Fig. 8, we can also see that not many pixels can be correctly classified with a few stumps. Therefore, we would not expect an attention cascade [33] to significantly speed up our application.

Table 2 compares the accuracy of our motion-shape unary potentials U^{MS} with the stereo potentials in [13] and the motion potentials of the monocular system in [8]. Our motion-shape unary potentials lead to accuracy comparable to those of stereo[7] and superior to the motion-based potentials. Fig. 9d also visualizes the classification accuracy

5. The data are available at http://research.microsoft.com/vision/cambridge/i2i/DSWeb.htm. For the six sequences that are captured in stereo setting, only the left-camera videos are used here, and *only* for testing.

6. The final segmentation results are significantly improved by integrating color, contrast, and spatial priors using the CRF model, shown in Section 6.2 and Fig. 1. Note that all of the other results are based on the motion-shape unary U^{MS} only.

7. Here, the accuracy of the stereo likelihood has been improved with respect to [13] by setting the LLR to zero in low texture areas (uncertain for stereo). The pixelwise stereo error rate would increase to 17.51 percent without such postprocessing. This further illustrates the importance of shape information in our bilayer segmentation application.

TABLE 2
Comparison with State of the Art

Stereo	Monocular				
Stereo [13]	Motion [8]	Learnt motion-shape			
		RF	BT	GB	EB
5.55%	23.66%	**9.93%**	11.42%	11.76%	10.95 %

Comparison between the proposed classifiers and existing stereo and monocular unaries.

of one of the frames in testing video sequences (shown as the foreground).

6.1.2 Accuracy versus Efficiency

Table 3 compares the four classifiers in terms of both accuracy and speed. The upper chart reports the lowest classification error achieved and the corresponding frame rate for each of the three algorithms at their "optimal" parameter setting according to validation. Having confirmed that RF produces the lowest errors, we then evaluate the *improved* speed of RF when it is allowed to produce the same error level as GB, BT, and EB in the first two columns of the lower chart. In the last two columns of the lower chart, we reduce the size of the RF ensemble and observe its *suboptimal* classification error when RF evaluates at the same speed as GB, EB, and BT. In all cases, RF outperforms the other classifiers.

6.2 Assessing Segmentation Accuracy

This section analyzes the segmentation results obtained by the full CRF model with U^{MS} estimated by RF.

6.2.1 Qualitative Evaluation

Fig. 9 compares the segmentation of the "IU" test sequence obtained by our algorithm with those in [8] and [13]. In this sequence, two people walk behind the person in the foreground. Varying the scene illumination constitutes a further source of difficulty. The motion-based method in [8] classifies the people in the background as being in the foreground (as it is designed to do). The proposed algorithm instead produces a segmentation similar to that of the stereo system in [13], in which background motion is effectively ignored. Figs. 1, 10, 11, and 12 provide more segmentation results.[8]

6.2.2 Quantitative Evaluation

Fig. 13 shows segmentation errors with respect to ground truth for four randomly chosen sequences of our test data. The median error is around or below one percent. Median errors for 10 randomly chosen test sequences are also reported in Table 4.

6.2.3 Automatic Initialization

In our algorithm, segmentation of past frames affects only the current frame from the learned color model [13]. Having little memory helps minimize problems such as drifting, and enables the system to (re)initialize itself.

Fig. 14 illustrates how the system initializes itself. At the beginning, the subject is stationary, and the segmentation is inaccurate. However, a small motion of the head is sufficient to achieve the correct segmentation (Fig. 14d). This "burn-in" effect may also be observed for a different test sequence in the error plot in Fig. 13d.

The plot in Fig. 13a demonstrates how our algorithm can recover automatically from possible mistakes. In fact, in frames 60-85 of the "JM" sequence, the subject leans very close to the camera and so the image looks very different from the training frames, and segmentation errors occur. The segmentation recovers promptly following this error.

6.2.4 Inaccurate Segmentation

Figs. 15a' and 15b' show examples of inaccurate segmentation. Under harsh lighting conditions, unary potentials may not be very strong; thus, the Ising smoothness term may force the segmentation to "cut through" the shoulder and hair regions. Similar effects may occur in stationary frames. Noise in the temporal derivatives also affects the results. This situation can be detected by monitoring the magnitude of the motion; in addition, enabling a temporal model such as that in [8] may help reduce the problem.

7 Discussion and Extensions

In this section, we discuss and experimentally justify several previous claims we made about the tree-based classifiers and the motons.

7.1 Learning the Correct Level of Geometric Rigidity

A straightforward bilayer segmentation approach in the videochat application is to manually build a foreground model with a "template," such as the "spring model" [38] in face recognition, to model the level of geometric rigidity of the human upper body. We believe that such template-based methods are generalized by our tree-based classifiers in the following sense: Prior knowledge employed by the template approach can be learned by tree-based classifiers with sufficient depth; furthermore, variability in the data can be maintained by a forest with a sufficient number of trees that corresponds to a rather "adaptive" template. In this section, we experimentally validate such a claim.

Let RF46 denotes the RF classifier trained on all 46 training frames in Section 6, and RF7 denotes a separate RF classifier trained on the seven frames from training video GTTS01 used by RF46. Due to the small number of training examples, the training accuracy of RF7 stops improving after accumulating 10 trees. Thus, for both RF7 and RF46, we use only the first 10 trees for a fair comparison. For three video frames in Figs. 16a, 16b, and 16c, we compute the bilayer segmentation LLR by RF7 and RF46, in which foreground predictions ($\mathrm{LLR} > 0$) are colored with green and background predictions ($\mathrm{LLR} < 0$) are colored with magenta (dark indicates large values). The first frame, Fig. 16a, is from the same training video clip used for RF7 and RF46, but not used in the training. Therefore, this frame is very similar to the training frames. The other two frames, Figs. 16b and 16c, are from the test clips. These segmentation results show that RF7 fits the "familiar" data better than RF46. RF7 overfits the training data and memorizes a rigid head-shoulder template. In contrast, the RF46 classifier is more robust toward unseen

8. These sequences are randomly chosen.

Fig. 9. Segmentation results on the "IU" test sequence. (a) Original, (b) stereo-based segmentation from [13] (after CRF smoothing), (c) monocular segmentation from [8] (after CRF smoothing), in which background motion pops into the foreground, (d) monocular segmentation from the proposed binary classifier (without CRF smoothing), and (e) monocular segmentation from the proposed algorithm (after CRF smoothing).

TABLE 3
Comparing Testing Accuracy and Efficiency for GB, EB, BT, and RF in One of Five Testing Trials (See Text)

Algorithm	GB (best)	EB (best)	BT (best)	RF (best)
Classification error (%)	11.76	10.95	11.42	**9.93**
Speed (fps)	1.2	0.8	2.9	1.2

Algorithm	RF (same err as GB and BT)	RF (same err as EB)	RF (same speed as GB and EB)	RF (same speed as BT)
Classification error (%)	11.23	10.90	**9.93**	**10.11**
Speed (fps)	**7.7**	**3.8**	1.2	2.9

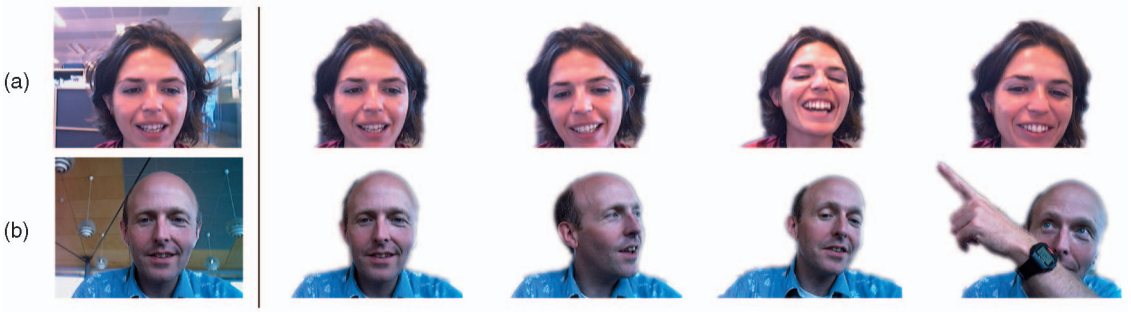

Fig. 10. Segmentation results. (a) An original frame and four segmented frames for test sequence "41." (b) An original frame and four segmented frames for test sequence "54." Border matting [25] could be applied here to improve the hair regions. This paper is concerned with binary segmentation only.

Fig. 11. More segmentation results. (a) and (b) Original frames from test sequence "56," in which the picture on the TV set changes. (a') and (b') Corresponding segmentations. (c')-(d') More segmentation results on test sequence "43" and "50."

sequences, especially with exotic movements such as those in Fig. 16b. This experiment illustrates that RF46 has learned a more flexible geometric template with "the correct level of rigidity."

7.2 Motons as An Effective Motion Representation

Motons enable an efficient indexing of motion cues and reduce noise for bilayer segmentation. To show that the proposed motons are effective features representing motion, we have conducted two experiments. In the first, we built the bilayer GB/EB/BT/RF classifiers in the same manner as we did in Section 6, but the features were $\mathbf{O}_m$ directly without clustering. This variation created 5 to 15 percent more error than their moton-based counterparts. The second experiment used random motons, that is, $\mathbf{O}_m$ were assigned with random moton labels and the mean and the variance of the motons computed accordingly without EM. GB/EB/BT/RF classifiers built using

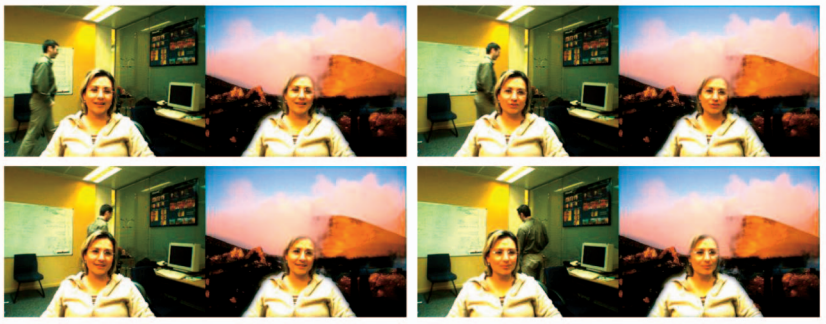

Fig. 12. Background substitution on the "IU" test sequence from [13]. Original and corresponding segmented frames with background substitution. People moving behind the person in the foreground are correctly classified as being in the background.

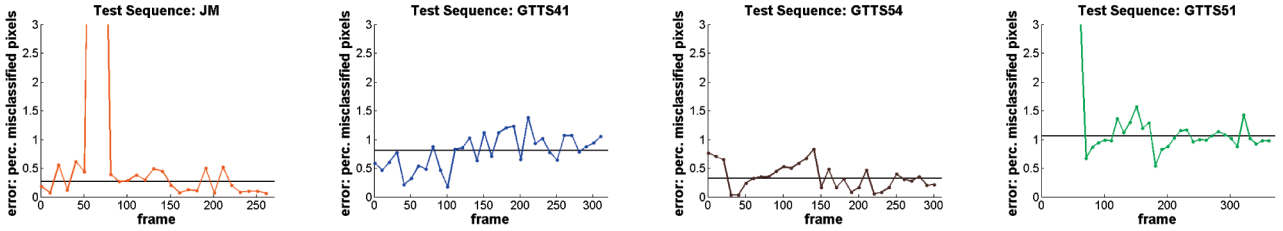

Fig. 13. Accuracy of segmentation. (a) The percentage of misclassified pixels on the "JM" test sequence from [13]. Note how the system promptly recovers from possible mistakes. The median error (horizontal line) is well below 0.5 percent. (b)-(d) The percentage of misclassified pixels for test sequences "41," "54," and "51," respectively. (d) After an initial "burn-in" period, the segmentation converges to a good accuracy level (around or below 1 percent of misclassified pixels).

random motons exhibited a 20 to 30 percent higher error rate. These two experiments prove that motion and its spatial context can be compactly represented by motons for bilayer segmentation.

7.3 Generalization Ability of Different Tree-Based Classifiers

7.3.1 Bias and Variance Decomposition

For a classification algorithm, bias describes its modeling power while variance describes its stability [14]. Note that bias and variance are different than the mean error and the variance of error. The bias-and-variance decomposition of the RF, GB, EB, and BT classifiers at similar evaluation complexity (without parallelization) in Table 5 helps us to understand the behavior of the three basic combination moves and the nature of our task. This decomposition is computed from five trials on the testing set, the results of which are discussed below.

1. BT and RF yield lower bias than GB and EB.

   The linear combination property of the B and A-moves requires that the decision boundary of the classification task be additive in terms of the decision boundary of the weak learners, i.e., the capacity of the base learning algorithm matches the complexity of the problem [11]. By moving along the H direction, bigger trees that are capable of modeling higher order interaction between variables are constructed. For example, a decision stump contains only one feature while the typical decision path of a 50-node binary tree equals a conjunction term of five to six features. Therefore, result 1 indicates that the segmentation task is complex, and the decision boundary is better approximated by deeper trees using the H-move.
2. BT has lower bias than RF. This result confirms that the B-move achieves additional reduction in bias by aggressively perturbing the training process so that it focuses on difficult samples [7], which, however, sometimes results in overfitting (see Section 7.3.2).
3. EB and RF have the lowest variance. Boosting persistently increases the minimum margin (of a few incorrectly classified samples) at a potential cost of decreasing the average margin (over all training data) [37]. Therefore, boosting does not generalize well in the presence of *label noise*.[9] In contrast, EB and RF are quite robust to such kind of noise. EB trades empirical risk with structural risk [31] by constructing a relatively small booster. In RF, the effect of a few mistakenly labeled samples is locally restricted to particular leaf nodes without sacrificing the accuracy of other nodes or trees. Therefore, not surprisingly, overfitting causes higher error in the B-move than in the A-move. Similar phenomena have also been reported in [7].

These insights also suggest that AB and AH may not be favorable in building an efficient classifier with a low-test error.

TABLE 4
Segmentation Errors for 10 of the Test Sequences

Test Sequence	41	43	50	51	54
Segmentation Error (%)	0.80	**0.02**	1.31	1.06	0.33
Test Sequence	56	58	60	IU [13]	JM [13]
Segmentation Error (%)	0.93	0.79	**6.33**	2.56	0.27

9. Label noise is unavoidable in segmentation problems.

Fig. 14. Self-initialization. (a) Original frame from test sequences "58." Because of harsh lighting conditions, the segmentation problem is challenging. (b)-(d) Segmentation for different frames in chronological order. After an initial "burn-in" period, the algorithm converges to the correct Fg/Bg separation. A "clean" background image or other types of initialization was not necessary.

Fig. 15. Some inaccurate segmentation. (a) A frame from test sequence "41." (b) A frame from test sequence "60." (a') and (b') corresponding segmentations. Strong background contrast and bad scene illumination produce inaccurate segmentation in the hair region of (a') and in the shoulder region of (b'), respectively. Note the errors in Table 4.

7.3.2 The Error Divergence Graph of the Classifiers

We plot the training error and the divergence of the training and the testing errors (computed by the testing error minus the training error) in Fig. 17. The greedy optimization of boosting (GB and BT) effectively reduces the empirical error in training. However, boosting is subject to more severe overfitting than RF as training proceeds. Note that EB, constructed by 50 short ensembles of size 6, shares the spirit of "voting by committee" like RF, rather than the aggressive optimization of GB. Therefore, the divergence of the training and the testing errors for EB is relatively lower. However, since each short booster in EB is unable to approximate the complicated decision boundary well, EB has a higher error in both training and testing than RF.

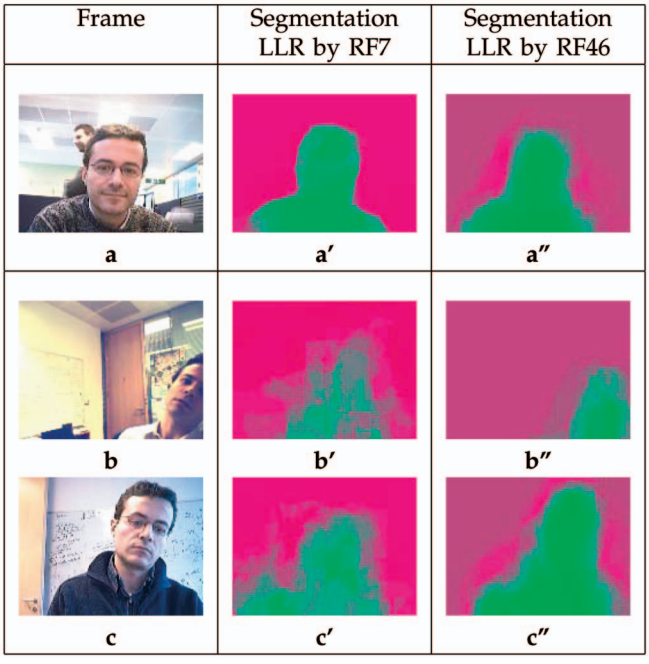

Fig. 16. Diversity in training data affects the model flexibility in testing. (a)-(c) Original frames from the sequence "01," "AC," and "15." "01" is a training sequence, while the other two are testing sequences. (a')-(c') Bilayer segmentation LLR by 10-tree RF trained on "01" only (seven frames, where the foreground head mostly stays steady) (a')-(c') Motion-shape likelihood by 10-tree RF trained on all the 46 training frames.

7.4 Using the Tree-Cube Taxonomy to Design an Order-3 Classifier

Our experiments show that the H-move creates a reasonably complicated decision boundary that matches the bilayer segmentation task; the B-move maneuvers the base learners and reduces the bias even for the HB combination, but it suffers from noise; the A-move is very effective in reducing the classification variance for HA and BA. Projecting these observations onto the tree-cube taxonomy motivates us to combine the merits of the three moves. Our intuition is to reduce the bias first and to obtain a proper fit to data from overfitted classifiers later. Therefore, we use H (decision trees) as the base learners; then, we let B search for the optimal linear combination of the trees to approximate the empirical decision boundary and, finally, utilize A to reduce overfitting. This yields an order-3 classifier HBA: an ensemble of boosted trees (EBT). We construct an EBT with 10 independent BTs, in which each BT has 19 trees obtained by gentle boost. The testing error and the evaluation speed of the unary potential computed by EBT with the other four classifiers are compared in Table 6. As we expected, the error[10] of HBA (EBT) is lower than that of HB (BT), HA (RF), or BA (EB). The HBA combination harnesses the efforts of H-move, B-move, and A-move from the tree cube. However, this order-3 ensemble, which contains $19^*10 = 190$ trees, evaluates much slower than the other four classifiers. Thus, to attain live streaming speed for EBT, we must employ proper parallelization.

10. Since the classification output of EBT is the sum of the outputs of S independently trained BTs, the testing error of EBT also estimates the classification bias of BT. Therefore, the bias estimation for BT using $S = 5$ in Table 5 is about one percent higher than that using $S = 10$ here. However, this minor difference does not affect our conclusion in Section 7.3.1.

TABLE 5
Bias/Variance Analysis

Method	RF	BT	GB	EB
Bias (%)	10.20	9.95	10.31	11.23
Variance (%)	0.39	2.09	1.48	0.11

Bias and variance of RF, BT, GB, and EB.

7.5 Random Forests and Random Ferns

Another tree-based classifier that has become popular recently is "random ferns" [21]. Random ferns can be considered a multidimensional stump and a restricted version of random forests. Without losing generalizability, we assume a binary classification in our discussion below, which can easily be extended to a multiclass case.

For a binary fern of depth d, the same d tests are applied to any testing samples, which provides an ordered binary coding of the output bins. The posterior outputs are computed according to the empirical posterior distribution at the corresponding bins. In contrast, a binary forest of depth d contains d (as a degenerated tree) to $2^{d-1}-1$ (as a complete binary tree) tests. Different tests may be evaluated for different samples according to their "decision path" in the tree.

The ferns trade efficiency with flexibility in the following ways: 1) The ferns do not store any intermediate nodes other than the leaf nodes, and 2) all of the tests of all of the ferns can be evaluated in parallel for efficient processing, while the tests of the same tree have to be evaluated sequentially in the forests. However, the trees are more flexible for modeling data distributions. We usually do not expect that all the 2^{d-1} bins of ferns will be "useful" for several reasons. First, the posterior for many bins may be extremely biased toward one class such that a confident decision can be made with less than d tests. For example, a degenerated decision tree (e.g., an attention cascade [33]) is able to achieve surprisingly high accuracy and efficiency in face detection [33]. Second, the ferns require that the same tests be applied to all the samples. Thus, random ferns are less likely to overfit than forests with the same depth; however, this restriction may also demand deeper or "fatter" ferns to achieve empirical risk comparable to that of the forests.

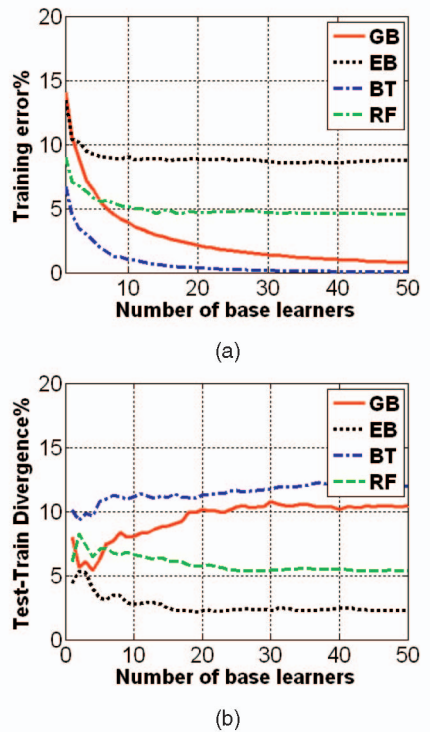

Fig. 17. Comparing the generalization ability of classifiers. (a) Training unary accuracy with respect to the complexity of the ensembles in one trial (the same trial as that in Fig. 8). (b) The divergence graph of testing error and training errors (see text.)

TABLE 6
Revisit: Comparing Testing Accuracy and Efficiency for GB, EB, BT, RF, and EBT (See Text)

Algorithm	GB (best)	EB (best)	BT (best)	RF (best)	EBT (best)
Test error (%)	11.76	10.95	11.42	9.93	**9.83**
Speed (fps)	1.2	0.8	2.9	1.2	**0.29**

Whether random ferns or random forests have lower structural risk depends on the application and the data. Superior classification by ferns is reported in [21], but the performance gain, as illustrated in [20], is mainly from the summation of the log posteriors by the ferns compared to the summation of the posteriors by the forests. That is, in their application, the geometric mean of the posteriors are better than the arithmetic mean of the posteriors. The geometric mean is particularly good at suppressing false positives, a property that can be quite effective in rare event detection with biased risk, such as key points recognition and face detection. However, reducing false negatives is equally important as reducing false positives in classification tasks, such as bilayer video segmentations. Besides, the impact of label noise is not negligible. Therefore, we believe that random forests are more robust in our video layer segmentation task because they take the arithmetic means of posteriors.

8 Conclusions and Future Work

We present an algorithm for the efficient segmentation of the foreground and the background in monocular video sequences. Our algorithm is capable of inferring bilayer segmentation monocularly even in the presence of distracting background motion without the need for manual initialization.

This research has accomplished the following: 1) introduced novel visual features that capture motion and motion context efficiently, 2) provided a general understanding of tree-based classifiers, and 3) determined an efficient and accurate classifier in the form of random forests. Experiments and the related analysis of existing test data and our own database confirm accurate and robust layer segmentation in videochat sequences. Similarly to a stereo-based system, our approach manages to separate the foreground from the background even when distracting background motion occurs.

The segmentation results can also be used in the head tracking application shown in Fig. 18. Given bilayer segmentation, we simply extract the top point of the foreground pixels and use it as a reference point for our

Fig. 18. A head tracking application. A simple head tracking device may be built on top of our segmentation algorithm. In this monocular test sequence, both side views and frontal views are tracked successfully despite background motion.

framing window (lighter in Fig. 18). This result can be used for the "smart framing" of videos. On the one hand, this simple head tracker is efficient. It works for both frontal and side views, and it is not affected by background motion. On the other hand, it is heavily tailored toward the videochat application.

The tree-cube taxonomy has facilitated our theoretical analysis in Sections 7.3 and 7.4 and enabled fair experimental comparisons of classifiers like booster of trees and random forests. The insights gained from the comparison assisted us in designing a new classifier, namely, an ensemble composed of boosters of trees that combines the spirit of bias-variance reduction, boosting, and randomization. It achieves the lowest unary classification error in our videochat application. Next, we would like to apply our classifier taxonomy to other domains and applications to assess the merits of different types of classification algorithms in various situations. More investigation on learning "the correct level of rigidity" can be done by comparing the learned segmentation model with a random background without subjects. Further comparisons between tree-based classifiers and kernel machines [26] (such as SVMs) are also necessary.

ACKNOWLEDGMENTS

The authors would like to thank Dr. Jianxin Wu for the helpful discussions.

REFERENCES

[1] Y. Amit and D. Geman, "Shape Quantization and Recognition with Randomized Trees," *Neural Computation,* vol. 9, no. 7, pp. 1545-1588, 1997.
[2] S. Baker, R. Szeliski, and P. Anandan, "A Layered Approach to Stereo Reconstruction," *Proc. IEEE CS Conf. Computer Vision and Pattern Recognition,* pp. 434-441, 1998.
[3] J. Bergen, P. Burt, R. Hingorani, and S. Peleg, "A Three-Frame Algorithm for Estimating Two-Component Image Motion," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 14, no. 9, pp. 886-896, Sept. 1992.
[4] C. Bishop, *Neural Networks for Pattern Recognition.* Oxford Univ. Press, 1995.
[5] Y. Boykov, O. Veksler, and R. Zabih, "Fast Approximate Energy Minimization via Graph Cuts," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 23, no. 11, pp. 1222-1239, Nov. 2001.
[6] L. Breiman, "Arcing Classifiers," *Annals of Statistics,* vol. 26, no. 3, pp. 801-824, 1998.
[7] L. Breiman, "Random Forests," Technical Report TR567, Univ. of California Berkeley, 1999.
[8] A. Criminisi, G. Cross, A. Blake, and V. Kolmogorov, "Bilayer Segmentation of Live Video," *Proc. IEEE CS Conf. Computer Vision and Pattern Recognition,* pp. 53-60, 2006.
[9] T. Deselaers, A. Criminisi, J. Winn, and A. Agarwal, "Incorporating On-Demand Stereo for Real Time Recognition," *Proc. IEEE CS Conf. Computer Vision and Pattern Recognition,* 2007.
[10] Y. Freund and R. Schapire, "A Decision Theoretic Generalization of On-Line Learning and Application to Boosting," *J. Computer and System Science,* vol. 55, no. 1, pp. 119-139, 1997.
[11] J. Friedman, T. Hastie, and R. Tibshirani, "Additive Logistic Regression: A Statistical View of Boosting," *Annals of Statistics,* vol. 38, pp. 337-374, 2000.
[12] N. Jojic and B. Frey, "Learning Flexible Sprites in Video Layers," *Proc. IEEE CS Conf. Computer Vision and Pattern Recognition,* pp. 199-206, 2001.
[13] V. Kolmogorov, A. Criminisi, A. Blake, G. Cross, and C. Rother, "Bi-Layer Segmentation of Binocular Stereo Video," *Proc. IEEE CS Conf. Computer Vision and Pattern Recognition,* pp. 407-414, 2005.
[14] E. Kong and T. Dietterich, "Error-Correcting Output Coding Corrects Bias and Variance," *Proc. 12th Int'l Conf. Machine Learning,* pp. 313-321, 1995.
[15] S. Kumar and M. Hebert, "Discriminative Random Fields: A Discriminative Framework for Contextual Interaction in Classification," *Proc. IEEE Int'l Conf. Computer Vision,* pp. 1150-1157, 2003.
[16] J. Lafferty, A. McCallum, and F. Pereira, "Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data," *Proc. 18th Int'l Conf. Machine Learning,* pp. 282-289, 2001.
[17] V. Lepetit, P. Lagger, and P. Fua, "Randomized Trees for Real-Time Keypoint Recognition," *Proc. IEEE CS Conf. Computer Vision and Pattern Recognition,* June 2005.
[18] T. Leung and J. Malik, "Representing and Recognizing the Visual Appearance of Materials Using Three-Dimensional Textons," *Int'l J. Computer Vision,* vol. 43, no. 1, pp. 29-44, 2001.
[19] R. Lienhart, A. Kuranov, and V. Pisarevsky, "Empirical Analysis of Detection Cascades of Boosted Classifiers for Rapid Object Detection," *Proc. DAGM, 25th Pattern Recognition Symp.,* pp. 297-304, 2003.
[20] M. Özuysal, M. Calonder, V. Lepetit, and P. Fua, "Fast Keypoint Recognition Using Random Ferns," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 32, no. 3, pp. 448-461, Mar. 2009.
[21] M. Özuysal, P. Fua, and V. Lepetit, "Fast Keypoint Recognition in Ten Lines of Code," *Proc. IEEE CS Conf. Computer Vision and Pattern Recognition,* 2007.
[22] F. Perronnin, G. Dance, C. Csurka, and M. Bressan, "Adapted Vocabularies for Generic Visual Categorization," *Proc. IEEE European Conf. Computer Vision,* 2006.
[23] J. Quinlan, "Bagging, Boosting, and C4.5," *Proc. Nat'l Conf. Artificial Intelligence,* pp. 725-730, 1996.
[24] L. Rabiner and B. Juang, *Fundamentals of Speech Recognition.* Prentice Hall, 1993.
[25] C. Rother, V. Kolmogorov, and A. Blake, "GrabCut: Interactive Foreground Extraction Using Iterated Graph Cuts," *ACM Trans. Graphics,* vol. 23, no. 3, pp. 309-314, 2004.
[26] B. Scholkopf, "Statistical Learning and Kernel Methods," Technical Report MSR-TR 2000-23, 2000.
[27] J. Shotton, J. Winn, C. Rother, and A. Criminisi, "Textonboost: Joint Appearance, Shape and Context Modeling for Multi-Class Object Recognition and Segmentation," *Proc. IEEE European Conf. Computer Vision,* 2006.
[28] J. Sun, W. Zhang, X. Tang, and H. Shum, "Background Cut," *Proc. IEEE European Conf. Computer Vision,* pp. 628-641, 2006.
[29] P.H.S. Torr, R. Szeliski, and P. Anandan, "An Integrated Bayesian Approach to Layer Extraction from Image Sequences," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 23, no. 3, pp. 297-303, Mar. 2001.
[30] A. Torralba, K. Murphy, and W. Freeman, "Sharing Features: Efficient Boosting Procedures for Multiclass Object Detection," *Proc. IEEE CS Conf. Computer Vision and Pattern Recognition,* pp. 762-769, 2004.

[31] V. Vapnik, *Statistical Learning Theory.* Wiley-Interscience, Sept. 1998.
[32] M. Varma and A. Zisserman, "A Statistical Approach to Texture Classification from Single Images," *Int'l J. Computer Vision,* vol. 62, nos. 1-2, pp. 61-81, 2005.
[33] P. Viola and M. Jones, "Robust Real-Time Object Detection," *Int'l J. Computer Vision,* vol. 57, no. 2, pp. 137-154, 2004.
[34] J.Y.A. Wang and E.H. Adelson, "Layered Representation for Motion Analysis," *Proc. IEEE CS Conf. Computer Vision and Pattern Recognition,* pp. 361-366, 1993.
[35] J.Y.A. Wang and E.H. Adelson, "Representing Moving Images with Layers," *IEEE Trans. Image Processing,* vol. 3, no. 5, pp. 625-638, Sept. 1994.
[36] J. Winn, A. Criminisi, and T. Minka, "Object Categorization by Learned Universal Visual Dictionary," *Proc. IEEE Int'l Conf. Computer Vision,* pp. 1800-1807, 2005.
[37] P. Yin, I. Essa, T. Starner, and J.M. Rehg, "Discriminative Feature Selection for Hidden Markov Models Using Segmental Boosting," *Proc. IEEE Int'l Conf. Acoustics, Speech and Signal Processing,* Mar. 2008.
[38] A. Yuille, "Deformable Templates for Face Recognition," *J. Cognitive Neuroscience,* vol. 3, no. 1 pp. 59-70, 1991.
[39] C. Zhang, P. Yin, Y. Rui, R. Cutler, P. Viola, X. Sun, N. Pinto, and Z. Zhang, "Boosting-Based Multimodal Speaker Detection for Distributed Meeting Videos," *IEEE Trans. Multimedia,* vol. 10, no. 8, pp. 1541-1552, Dec. 2008.

Pei Yin received the BS degree from Tsinghua University, Beijing, China, and the MS degree from the Georgia Institute of Technology, where he is currently working toward the PhD degree on the topic of segmental analysis for sequence classification. His research interests include machine learning, computer vision, and multisource information fusion. He was a recipient of the Dean's Fellowship Award by the College of Computing at the Georgia Institute of Technology. He is currently working at Microsoft Bing. He is a student member of the IEEE.

Antonio Criminisi received the degree in electronic engineering from Palermo University in 1996, and the PhD degree in computer vision from the University of Oxford in 2000. In 1990, he was nominated Alfiere del lavoro by the Italian President Francsco Cossiga. His PhD thesis won the "British Computing Society Distinguished Dissertation Award" in 2001. Since then, he has been working as a researcher for Microsoft Research in various areas, such as stereo vision, object recognition, history of art, and medical imaging.

John Winn received the doctorate degree from the University of Cambridge in 2003. He is currently a researcher at Microsoft Research Cambridge in the Machine Learning and Perception Group. His main research interests are machine learning, machine vision, especially object recognition, and computational biology. His paper on nonlinear Bayesian image modeling with Christopher Bishop was awarded the Best Paper Prize at ECCV 2000. His work in machine learning includes development of variational message passing and he also cocreated, with Tom Minka, the Infer.NET (http://research.microsoft.com/infernet) framework for automatic inference. He also coorganizes the Pascal Visual Object Classes challenge.

Irfan Essa received the MS degree in 1990 and the PhD degree in 1994. He joined the Georgia Institute of Technology (Georgia Tech) faculty in 1996, and from 1988 to 1996, he held a research faculty position at the Massachusetts Institute of Technology (Media Lab). He is a professor in the School of Interactive Computing of the College of Computing, and and adjunct professor in the School of Electrical and Computer Engineering, Georgia Tech, Atlanta. He works in the areas of computer vision, computer graphics, computational perception, robotics, and computer animation, with potential impact on video analysis and production (e.g., computational photography and video, image-based modeling and rendering, etc.) human-computer interaction, and artificial intelligence research. Specifically, he is interested in the analysis, interpretation, authoring, and synthesis (of video), with the goals of building aware environments, recognizing and modeling human activities and behaviors, and developing dynamic and generative representations of time-varying streams. He has published more than 120 scholarly articles in leading journals and conference venues on these topics. He has been awarded the US National Science Foundation (NSF) CAREER Award and, within Georgia Tech, he has won the College of Computing's Junior and Senior Research Faculty Awards, Outstanding Teacher Award, the Institute's Educational Innovation Award, and the Dean's Award. He is also a recipient of the GVU Center's 15 years of Impact Award. He is a senior member of the IEEE and the ACM.