Journal of Intelligent Learning Systems and Applications, **2011, 3, **-****
doi:10.4236/jilsa.2011.31001 Published Online February 2011 (http://www.SciRP.org/journal/jilsa)

Categorization and Reorientation of Images Based on Low Level Features

Rajen Bhatt[1], Gaurav Sharma[1], Abhinav Dhall[1], Naresh Kumar[1], Santanu Chaudhury[2]

[1]Samsung India Software Center, Noida, India; [2]Electrical Engineering Department, Indian Institution of Technology Delhi, New Delhi, India.
Email: rajen.bhatt@gmail.com, schaudhury@gmail.com

Received June 22nd, 2010; revised July 29th, 2010; accepted September 23rd, 2010

ABSTRACT

A hierarchical system to perform automatic categorization and reorientation of images using content analysis is presented. The proposed system first categorizes images to some a priori defined categories using rotation invariant features. At the second stage, it detects their correct orientation out of {0°, 90°, 180°, *and* 270°} *using category specific model. The system has been specially designed for embedded devices applications using only low level color and edge features. Machine learning algorithms optimized to suit the embedded implementation like support vector machines* (*SVMs*) *and scalable boosting have been used to develop classifiers for categorization and orientation detection. Results are presented on a collection of about* 7000 *consumer images collected from open resources. The proposed system finds it applications to various digital media products and brings pattern recognition solutions to the consumer electronics domain.*

Keywords: *Categorization, Digital Content Management, Feature Selection, Orientation Detection, Scalable Boosting, Support Vector Machines*

1. Introduction

Digital Content Management (DCM) has become an area of vital importance with growing multimedia information available to consumers through various mediums such as internet, digital cameras, camcorders, mobile phones, and home recorders. Digital still images are most important form of memories captured by various electronic means. With the advancements in digital cameras and in-camera imaging technologies, high resolution and fine quality images are increasing at the consumer end. Various occasions like family parties and gatherings, outbound holiday tours, adventure, and still photography hobby generates large number of still images. In this paper, we present a DCM solution for still digital images which can automatically categorize images into four broad categories and detect their orientations from {0°, 90°, 180°, and 270°} based on the content analysis. Categorization of images is an important step for enabling many tasks, e.g., indexing, searching, and retrieving images from large collections. Orientation detection is an important problem because often digital camera users rotate the camera before image capture and when such images are fetched to the computer they are rotated and not synchronized with the view geometry and human visual system. Automatic orientation detection adds convenience for users as it eliminates the need to manually correct the orientation of many rotated images.

The solution proposed by us is hierarchical in nature, *i.e.*, first categorization and then orientation detection. For the categorization of images we have considered four broad categories namely Mountains, Monuments/Building/Architecture, Water bodies, and Portraits. As the rotation of the input image is unknown to the system, we extract rotation invariant features for the categorization task. Then within each category, we detect the correct orientation of images by the method tuned to the content-features of that particular category.

To make the proposed approach suitable for embedded devices, we make use of simple and computationally inexpensive features based on color and edge information of images. To further speed up the later stages of classification and orientation detection, we use Support vector machines (SVM) for categorization and highly optimized scalable boosting for the orientation detection.

The present paper is organized as follows. In Section 2 some of the related work has been described. We describe the proposed approach for categorization and orientation detection in Section 3. Computational experiments have been reported in Section 4. Section 5 concludes the paper with target applications and directions for further research.

2. Related Work

Image categorization is an area of much recent research. However, most of the work uses high time complexity point detector scale invariant features (SIFT) [1]. SIFT features require high computational complexity and memory usage. A recent representation of images, bag of words [2], derived from text document analysis has been shown to be very good for categorization tasks [2-4]. However, one cannot make use of such complex features because of hardware limitations and hence one cannot perform point detection and description for potentially multiple points on a full image.

Orientation detection is also a well researched field. A Bayesian learning approach has been proposed in [5]. In [6,7], authors have used SVMs with content-based features for orientation detection. They have used spatial color moments (CM) and edge direction histograms (EDH) and trained eight parallel SVMs, two for each class with the two different features, *i.e.*, CM and EDH. They have used classifier combination using averaging and another SVM on top of the eight SVM outputs. Adaboost-based approach proposed in [8] also used CM and EDH features. They have trained an indoor versus outdoor classifier using similar Adaboost algorithm and then used the category specific orientation detection. However, we feel that classifying images into indoor and outdoor is very broad and does not cover conceptually classes used by consumers to sort the images. Novel features based on Quadrature Mirror Filters (QMF) were proposed in [9]. These features were used with a two stage hierarchy of nonlinear SVM. The first stage classified the image as portrait or landscape and the second stage further classified landscape as 0 or 180 degrees and the portrait as 90 or 270 degrees. Scalable boosting-based approach has been invented by Google researcher in [10]. In [11], a large scale performance evaluation for image reorientation problem using different features and color spaces have been proposed to identify most useful ones. SVM have been trained for different subsets of the features and their performance comparisons were presented. In [12], pre-classification into known categories have been presented before orientation detection. The four categories considered were close-ups and wide views of natural and artificial (man-made) things, giving a total of 4 categories. However, the categorization is not enough contextually and more specific categorization should be considered for the consumer applications.

3. The Proposed DCM Framework

The proposed DCM framework is first categorization and then category specific orientation detection as articulated in **Figure 1** with some example images. In the following sections we describe the features and machine learning approaches used for the stated problem.

3.1. Image Categorization

We categorize images into four classes namely Mountains, Monuments/Building/Architecture, Waterbodies, and Portraits. **Figure 2** shows examples of the images

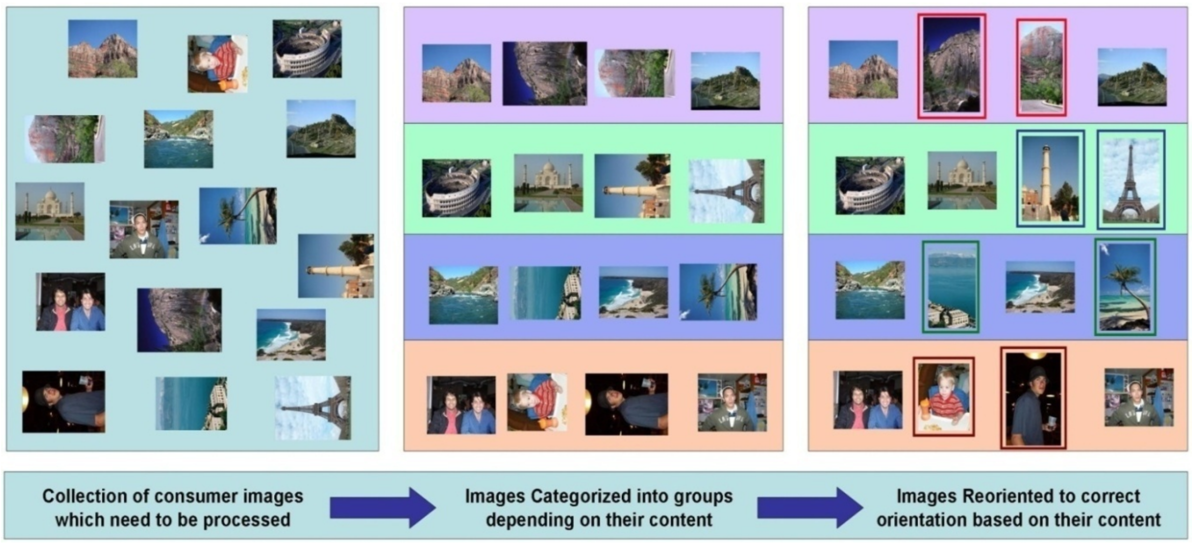
Collection of consumer images which need to be processed
Images Categorized into groups depending on their content
Images Reoriented to correct orientation based on their content

Figure 1. Categorization and orientation correction.

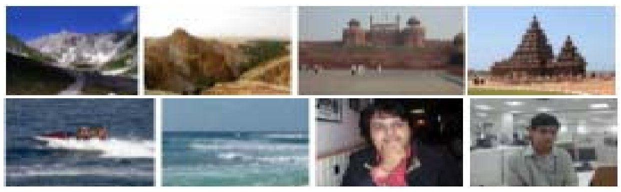

Figure 2. Example images from the four categories.

from the database. Portraits images are those images where human faces are prominent.

3.1.1. Low Level Color Correlogram Features

We solve the categorization problem with color correlogram features and SVM. For any pixel, the color correlogram gives the probability of a pixel at a distance k away to be of certain color. To be precise, pick any pixel p_1 of color C_i in the image I, at distance k away from p_1 pick another pixel p_2, the probability that p_2 is of certain color C_j can be defined by color correlogram feature. Color correlogram features are defined as [13]

$$\gamma^{k}_{C_i,C_j} = Pr\left\{\left|p_1 - p_2\right| = k, p_1 \in I_{C_i}, p_2 \in I_{C_j}\right\} \quad (1)$$

We use the dynamic programming algorithm given in [13] to compute the color correlogram feature for the given image.

First we compute the following two quantities:

$$\lambda^{c,h}_{(x,y)}(k) \triangleq \left|\left\{(x+i,y)\epsilon \mathrm{I_c} \mid 0 \le i \le k\right\}\right|$$

$$\lambda^{c,v}_{(x,y)}(k) \triangleq \left|\left\{(x,y+j)\epsilon \mathrm{I_c} \mid 0 \le j \le k\right\}\right| \quad (2)$$

Here λ are the counts of the number of pixels of a given color within a given distance from a fixed pixel in the positive horizontal and vertical directions. Then ignoring boundaries, the un-normalized correlogram is computed as

$$\Gamma^{(k)}_{c_i,c_j}(I) = \sum\nolimits_{(x,y)\in I_{Ci}} \Big(\lambda^{c_j,h}_{(x-k,y+k)}(2k) + \lambda^{c_j,h}_{(x-k,y-k)}(2k) + \lambda^{c_j,v}_{(x-k,y-k+1)}(2k-2) + \lambda^{c_j,v}_{(x+k,y-k+1)}(2k-2)\Big) \quad (3)$$

Then finally the correlogram is computed as

$$\lambda^{(k)}_{c_i,c_j}(I) = \Gamma^{(k)}_{c_i,c_j}(I) \Big/ \left(h_{c_i}(I) \times 8k\right) \quad (4)$$

The choice of color correlogram as a feature was motivated by two factors: (a) the input images may be rotated by multiples of 90 degrees and color correlogram is rotation invariant; and (b) images are expected to be color images and color distribution (global and spatially relative) is definitely a good discriminator for the different classes. While a histogram feature also captures color information from an image, it loses all the spatial information. Color correlogram feature captures the spatial (relative) correlation of colors while enabling efficient computation. It has been proposed and used for image indexing and retrieval [13]. We show that the feature along with a suitable classifier can be used for image categorization giving good results.

3.1.2. Color Codebook Formation

The color correlogram features are of dimension $d \times d \times c$ were d is the distance parameter and c is the number of colors in the image. To obtain a decent length of feature vector with good compromise between vector length and discrimination power, the color space has to be quantized and the image has to be downscaled in the color space. To quantize the color space, the simplest method is standard downscaling using uniform bin quantization. However, uniform bin quantization assumes a uniform distribution of colors in the database images. Since the images are not random but constrained to be consumer images, the distribution of the colors in the images is expected to be a complex multimodal distribution instead of a simple uniform distribution. To capture the modes of the distribution we opted to sample pixels from the database images and then to form the codebook for quantization. We use the pixel samples and perform K-means clustering to find the representative codebook vectors. These vectors are then used for downscaling the images by assigning each image pixel to the nearest codebook vector based on the Euclidean distance.

3.1.3. Support Vector Machines (SVM)

Once the images are represented with color correlogram vectors, we use support vector machines (SVM) [14] for categorization of the vectors into 4 classes. SVM are statistical classifiers which, given training instance-label pairs $(x_i, y_i), x_i \in R^n, y_i \in \{-1,+1\}$, solve the following optimization problem,

$$\min_{\omega,b,\xi} \frac{1}{2}\omega^T\omega + C\sum_{i=1}^{l}\xi_i$$

Subject to

$$y_i\left(\omega^T\phi(x_i) + b\right) \ge 1 - \zeta_{i,}\zeta_{i,} \ge 0 \quad (5)$$

SVM finds a separating hyper plane, in the φ(.) space which is induced high dimensional space, having the maximum margin [15] and has got proven generalization capability. **Figure 3** is the toy classification problem and separating hyper plane in 2D space. Samples on the margin are called support vectors.

We train one-*vs*-all SVM classifier on the training data. We have used the kernel SVM with radial basis function as a kernel. RBF kernels are described as

$$K\left(x_i, x_j\right) = \phi(x_i)^T \phi(x_i) = exp\left(-\gamma\left\|x_i - x_j\right\|^2\right), \gamma > 0. \quad (6)$$

The RBF kernel is one of the most important and popular kernel functions. It adds a bump around each data

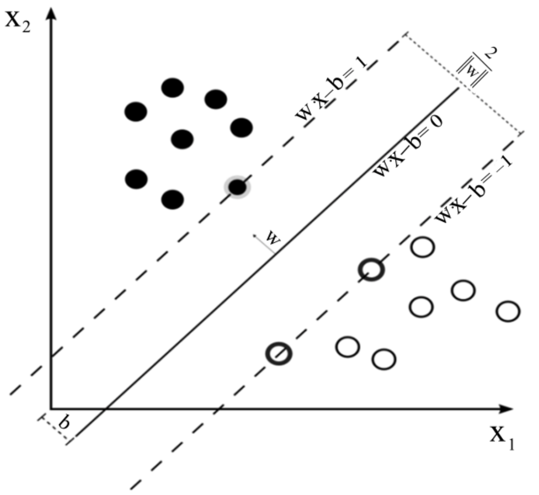

Figure 3. Example problem solved with linear SVM.

point:

$$f(x)=\sum_{i=1}^{m}\alpha_i \times exp\left(-\gamma\|x-x_i\|^2\right)+b \tag{7}$$

Using the RBF kernel each data point is mapped as articulated in the **Figure 4** below:

Example classification boundary generated by the RBF kernel classifier is shown in **Figure 5** below.

The cost parameter C and the kernel parameter γ were optimized using cross validation. We have used libSVM [15] for the target implementation.

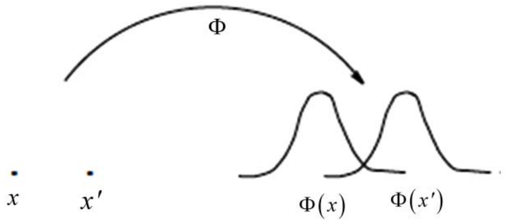

Figure 4. Mapping data points using the RBF kernel function.

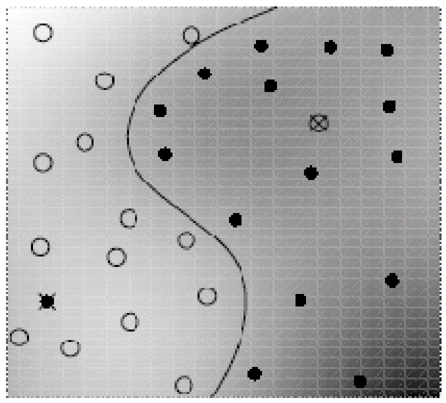

Figure 5. Example classification boundary generated by RBF kernel SVM.

3.1.4. Image Categorization Algorithm

Given a new image the categorization system works as follows. First, the image is rescaled to a smaller size so that the computation time can be saved. We have performed various experimentations by downscaling images to 640*480 and 320*240 resolution.

However, with 320*240 downscaling we have found that there is only marginal drop in the classification accuracy and that can be covered by tuning the SVM training parameters. While there is almost four times drop in the computation time and complexity of the feature extraction algorithm. Considering these computations, we have downscaled every image to 320*240 resolution during training and testing time. Then the image is also downscaled in color dimension to lower number of colors using the codebook of colors formed by *K*-means clustering of database pixel samples explained above in II.A.2. Once the image is downscaled to smaller number of colors, an indexed image is formed containing the pixel-to-codebook color index information. The color correlogram is then computed on this indexed image. The color correlogram is rotation invariant, *i.e.*, it is the same for any of the four orientation of the image on which it is being calculated. The correlogram is then used as input to the SVM model trained offline using the training dataset. The SVM returns the label of the category to which the image belongs. The category corresponding to the label is then given as the output of the system. **Algorithm 1** below gives the steps of the computation.

3.2. Image Orientation Detection

Once the images are categorized into four categories, we proceed to detect the correct orientation based on their content analysis. As our target is embedded and fast run time algorithm is required, we again work with simple low level color and edge features. For the classification task, we have identified the simple classifier combination with Adaboost algorithm.

3.2.1. Feature Extraction

The orientation detection algorithm used is based on the scalable boosting [10]. To suit the implementation over embedded devices and to meet the challenge of time and accuracies, we have optimized the scalable boosting approach. The vanilla version of scalable boosting uses the following feature extraction scheme.

From the original image, 15 simple transformed single channel images are computed as shown in **Figure 6**:

- 1-3: R,G,B, channels
- 4-6: Y,I,Q (transformation of R,G,B image) channels
- 7-9: Normalized versions of R,G,B channels (linearly scaled to span 0-255)

Algorithm 1: Rotation invariant categorization of input consumer images.

Input:	*Image with unknown orientation, color codebook built offline, SVM models trained offline*
Output:	*The category of the image, e.g., Mountains, Monuments/Building/Architecture, Water bodies, or Portraits, based on the content*

```
1) Downscale the input image in color dimension with respect to the codebook learnt offline. This will give an indexed image $I_{IND}$, whose each pixel will contain the index of the corresponding codebook vector.
2) Form the correlogram using the indexed image. We have used a codebook of size 12 and we have set the distance parameter $d = 4$ in our model. This gives us a vector of size $12 \times 12 \times 4 = 576$ dimensions.
3) Use the SVM model to obtain the label of the category to which the vector belongs to. Output the category corresponding to the label returned by the SVM.
```

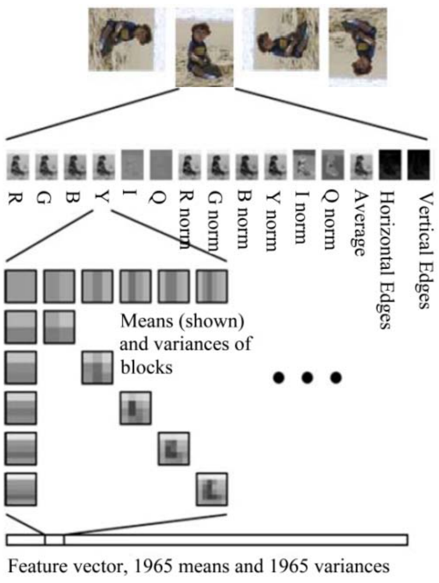
R
G
B
Y
I
Q
R norm
G norm
B norm
Y norm
I norm
Q norm
Average
Horizontal Edges
Vertical Edges
Means (shown) and variances of blocks
Feature vector, 1965 means and 1965 variances

Figure 6. Features calculated for scalable boosting algorithm.

- 10-12: Normalized versions of Y,I,Q channels (linearly scaled to span 0-255)
- 13: Intensity (simple average of R,G,B)
- 14-15: Horizontal and Vertical edge images, respectively, computed from intensity (*i.e.*, gray level) images

For each of these transformed images, the mean and variance of the entire image have been computed. Additionally, mean and variance of square sub-regions of the image have also been computed. The sub regions cover $\left(\frac{1}{2}\times\frac{1}{2}\right)$ to $\left(\frac{1}{6}\times\frac{1}{6}\right)$ of the image which gives total of 1 + 4 + 9 + 16 + 25 + 36 = 91 squares. Also the mean and variance of vertical and horizontal slices of the image that cover 1/2 to 1/6 of the image have been computed. There are total 2 + 3 + 4 + 5 + 6 = 20 horizontal and 20 vertical slices. Thus there are total of 1965 features $\left[15\times(91+20+20)\right]$ for mean and 1965 features for variance gives total features 3930.

However, calculating 3930 dimensional feature vector over 15 channels is computationally heavy for the embedded devices. We have optimized at this stage by first computing integral image over each channel [16]. An integral image is calculated by summed area table algorithm which quickly and efficiently generates the sum of values in a rectangular subset of a grid. The integral image at location *x*,*y* contains the sum of the pixels above and to the left of *x*,*y* with *x*,*y* inclusive (**Figure 7**), *i.e.*,

$$InI(x,y) = \sum_{x' \le x, y' \le y} I(x',y') \quad (8)$$

where $InI(x,y)$ is the integral image and $I(x',y')$ is the original image.

Using the following pair of recurrences:

$$\begin{aligned} s(x,y) &= s(x,y-1) + I(x,y) \\ InI(x,y) &= InI(x-1,y) + s(x,y) \end{aligned} \quad (9)$$

where $s(x,y)$ is the cumulative row sum, $s(x,-1)=0$ and $InI(-1,y) = 0$, an integral image can be computed. The integral image can be computed in one pass over the original image. From the database images (almost 7000 in number and scaled down to 640*480 for orientation detection) during training time with scalable boosting the feature extraction time was 36 hours. Using the integral image, this time has been reduced to 3 hours; 12 times reduction compared to the vanilla scalable boosting algorithm.

Another scope for optimization was found in moment (mean and variance) calculation when the images are rotated. Even though the image is rotated still the value of the block remains the same. Hence a mapping was done for a block at location (*x*,*y*) to its corresponding position in the rotated image. This saved in recalculation of block mean and variance value of the rotated integral

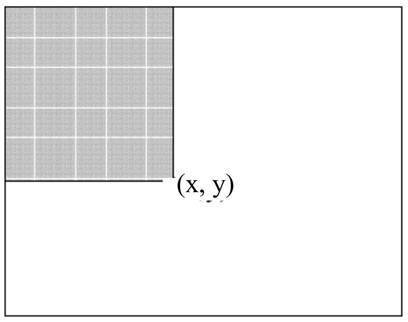

Figure 7. The value of the integral image at point (x, y).

image. This has further helped to reduce the feature extraction time during training and testing.

3.2.2. Scalable Boosting Classifier

Scalable boosting makes use of simple weak classifiers strategically combined with Adaboost algorithm to craft the final strong classifier. It uses the output of a simple binary comparison between two of the features described in the previous sub section as the weak classifier. Two simple Boolean operators (and their inverses) are used:

1) $\text{feature}_i > \text{feature}_j$

2) Difference of features within 25% of feature_i:

$$\left| feature_i - feature_j \right| \big/ feature_i < 0.25$$

Vanilla scalable boosting use the difference of features within 5% as well, however, we have found that by performing various optimizations, difference of features within 25% difference features are sufficient to achieve the target accuracies with reduction in the time complexity. Thus compared to vanilla scalable boosting which has 46334700 (almost 46 Million, $3930 \times 3930 \times 3$) features, our implementation has 30889800 (more than 30 Million, $3930 \times 3930 \times 2$) features. Still this is extremely large number of weak classifiers to consider.

The main steps of the Adaboost algorithm are shown in **Figure 8** [17]. At every iteration, Adaboost selects the best weak classifier for the weighted errors of the previous step. The weight changes in Step 4 are such that the weak classifier picked in Step 3 would have an error of 0.5 on the newly weighted samples, so it will not be picked again at the next iteration. Once all the weak classifiers are selected, they are combined to form a strong classifier by a weighted sum, where the weights are related to the re-weighting factors that were applied in Step 4.

Computing the accuracies of all the weak classifiers at each iteration is time-consuming, though it affects only the training time and not the runtime, computing all the features will be very prohibitive. To reduce the training time, we have randomly selected which weak classifiers will be evaluated at each iteration, and select the new classifier from only those that were evaluated.

At each iteration, the set of classifiers to be evaluated is randomly chosen again. Because of practical resource considerations during training, we have limited the number of weak classifiers examined in each iteration. We have experimented with 10000 to 50000 features at each iterations with the step size of 5000. This has varied training time for each AdaBoost classifier to approximately 1-3 days. In the final system, to meet the time-accuracy trade-off, we have adopted to use the strong classifiers generated by 30000 classifiers evaluation at each iteration of AdaBoost. Along with this setting of scalable boosting classifier, we have taken optimization steps described in the sub section III.B.1 above.

3.2.3. Image Orientation Detection Algorithm

We have optimized the scalable boosting algorithm as described in the above sections and weights for the combination of weak classifiers are learnt offline. Thus we have the weak classifiers and their weights learnt from offline training of the model. With a new image coming in, we use the features extracted to compute the response of the weak classifiers and combine the weighted responses to form the final response. This response is obtained for all 4 rotations of the image. The rotation with the maximum response is taken to be the correct rotation of the image and is given as the output to the user.

Algorithm 2 enumerates the steps for orientation detection using Scalable Boosting.

```
Input: samples $(x_1, y_1)...(x_N, y_N)$ where $x_i$ are the images and $y_i = 1$ for upright and 0 otherwise. N total training images.

Initialize weights:  for all upright samples i ($y_i = 1$):       $w_{1,i} = 0.5/|U|$
                     for all non-upright samples i ($y_i = 0$):   $w_{1,i} = 0.5/(N - |U|)$   where U is the set of upright images.

For $t = 1, \cdots, T$ (maximum number of weak classifiers to use):
    1. Normalize weights of the samples $w_{t,i}$ such that $\sum_i w_{t,i} = 1.0$
    2. Randomly select a set of weak classifiers to evaluate. For each weak classifier, $C_j$, measure the error with
          respect to the weight $w_t$: $\text{error}_t = \sum_i w_{t,i} |C_j(x_i) - y_i|$
    3. Choose the weak classifier (denoted $C_t$) with the lowest $\text{error}_t$.
    4. If example is classified incorrectly: $w_{t+1,i} = w_{t,i}$
         else: $w_{t+1,i} = w_{t,i} B_t$     where $B_t = \text{error}_t / (1 - \text{error}_t)$

For Classification:
The result of the strong classifier is: $S(x): \begin{cases} 1: \sum_{t=1}^{T} \log\left(\frac{1}{B_t}\right) * C_t(x) \geq 0.5 \sum_{t=1}^{T} \log\left(\frac{1}{B_t}\right) \\ 0: \text{otherwise} \end{cases}$
```

Figure 8. A boosting algorithm with {0, 1} targets.

Algorithm 2: Orientation detection of images based on content using the proposed optimized Scalable Boosting.

Input:	*Input image with unknown orientation, weak classifiers learnt offline using optimized scalable boosting*
Output:	*The correct orientation of the image based on the content*

```
1) Obtain the following images: R,G,B, R normalized, G normalized, B normalized, Y,I,Q, Y normalized, I normalized, Q normalized, Horizontal edge image and Vertical edge image (using the Sobel operator).
2) Extract moments (mean and variance) on blocks and strips of image regions from these 15 images and use them as the features. Figure 4 illustrates the feature extraction process.
3) Compute the response as the weighted combination of responses of the weak classifiers.
4) Do the steps 1-3 for all 4 rotations of the input image and thus obtain 4 response values from weighted combinations of weak classifier responses. Output the rotation which gives the maximum response as the correct response.
```

4. Computational Experiments

The proposed system has been tested on a database of over 7000 consumer images collected from the internet and personal image collections by an independent testing team. For the purpose of calculating the error rates and system testing we have divided database into ten disjoint sets with uniform distribution of classes and then performed tenfold cross validation. In the tenfold cross validation, each set is once used for the testing and training has to be performed by the remaining nine sets. This gives opportunity to each set for appearing in the training set for nine times and in testing set for once. We have chosen tenfold cross validation as it is most widely used and reported in the various statistical analyses of models [18]. The reported accuracies are average over tenfold runs.

All the 7000 images are ground truth edited, *i.e.*, categorized in the appropriate categories by an independent testing team. For the orientation detection experiments, we have rotated each of the images into four orientations, thus our orientation detection database is of the size of almost 28000 images.

For all the experiments we have used Intel Pentium IV with 3.2 GHz processor and 1 GB of RAM. We have used C++ programming architecture over Ubuntu release 8.10, Linux kernel 2.6.27-7 with GNOME 2.24.1. All the algorithms have been finally ported on the target system with Fedora Linux and the CPU of 1 GHz Intel with 256 MB of RAM.

4.1. Categorization Results

As described in Algorithm 1 and Section 3.1, for the categorization experiments we have color correlogram features with SVM classifier. We have trained SVM (one-vs-all framework) with normalized color correlogram features extracted from images in the training dataset randomly rotated to any of the four directions {0^o, 90^o, 180^o, 270^o}. The correlogram was normalized such that sum of the features were 100. The parameters of the SVM were optimized by cross validation during training. Parameter optimization of SVM has taken around 24 hours and then with the optimized parameters SVM training took around 2 hours. The dataset used for the cross validation to optimize the SVM parameters is the training set only; no testing sets were used for the parameter optimization. To claim that the proposed image categorization algorithm is independent of the orientation, we present the testing results with all four orientations, *i.e.*, after training we have rotated testing images to all the four orientations and checked the performance. Table I below gives the confusion matrices for the categorization task with all four orientations. In **Table 1** predicted classes are arranged in a row.

The overall performance of the method is 92.80% for the images rotated by 0^o, 92.40% for the images rotated by 90^o, 93.24% for the images rotated by 180^o, and 92.71% for the images rotated by 270^o. Computational experiments show that the difference in accuracies with all the four orientations is less than 1%. The overall performance of the proposed categorization system irrespective of the orientation of images is 92.78%.

The performance of Water bodies class is distributed over those images where water bodies have been found with mountains and monuments photographs. For example, see **Figure 9** where both the images have ground truth Water bodies, however, they can be equally accepted as Mountains or Monuments/Building/Architecture. This improves our contextual categorization, *i.e.*, categorization by human perception.

The proposed categorization system module has taken almost 210 mSec for categorizing one image to its respective class. This includes image downscaling, codebook formation, color correlogram feature extraction, and final categorization by SVM.

4.2. Orientation Detection Results

As described in Section 3.2, we have adopted our proposed scalable boosting algorithm for orientation detection in each class. Parameters for the scalable boosting have been trained offline and ported to the system for the system testing. The parameters obtained from the scalable boosting are features to be used and their weights. Only features learned during the training phase have been extracted during runtime to obtain the best timing performance.

For the orientation detection experiments we have rotated all the images to four orientations giving total of

Table 1. Confusion matrices for the categorization.

Rotated by 0°	Mountains	Monuments	Water Bodies	Portraits
Mountains	93.648649	4.234234	1.936937	0.180180
Monuments	0.275482	99.559229	0.055096	0.110193
Water Bodies	4.714202	8.839128	85.9128	0.471420
Portraits	0.151515	7.575758	0.151515	92.121212
Rotated by 90°				
Mountains	90.045045	3.918919	5.765766	0.270270
Monuments	0.991736	98.181818	0.771350	0.055096
Water Bodies	7.012375	5.067767	87.684148	0.235710
Portraits	0.303030	5.984848	0	93.712121
Rotated by 180°				
Mountains	94.774775	3.693694	1.486486	0.045045
Monuments	0.385675	99.559229	0.055096	0
Water Bodies	5.715969	7.778433	85.798468	0.707130
Portraits	0.227273	6.742424	0.151515	92.878788
Rotated by 270°				
Mountains	94.909910	3.063063	1.936937	0.090090
Monuments	1.432507	98.016529	0.220386	0.330579
Water Bodies	11.019446	4.773129	83.794932	0.412493
Portraits	0.454545	5.151515	0.227273	94.166667

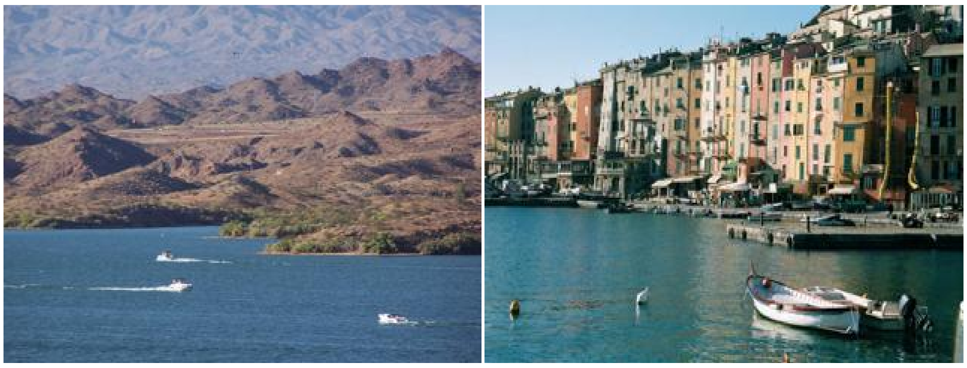
(a) (b)

Figure 9. Ambiguous images; (a) an image which can be accepted as either Water Bodies or Mountains and (b) an image which can be accepted as either Monument/Building/Architecture or Water Bodies.

40000 images (nearly 10000 images for each class). We have adopted ten fold cross validation here as well and results reported are average of all the runs. This is repeated ten times and the results are averaged. The overall performance of the proposed method for orientation detection is 96.01% for Mountain class, 92.07% for the Mounments/Building/Architecture class, 89.10% for Waterbodies class, and 91.28% for Portrait class. **Table 2** below gives the confusion matrices for the orientation detection task with all four orientations. Here predicted classes are arranged in a row.

The proposed orientation detection system module has taken almost 63 mSec for categorizing one image to its respective class. This includes orientation of image to four directions and obtaining scalable boosting estimation of their orientation.

4.3. Coupled System (Categorization and Orientation)

The final testing has been performed with categorization system coupled with orientation detection system. During this testing, we have passed all the testing images have been passed through the coupled system where they have first categorized and then operated by the category spe-

Table 2. Confusion matrices for the orientation detection.

Mountains	0	90	180	270
0	100	0	0	0
90	1.486486	98.153153	0	0.360360
180	8.873874	0.450450	90	0.675676
270	2.837838	1.261261	0	95.900901
Monuments				
0	100	0	0	0
90	7.382920	91.129477	0.275482	1.212121
180	9.201102	1.212121	88.209366	1.377410
270	9.035813	1.763085	0.220386	88.980716
Water Bodies				
0	99.941072	0	0	0.058928
90	4.007071	91.868002	0	4.124926
180	26.104891	0.117855	72.657631	1.119623
270	5.303477	2.710666	0	91.985857
Portraits				
0	91.590909	2.196970	4.166667	2.045455
90	2.424242	90.833333	2.727273	4.015152
180	3.484848	2.651515	90.984848	2.878788
270	2.272727	3.030303	2.954545	91.742424

cific scalable boosting to detect and correct their orientations. The system accuracy has been found to be 85.46% with average time taken for categorization and orientation detection of single image was almost 300 mSec. This time includes the correction of orientation of image if it is found to be other than 0^o.

We have adopted to tag these images by their respective category by adding the metadata information in the EXIF (Exchangeable Image File) format of JPEG images. Images with incorrect orientations have been redirected to the 0^o. From the second page onwards, start from the top of the page in a single column format. The JILSA header and authors' information from page one are not displayed. Do not insert include any header, footer or page number. JILSA editors will include the page number, authors' initials and the article number.

5. Conclusions and Further Research

In this paper, we have proposed the digital content management solution which is computationally less complex to result high time performance over embedded system and having better accuracy for categorization and orientation detection for general consumer images. The present system achieves 93.24% for rotation invariant categorization for 4 classes. When compared to scene categorization method of [19], we achieve a performance of 93.24% on 4 categories, while they achieve 89% on 4 categories (forest, mountains, country, and coast). In addition to the higher accuracy, the categories chosen by us are more general from consumer perspective and we make use of simple low level features for reducing time complexity. For example, Country category considered by [19] is of no use in consumer scenarios and they have not considered the Portrait class which is the most common class of consumer images.

Compared to the best performance of image orientation detection 79.1% reported in [14], we achieve 96.01% for orientation detection for Mountain class, 92.07% for Monuments class, 89.10% for Water bodies, and 91.28% for the portrait class. Further, database used in [11] was simple and taken from Corel collection containing scenic shots, tourist, and people shots. We are able to achieve best accuracies than [14] by first categorizing the images into more homogeneous categories and then training the algorithm independently on each category. The overall accuracy of the proposed system is 85.46%, which is higher than the one reported in [11].

The proposed system has been developed to run on embedded platform with limited computing resources. This restriction did not allow us to compute higher complexity SIFT features [1], and we have investigated some low level features combined with strong optimized machine learning algorithms to achieve the presented accuracies and timing performance.

The DCM (Digital Content Management) applications developed in addition to the proposed core algorithms for categorization and orientation detection form an important part of the product.

In general, we are proposing true DCM solution in which images are auto categorized first and then reoriented to reduce the burden of doing these operations manually or by inserting manual metadata in the EXIF files of JPEG encoded images. Various other features like color-based clustering, time and date-based clustering, similar scene detection, blur detection, and mapping image colors with music mood selection for album play along with the proposed categorization and orientation detection solutions bring pattern recognition-based consumer electronic products to the end users. To the best of our knowledge accuracies reported by us are best known in the literature and the proposed is the first of its kind product presenting a comprehensive DCM solution to the consumer electronics.

REFERENCES

[1] D. G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *International Journal on Computer Vision*, Vol. 60, No. 2, 2004, pp. 91-110. doi:10.1023/B:VISI.0000029664.99615.94

[2] J. Willamowski, D. Arregui, G. Csurka, C. R. Dance and L. Fan, "Categorizing Nine Visual Classes Using Local Appearance Descriptors," *International Conference on Pattern Recognition*, Cambridge, 2004.

[3] L. Fei-Fei and P. Perona, "A Bayesian Hierarchical Model for Learning Natural Scene Categories," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, San Diego, 20-25 June 2005, pp. 524-531.

[4] S. Lazebnik, C. Schmid and J. Ponce, "Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories," *Proceedings of Computer Vision and Pattern Recognition*, IEEE, New York, 2006, pp. 2169-2178.

[5] A. Vailaya, H.-J. Zhang, C. Yang, F.-I. Liu and A. K. Jain, "Automatic Image Orientation Detection," *IEEE Transactions on Image Processing*, Vol. 11, No. 7, 2002, pp. 746-755. doi:10.1109/TIP.2002.801590

[6] Y. Wang and H. Zhang, "Content-Based Image Orientation Detection with Support Vector Machines," *Proceedings of IEEE Workshop on Content-Based Access of Image and Video Libraries*, IEEE, New York, 2001, pp. 17-23. doi:10.1109/IVL.2001.990851

[7] Y. M. Wang and H.-J. Zhang, "Detecting Image Orientation Based on Low-Level Visual Content," *Computer Vision and Image Understanding*, Vol. 93, No. 3, 2004, pp. 328-346. doi:10.1016/j.cviu.2003.10.006

[8] E. Tolstaya, "Content-Based Image Orientation Recognition," *GraphiCon*, Moscow, 2007.

[9] S. Lyu, "Automatic Image Orientation Detection with Natural Image Statistics," *Proceedings of* 13th *International Multimedia Conference*, Singapore, 6-11 November 2005, pp. 491-494. doi:10.1145/1101149.1101259

[10] S. Baluja, "Automated Image-Orientation Detection: A Scalable Boosting Approach," *Pattern Analysis and Applications*, Vol. 10, No. 3, 2007, pp. 247-263. doi:10.1007/s10044-006-0059-1

[11] S. Baluja and H. A. Rowley, "Large Scale Performance Measurement of Content-Based Automated Image-Orientation Detection," *Proceedings of International Conference on Image Processing*, Vol. 2, 2005, pp. 514-517.

[12] H. L. Borgne and N. E. Oconnor, "Pre-Classification for Automatic Image Orientation," *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, Tou Louse, 14-19 May 2006, pp. 125-128. doi:10.1109/ICASSP.2006. 1660295

[13] J. Huang, S. R. Kumar, M. Mitra, W.-J. Zhu and R. Zabih, "Image Indexing Using Color Correlograms," *IEEE Computer Vision and Pattern Recognition*, San Juan, 1997, pp. 762-768.

[14] B. Scholkopf and A. J. Smola, "Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond," The MIT Press, Cambridge, 2001.

[15] C.-C. Chang and C.-J. Lin, "LIBSVM: A Library for Support Vector Machines," 2001. http://www.csie,ntu.edu.tw/~cjlin/libsvm

[16] F. Crow, "Summed Area Tables for Texture Mapping," *Proceedings of Special Interest Group in Graphics*, ACM, Minneapolis, 1984, pp. 207-212.

[17] Y. Freund and R. E. Schapire, "A Short Introduction to Boosting," *Journal of Japanese Society of Artificial Intelligence*, Vol. 14, No. 5, 1999, pp. 771-780.

[18] R. Kohavi, "A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection," *Proceedings of* 15*th International Joint Conference on Artificial Intelligence*, Nagoya, 23-29 August 1997, pp. 1137-1143.

[19] A. Oliva and A. Toralba, "Modeling the Shape of the Scene : A Holistic Representation of the Spatial Envelope," *International Journal on Computer Vision*, Vol. 42, No. 3, 2001, pp. 145-175. doi:10.1023/A:1011139631724