Automatic fruit and vegetable classification from images

Anderson Rocha*[,1], Daniel C. Hauagge[2], Jacques Wainer[1], Siome Goldenstein[1]

Abstract

Contemporary Vision and Pattern Recognition problems such as face recognition, fingerprinting identification, image categorization, and DNA sequencing often have an arbitrarily large number of classes and properties to consider. To deal with such complex problems using just one feature descriptor is a difficult task and feature fusion may become mandatory. Although normal feature fusion is quite effective for some problems, it can yield unexpected classification results when the different features are not properly normalized and preprocessed. Besides it has the drawback of increasing the dimensionality which might require more training data. To cope with these problems, this paper introduces a unified approach that can combine many *features* and *classifiers* that requires less training and is more adequate to some problems than a naïve method, where all features are simply concatenated and fed independently to each classification algorithm. Besides that, the presented technique is amenable to continuous learning, both when refining a learned model and also when adding new classes to be discriminated. The introduced fusion approach is validated using a multi-class fruit-and-vegetable categorization task in a semi-controlled environment, such as a distribution center or the supermarket cashier. The results show that the solution is able to reduce the classification error in up to 15 percentage points with respect to the baseline.

Key words: Feature and Classifier Fusion, Multi-class from Binary, Automatic Produce Classification, Image Classification

1. Introduction

Recognizing different kinds of vegetables and fruits is a recurrent task in supermarkets, where the cashier must be able to point out not only the species of a particular fruit (i.e., banana, apple, pear) but also its variety (i.e., Golden Delicious, Jonagold, Fuji), which will determine it's price. The use of barcodes has mostly ended this problem for packaged products but given that consumers want to pick their produce, they can not be packaged, and thus must be weighted. A common solution to this problem is issuing codes for each kind of fruit/vegetable; which has problems given that the memorization is hard, leading to errors in pricing.

As an aid to the cashier, many supermarkets issue a small book with pictures and codes; the problem with this solution is that flipping over the booklet is time-consuming.

This paper reviews several image descriptors in the literature and introduces a system to solve the problem by adapting a camera to the supermarket scale that identifies fruits and vegetables based on color, texture, and appearance cues.

Formally, given an image of fruits or vegetables of only one variety, in arbitrary position and number, the system must return a list of possible candidates of the form (species, variety). Sometimes, the object can be inside a plastic bag that can add specular reflections and hue shifts.

Given the variety and the impossibility of predicting which kinds of fruit/vegetables are sold, training must be done on-site by someone with little or no technical knowledge. Therefore, the system must be able to achieve a high level of precision with only a few training examples (e.g., up to 30 images).

Often, one needs to deal with complex classification problems. In such scenarios, using just one feature descriptor to capture the classes' separability might not be enough and feature fusion may become necessary. Although normal feature fusion is quite effective for some problems, it can yield unexpected classification results when the different features are not properly normalized and preprocessed. Besides it has the drawback of increasing the dimensionality of the data which might require more training examples.

This paper presents a unified approach that can combine many *features* and *classifiers*. It requires less training and is more adequate to some problems than a naïve method, where all features are simply concatenated and fed independently to each classification algorithm. We expect that this solution will endure beyond the problem solved in this paper.

The introduced fusion approach is validated using an image data set collected from the local fruits and vegetables distribution center and make it public. The image data set contains 15 produce categories comprising 2,633 images collected on-site in a period of five months under diverse conditions. The implemented solution achieves a classification error less than 2% for the top one responses. With the top two responses such error is smaller than 1%.

Section 2 gives a brief overview of previous work in object

*Corresponding author Dr. Anderson Rocha — Av. Albert Einstein, 1251, Cx. Postal 6176, CEP 13083-970, Campinas/SP, Brazil — Tel.: +55 19 8801 0992; fax: +55 19 3521 5888.

Email address: anderson.rocha@ic.unicamp.br (Anderson Rocha)

URL: http://www.ic.unicamp.br/~rocha (Anderson Rocha)

[1]Institute of Computing, University of Campinas (Unicamp), Campinas, Brazil.

[2]Department of Computer Science, Cornell University, Ithaca, United States.

recognition and image categorization. Section 3 presents the different kinds of image descriptors used in this paper as well as the produce data set. Section 4 introduces the solution for feature and classifier fusion, and Section 5 presents experimental results. Finally, Section 6 draws the conclusions and future directions.

2. Literature review

Recently, there has been a lot of activity in the area of *Image Categorization*. Previous approaches considered patterns in color, edge and texture properties (Stehling et al., 2002; Unser, 1986; Pass et al., 1997); low- and middle-level features to distinguish broad classes of images (Rocha and Goldenstein, 2007; Lyu and Farid, 2005; Cutzu et al., 2005; Serrano et al., 2004); In addition, Heidemann (2004) has presented an approach to establish image categories automatically using histograms, colors and shape descriptors with an unsupervised learning method.

With respect to the produce classification problem, *VeggieVision* (Bolle et al., 1996) was the first attempt of a supermarket produce recognition system. The system uses color, texture and density (thus requiring extra information from the scale). However, as this system was created sometime ago, it does not take advantage of recent developments. The reported accuracy was $\approx$ 95% in some scenarios but to achieve such result it uses the top four responses. The data set used in this paper is more demanding in some respects; while theirs had more classes, the image capturing hardware gave a more uniform color and suppressed specular lights. The data set assembled in this paper has greater illumination and color variation among images, also there is no measure to suppress specularities.

In general, the produce classification problem can be seen as a special instance of object's categorization. Turk and Pentland (1991) employed principal component analysis and measured the reconstruction error of projecting the image to a subspace and returning to the original image space. We believe this is ill suited for produce classification because it depends heavily on illumination, pose and shape.

Recently, Agarwal et al. (2004) and Jurie and Triggs (2005) adopted approaches that break down the categorization problem to the recognition of specific parts that are characteristic of each object class. These techniques, generally called bag-of-features (Marszalek and Schmid, 2006; Grauman and Darrel, 2005; Sivic et al., 2005), showed promising results even though they do not try to model spatial constraints among features.

Weber (2000) takes into account spatial constraints using a generative constellation model. The algorithm can cope with occlusion in a very elegant manner, albeit very costly (exponential in the number of parts). A further development made by Fei-Fei et al. (2006) introduced prior knowledge into the estimation of the distribution, thus reducing the number of training examples to around 10 images while preserving a good recognition rate. Even with this improvement, the problem of exponential growth with the number of parts persists, which makes it unpractical for the problem presented in this paper, which requires speed for on-line operation.

Another interesting technique was proposed by Berg et al. (2005). In that work, feature points are found in a gradient image. The points are connected by a joining path and a match is signalized if the found contour is similar enough to the one in the database. A serious drawback of this method for produce classification is that it requires a nonlinear optimization step to find the best contour; besides that it relies too heavily on the silhouette cues, which are not a very informative feature for fruits like oranges, lemons and melons.

3. Materials and methods

In general, image categorization relies on combinations of statistical, structural and spectral approaches. Statistical approaches describe the objects using global and local descriptors such as mean, variance, and entropy. Structural approaches represent the object's appearance using well-known primitives such as patches of important parts of the object. Finally, spectral approaches describe the objects using some spectral space representation such as Fourier spectrum (Gonzalez and Woods, 2007).

This paper analyzes statistical color and texture descriptors as well as structural appearance descriptors to categorize fruits and vegetables in a multi-class scenario. Since the best combination of features was not known for this problem, we analyze several state-of-the-art Computer Vision features in many different ways, and assemble a system with good overall accuracy using underpinned cross-validation procedures that allows the combination of the best features and classifiers into a single and unified approach.

The following sections present the statistical and structural descriptors used in this paper, as well as the data set assembled for the validation process.

3.1. Supermarket Produce data set

The *Supermarket Produce* data set is one of the contributions in this paper[3]. In general, there are a few well-documented image data sets available for image categorization and content-based image retrieval tasks for testing algorithm performance. ALOI[4] and Caltech[5] are two examples of such data sets for general categorization.

The *Supermarket Produce* data set is the result of five months of on-site collecting in the local fruits and vegetables distribution center.

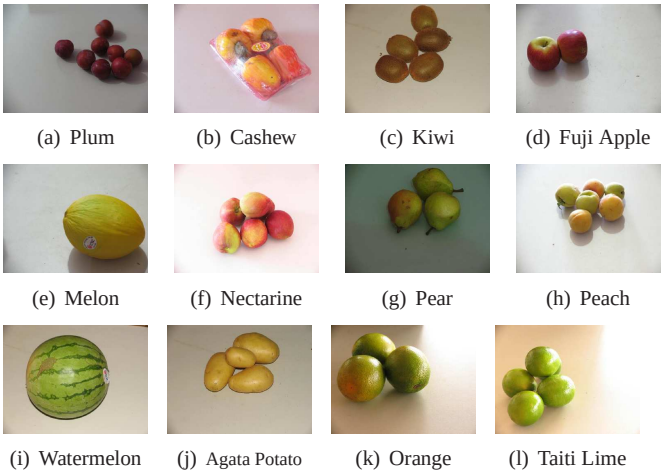
The images were captured on a clear background at the resolution of $1,024 \times 768$ pixels, using a Canon PowerShot P1 camera. For the experiments in this paper, they were downsampled to 640×480. The data set comprises 15 different categories: Plum (264), Agata Potato (201), Astexrix Potato (182), Cashew (210), Onion (75), Orange (103), Taiti Lime (106), Kiwi (171), Fuji Apple (212), Granny-Smith Apple (155),

[3] Freely available from `http://www.liv.ic.unicamp.br/~undersun/pub/communications.html`

[4] `http://staff.science.uva.nl/~aloi`

[5] `http://www.vision.caltech.edu/Image_Datasets/`

Watermelon (192), Honeydew Melon (145), Nectarine (247), Williams Pear (159), and Diamond Peach (211); totalizing 2,633 images. Figure 1 depicts some of the classes of the data set.



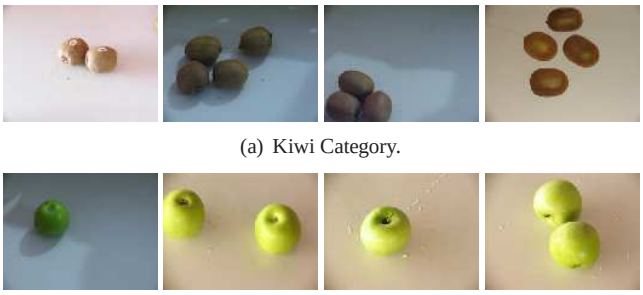
(a) Plum (b) Cashew (c) Kiwi (d) Fuji Apple

(e) Melon (f) Nectarine (g) Pear (h) Peach

(i) Watermelon (j) Agata Potato (k) Orange (l) Taiti Lime

Figure 1: *Supermarket Produce* data set.

All of the images were stored in RGB color-space at 8 bits per channel. The images were gathered at various times of the day and in different days for the same category. These features increase the data set variability and represent a more realistic scenario. Figure 2 shows an example of Kiwi and Granny-Smith Apple categories with different lighting. The differences are due to illumination, no image pre-processing was performed.



(a) Kiwi Category.

(b) Granny-Smith Apple Category.

Figure 2: Illumination differences within categories.

The *Supermarket Produce* data set also comprises differences in pose and in the number of elements within an image. Figure 3 shows examples of the Cashew category. Note that there are variations in the pose of the Cashew's plastic repository. In addition, Figure 4 shows the variability in the number of elements within an image.

Sometimes, the elements are inside a plastic bag which adds specular reflections to the analyzed image. Furthermore, the presence of shadows (e.g., second and third images of Figure 2(a)) and cropping/occlusions (e.g., Figure 5) makes the data set more realistic.

Figure 3: Pose differences. Cashew category.

Figure 4: Variability on the number of elements. Plum category.

3.2. Image descriptors

In this section, we describe statistical color, texture, and structural appearance descriptors (bag-of-features) in order to propose a system to solve a multi-class fruits/vegetables categorization problem.

3.2.1. Global Color Histogram (GCH)

The simplest approach to encode the information present in an image is the Global Color Histogram (GCH) (Gonzalez and Woods, 2007). A GCH is a set of ordered values, one for each distinct color, representing the probability of a pixel being of that color. Uniform quantization and normalization are used to reduce the number of distinct colors and to avoid scaling bias (Gonzalez and Woods, 2007). This paper uses a 64-d GCH feature vector, which means a histogram with 64 bins (features).

3.2.2. Unser's descriptors

Unser (1986) has shown that the sum and difference of two random variables with same variances are de-correlated and define the principal axes of their associated joint probability function. Hence, the author introduces sum s and difference d histograms as an alternative to the usual co-occurrence matrices for image texture description.

The non-normalized sum and difference associated with a relative displacement (δ_1, δ_2) for an image I, are defined as

$$s_{k,l} = I_{k,l} + I_{k+\delta_1,l+\delta_2}, \quad (1)$$

$$d_{k,l} = I_{k,l} - I_{k+\delta_1,l+\delta_2}. \quad (2)$$

The sum and difference histograms over the domain D are defined similarly to the spatial level co-occurrence or dependence matrix definition:

$$h_s(i;\delta_1,\delta_2) = h_s(i) = Card\{(k,l) \in D, s_{k,l} = i\}, \quad (3)$$

$$h_d(j;\delta_1,\delta_2) = h_d(j) = Card\{(k,l) \in D, d_{k,l} = j\}. \quad (4)$$

In addition to the histograms, as Table 1 shows, the method uses some associated global measures: mean (μ), contrast (C_n), homogeneity (H_g), energy (E_n), variance (σ^2), correlation (C_r), and entropy (H_n)) over the histograms.

This paper uses a 32-d Unser feature vector (histogram with 32 bins), calculated in the grayscale representation of the images.

Figure 5: Examples of cropping and partial occlusion.

Mean	$\mu = \frac{1}{2}\sum_i i h_s[i]$
Contrast	$C_n = \sum_j j^2 h_d[j]$
Homogeneity	$H_g = \frac{1}{1+j^2} h_d[j]$
Energy	$E_n = \sum_i h_s[i]^2 \sum_j h_d[j]^2$
Variance	$\sigma^2 = \frac{1}{2}\left(\sum_i (i-2\mu)^2 h_s[i] + \sum_j j^2 h_d[j]\right)$
Correlation	$C_r = \frac{1}{2}\left(\sum_i (i-2\mu)^2 h_s[i] - \sum_j j^2 h_d[j]\right)$
Entropy	$H_n = -\sum_i h_s[i]\log(h_s[i]) - \sum_j h_d[j]\log(h_d[j])$

Table 1: Histogram-associated global measures.

3.2.3. Color Coherence Vectors (CCVs)

Pass et al. (1997) presented an approach to compare images based on color coherence vectors. They define color coherence as the degree to which pixels of that color are members of a large region with homogeneous color. They refer to these significant regions as coherent regions. Coherent pixels are part of some sizable contiguous region, while incoherent pixels are not.

In order to compute the CCVs, the method blurs and discretizes the image's color-space to eliminate small variations between neighboring pixels. Then, it finds the connected components in the image in order to classify the pixels within a given color bucket as either coherent or incoherent.

After classifying the image pixels, CCV computes two color histograms: one for coherent pixels and another for incoherent pixels. The two histograms are stored as a single histogram. This paper uses a 64-d CCV feature vector, which means two 32-bin histograms.

3.2.4. Border/Interior (BIC)

Stehling et al. (2002) presented the border/interior pixel classification (BIC), a compact approach to describe images. BIC relies on the RGB color-space uniformly quantized in 4×4×4 = 64 colors. After the quantization, the image pixels are classified as *border* or *interior*. A pixel is classified as *interior* if its 4-neighbors (top, bottom, left, and right) have the same quantized color. Otherwise, it is classified as *border*.

After the image pixels are classified, two color histograms are computed: one for border pixels and another for interior pixels. The two histograms are stored as a single 128-dimension vector.

3.2.5. Appearance descriptors

Agarwal et al. (2004) and Jurie and Triggs (2005) have proposed to describe local appearance using a vocabulary of parts.

In such cases, images are converted to grayscale to locate interest points and patches are extracted from the gradient magnitude image or the original grayscale image. Lowe's feature point detector, for instance, can be used to find the coordinates of interest points, together with orientation and scale. Once found, a square region around the point is extracted. The square side is proportional to the scale and the orientation follows that of the feature point. Once extracted, all patches are resized to 13×13 pixels.

All patches in the training set are clustered using K-means. The cluster centers are used as a part dictionary. The found centroids can be seen on Figure 6.

This paper uses two different schemes for values of the components of the feature vectors. In the first one, the value for each component is equal to the distance between the dictionary part d_i and the closest patch p_j in the given image, as in the equation

$$f_i = \min_{\forall j} \frac{p_j \cdot d_i}{||p_j|| ||d_i||}. \tag{5}$$

In the second scheme, the value for the component is equal to 1 if this part is the closest one for some patch of the input image, and it is 0 otherwise

$$f_i = \begin{cases} 1 & \text{if } i = \arg\min_i \frac{p_j \cdot d_i}{||p_j|| ||d_i||} \text{ for some } j \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

When convenient, the name of the algorithm is suffixed with the size of the used feature vector. For instance, K-Means-98 refers to the use of K-Means algorithm on a code-book (feature space) of 98 dimensions.

The vocabulary of parts uses some images from the *Supermarket Produce* data set in the vocabulary creation stage. The images used for the vocabulary generation are excluded from the data set in the posterior training/testing tasks.

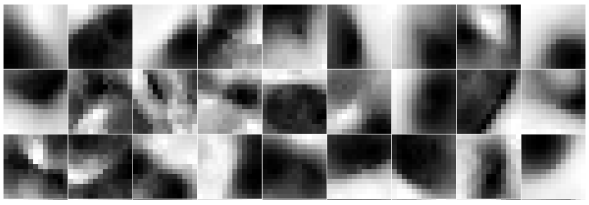


Figure 6: Dictionary of parts (partial), clustered using K-means.

3.3. Supervised learning techniques

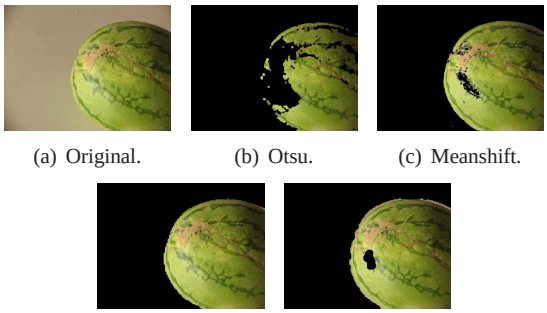
Supervised learning is a machine learning approach that aims to estimate a classification function f from a *training data set*. The commonest output of the function f is a label (class indicator) of the input object under analysis. The learning task is to predict the function outcome of any valid input object after having seen a sufficient number of training examples.

In the literature, there are many different approaches for supervised learning such as Linear Discriminant Analysis (LDA), Support Vector Machines (SVMs), Classification Trees, Neural Networks (NNs), and Ensembles of Classifiers (Bishop, 2006).

3.4. Background subtraction

For a real application in a supermarket, it might be necessary to cope with illumination variations, sensor capturing artifacts, specular reflections, background clutter, shading, and shadows. Therefore, in order to reduce the scene complexity, it might be interesting to perform background subtraction and focus in the object's description.

Figure 7 depicts results for some segmentation approaches considered in this paper. Otsu background algorithm (Otsu, 1979) is the fastest tested approach requiring only 0.3 seconds to segment an input image of 640×480 pixels. Meanshift (Comaniciu and Meer, 2002) provides a good image segmentation within 1.5 seconds in average but requires the correct tuning of parameters for different image classes. Normalized cuts (Shi and Malik, 2000) approach produces good results but, as it needs to calculate the eigenvectors of the image, it requires ≈ 5 seconds to perform the segmentation, even for a reduced image (128×96 pixels). Therefore, this paper also introduces a background extraction procedure, summarized in Algorithm 1, which produces acceptable results in less than a second and is illustrated in Figure 7(e).



(a) Original. (b) Otsu. (c) Meanshift.

(d) Norm. Cuts. (e) K-Means.

Figure 7: Background subtraction results for four different approaches.

Algorithm 1 Background subtraction based on K-Means

Require: Input image I stored in HSV;

1: $I_{down} \leftarrow$ Down-sample the image to 25% of its original size using simple linear interpolation.
2: Get the S channel of I_{down} and consider it as an 1-d vector V of pixel intensities.
3: Perform $D_{bin} \leftarrow$ K-Means(V, k=2)
4: Map $M \leftarrow D_{bin}$ back to image space. For that just do a linear scan of D_{bin}.
5: $M_{up} \leftarrow$ Up-sample the generated binary map M back to the input image size.
6: Close small holes on M_{up} using the *Closing* morphological operator with a disk structuring element of radius 7 pixels.

We have found that, for the produce classification problem, the best channel to perform the background subtraction is the S channel of HSV-stored images. This is understandable given that the S channel is much less sensitive to lighting variations than any of the RGB color channels (Gonzalez and Woods, 2007). All approaches discussed here were implemented in the HSV color channel. After segmentation, all the features are calculated within the object region defined by the masks.

4. Feature and classifier fusion

This section shows the motivation and design of the feature and classifier fusion introduced in this paper.

4.1. Motivation

When copping with complex multi-class categorization problems, such as the produce classification in this paper, it might be necessary to use several different features to capture the classes' variability and nuances. Therefore, efficient and effective feature fusion policies need to be deployed.

In spite of the fact that feature-level combination is not straightforward for multi-class problems, for binary problems this is a simple task. In such scenario, it is possible to combine different classifiers and features by using classic rules such as `and`, `or`, `max`, `sum`, or `min` (Bishop, 2006). For multi-class problems, this is more difficult given that one feature might point out to an outcome class C_i and another feature might result the outcome class C_j, and even another one could result C_k. With many different resulting classes for the same input example, it becomes difficult to define a consistent policy to combine the selected features.

One approach sometimes used is to combine the feature vectors for different features into a single and big feature vector. Although quite effective for some problems, this approach can also yield unexpected classification results when not properly preprocessed. First, in order to create the combined feature vector, one needs to tackle the different nature of each feature. Some can be well conditioned such as continuous and bounded variables, others can be ill-conditioned for this combination such as categorical ones. In addition, some variables can be continuous and unbounded. To put everything together, one needs a well-suited normalization. However, this normalization is not always possible or sometimes leads to undesirable properties in the new generated feature vector such as equally weighting all the feature coefficients, a property that in general it is not wanted.

When combining feature vectors this way, eventually it is necessary to cope with the curse of dimensionality. With the addition of new features, there is an increase in the number of dimensions which then might require more training data.

Finally, if one feature needs to be added, it is necessary to redesign the normalization in order to deal with all the aforementioned problems. The following section introduces a simple and effective solution for feature and classifier fusion that addresses most of the previously discussed concerns.

4.2. The new Feature and Classifier Fusion Technique

The objective here is to combine a set of features and the most appropriate classifier for each one in order to improve the overall classification accuracy. To avoid the inherent problems of proper normalization and curse of dimensionality, we do not

create a big feature vector combining the selected features. Furthermore, doing that we would only perform feature fusion and we would still be limited in doing the classifier fusion.

This paper tackles the multi-class problem as a set of binary problems. For that, we define a *class binarization* as a mapping of a multi-class problem onto two-class problems (divide-and-conquer) and the subsequent combination of their outcomes to derive the multi-class prediction. We refer to the binary classifiers as *base learners*. Class binarization has been used in the literature to extend naturally binary classifiers to multi-class and Support Vector Machine (SVM) is one example of this (Dietterich and Bakiri, 1996; Anand et al., 1995; Narasimhamurthy, 2005). To our knowledge, this approach was not used before for classifier and feature fusion.

In order to understand the class binarization, consider a toy problem with 3 classes. In this case, a simple binarization consists in training three base learners, each one for two classes. In this sense, we need $\binom{N}{2} = O(N^2)$ binary classifiers, where N is the number of classes.

The ij^{th} binary classifier uses the patterns of class i as positive and the patterns of class j as negative examples. To obtain the final outcome, it is enough to calculate the minimum distance of the generated vector (binary outcomes) to the binary pattern (ID) representing each class.

Consider again a toy example with three classes as depicted in Figure 8. This example contains the classes: *Triangles* $\triangle$, *Circles* $\bigcirc$, and *Squares* $\square$.

To solve this toy problem, we train some binary classifiers differentiating two classes at a time, such as $\triangle \times \bigcirc$, $\triangle \times \square$, and $\bigcirc \times \square$. Each class receives a unique identifier as Table 2 shows. In this table, each column represents the differentiation of two classes at a time. To populate the table is straightforward. First, we perform the binary comparison $\triangle \times \bigcirc$ and tag the class $\triangle$ with the outcome $+1$, the class $\bigcirc$ with -1 and set the remaining entries in that column to 0. Thereafter, we repeat the procedure comparing $\triangle \times \square$, tag the class $\triangle$ with $+1$, the class $\square$ with -1, and the remaining entries in that column with 0. We repeat this procedure for each combination of two classes. In the end, each row in Table 2 represents the code of that class (e.g., $\triangle = \langle +1, +1, 0 \rangle$), where the entry 0 means a "Don't care" value.

	$\triangle \times \bigcirc$	$\triangle \times \square$	$\bigcirc \times \square$
$\triangle$	$+1$	$+1$	0
$\bigcirc$	-1	0	$+1$
$\square$	0	-1	-1

Table 2: Class' IDs for the toy example in Figure 8.

As Figure 8 depicts, a feature that can be used to categorize elements of these classes is the shape. Texture and color properties can be used as well.

To classify an input example, for instance, a triangle-shaped one, we first apply the binary classifiers to verify if the input example is a triangle or a circle based on shape, texture and color features. Each classifier will result a binary response. Let's say we obtain the outcomes $\langle +1, +1, -1 \rangle$ for the binary classifier

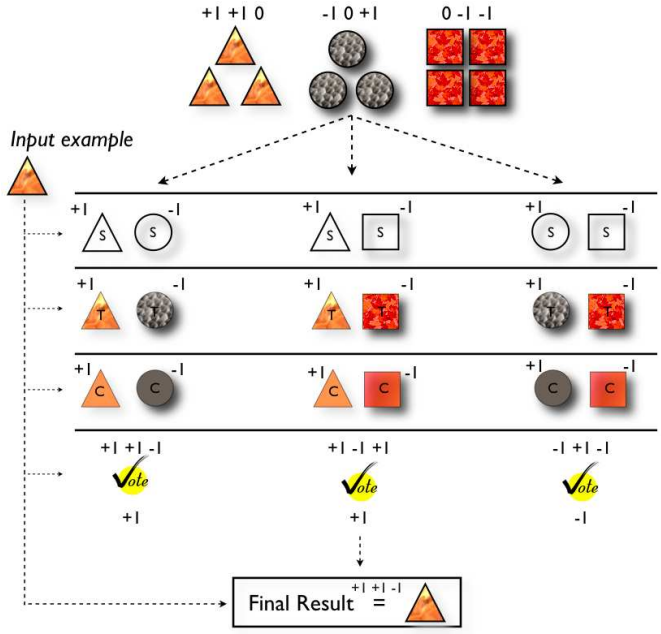


Figure 8: Toy example for feature and classifier fusion.

$\triangle \times \bigcirc$ considering shape, texture, and color features respectivelly. Majority voting can be used to select one response ($+1$ in this case, or $\triangle$). Then we repeat the procedure and test if the input example is a triangle or a square, again for each one of the considered features.

Finally, after performing the last test, we end up with a vector. Then we calculate the minimum distance from this vector to each one of the class unique IDs. In this example, the final answer is given by the minimum distance of

$$\min dist(\langle 1, 1, -1 \rangle, \{\langle 1, 1, 0 \rangle, \langle -1, 0, 1 \rangle, \langle 0, -1, -1 \rangle\}. \quad (7)$$

One aspect of this approach is that it requires more storage given that once the binary classifiers are trained their parameters need to be stored. With respect to running time, there is also a small increase given that more binary classifiers are needed to provide an outcome. However, many classifiers employ some sort of class binarization (e.g., SVM) and are considered fast. The majority voting for each binary classifier and the distance calculation to the unique class IDs are simple and efficient operations.

Although the approach requires more storage and increase the classification time with respect to a normal multi-class approach, it has some advantages:

1. The combination of independent features gives more confidence in the classification and it works as a simple error correcting mechanism that can withstand some misclassifications;
2. Well-tuned binary classifiers and features can be deployed to solve localized class confusions;
3. Less contributing features can be easily identified. This is not straightforward with normal binding in a big feature vector;

4. The addition of new classes and features only require training for the new binary classifiers and features;
5. As there is no increase in the size of any feature vector, the approach is less prone to the curse of dimensionality.

Finally, there is no requirement to combine the binary classifiers using all combinations of two classes at a time. One can reduce storage requirements and speed up the classification by selecting classes that are in confusion and designing specific binary classifiers to separate them (Rocha and Goldenstein, 2009). The expectation in this case is that fewer binary classifiers would be needed. Indeed, there is room for more research in this direction.

5. Results and discussions

In the quest for finding the best classification procedures and features for produce categorization, this paper analyzes several appearance-, color-, texture-, and shape-based image descriptors as well as diverse machine learning techniques such as Support Vector Machine (SVM), Linear Discriminant Analysis (LDA), Classification Trees, K-Nearest Neighbors (K-NN), and Ensembles of Trees and LDA (Bishop, 2006). All the experiments hereafter are made on real data (Section 3.1).

In the following experiments, we select the training images using sampling without replacement from the pool of each image class. If we are training with 10 images per class, we use the remaining ones for testing. We repeat this procedure 10 times, and report the average classification accuracy (μ), average error ($\epsilon = 1 - \mu$), and standard deviation (σ). We do not use the strictly 10-fold cross-validation, given that we are interested in different sizes of training sets. In each round, we report the accuracy for the 15 classes summing the accuracy of each class and dividing by the number of classes.

5.1. Results without fusion

Figures 9(a) and 9(b) show results for different classifiers and the features GCH and CCV. The *x*-axis represents the number of images per class in the training set and the *y*-axis represents the average accuracy in the testing set.

This experiment shows that Breiman's decision Tree does not perform very well. One possible explanation is that the descriptor data is not suitable for this kind of classifier. The ensemble of trees (BAGG), with 17 iterations performs better than simple decision Trees and it is more stable.

We also observe across the plots that LDA accuracy curve practically become flat for more than 24 examples. An ensemble of LDA (BLDA) performs random sampling across the training data and makes a better use of the provided information in such a way it can improve the classification. Therefore, the ensemble of LDA with 17 iterations performs better than straight LDA, Trees, or ensemble of Trees. Complementing, the simple K-Nearest Neighbors here is as good as ensemble of Trees.

A more complex approach such as the appearance descriptor for this particular problem does not yield a good classification accuracy, as Figure 9(c) depicts. We tested three different approaches for the appearance-based descriptor in Figure 9(c). Two interesting observations: (1) the approach based on patches with no gradient orientation is the only feature that does not benefit from more examples in the training; and (2) the approach based on patches with gradient orientation is the one which benefits more with more training examples. This suggests that, with enough training, it might provide much better results. Notwithstanding, the training data is limited as discussed in Sections 1 and 4.

Most likely, the appearance descriptor does not provide a significant classification accuracy because the used patches do not represent well all the images classes under analysis. Further investigation must be done in this direction to validate the use of appearance descriptor or any other similar model. Although, previous results in literature argue that it requires less examples for training, it is fact that it requires lots of good representative images to create effective appearance patches and accomplish such claims.

It is hard to solve a multi-class problem using just one feature descriptor. As the classifier results for different features can be quite different, it is possible that their combination, with each classifier custom-tailored for a particular feature, may boost the effectiveness. Section 5.2 shows such results.

5.2. Fusion results

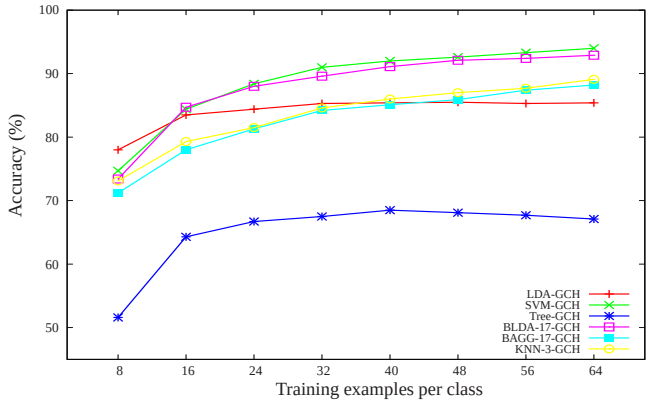
This section presents results for the fusion approach introduced in this paper (Section 4) and shows that the combined features and classifiers boost classification results when compared the standalone features and classifiers.

5.2.1. Top one response

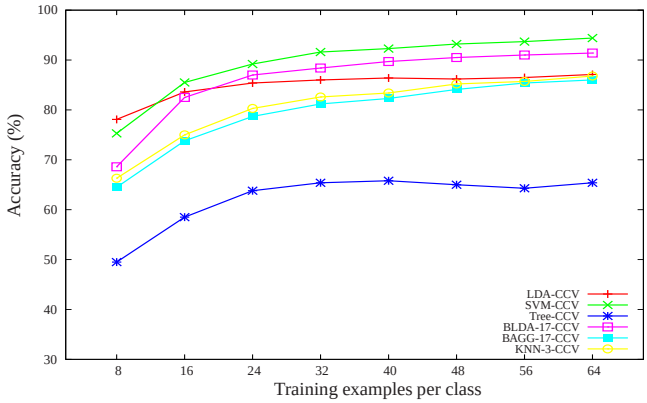
Figure 10 shows examples of the combination of the BIC, CCV, and Unser descriptors. This combination is interesting given that, BIC is a descriptor that analyzes color and shape in the sense that it codifies the object's border and interior, CCV codifies the color connected components and Unser accounts for the image's textures.

Figure 10 shows that the fusion works well regardless the classifier used. Consider the SVM classifier, with 32 examples per class in the training. In this case, the fusion results an average error of $\epsilon = 3.0\%$ and standard deviation of $\sigma = 0.43\%$. This is better than the best standalone feature, BIC, that is $\epsilon = 4.2\%$ and standard deviation of $\sigma = 0.32\%$. Although the absolute difference here seems small, it is about 3 standard deviations which means it is statistical significant. In general, to reduce one percentage point in the average error when the baseline accuracy is $\approx 95\%$ is a hard problem.

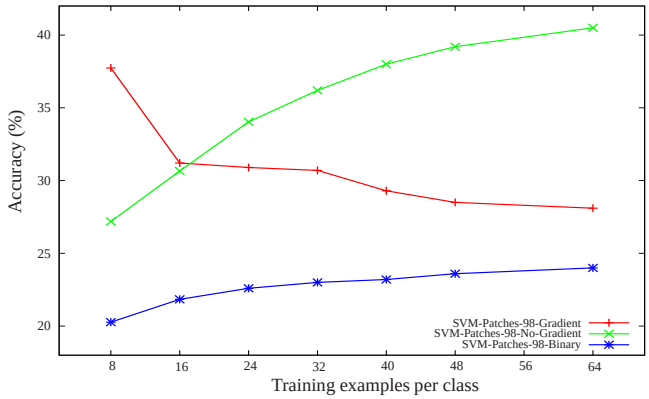
For the LDA classifier, the fusion requires at least 24 examples per class in the training to reduce the error. As observed in Section 5.1, LDA curves become flat with more than 24 examples per class in the training and adding more training data does yield better results. When combining different features, LDA does benefit from more training and indeed results lower error



(a) Feature: GCH



(b) Feature: CCV



(c) Feature: Appearance Patches

Figure 9: Average accuracy per class, WITHOUT fusion, considering diverse classifiers and features.

rates ($\epsilon = 3\%, \sigma = 0.59\%$), 9.8 standard deviations better than straight LDA on the BIC feature.

Figure 11 switches the BIC descriptor to a simpler one with half of the dimensionality (64-d). The results are comparable to the ones obtained with the fusion before. But now, the fusion shows even more power.

For 32 training examples and LDA classifier, for instance, the fusion reduces the classification error to about 5% while the best feature without any fusion results an error of 15%. With SVM classifier and 40 examples per class in the training, the fusion (using the features GCH, Unser, and CCV) yields an error $\epsilon \approx 5\%$ while the best feature with no fusion produces an error of $\epsilon \approx 9\%$.

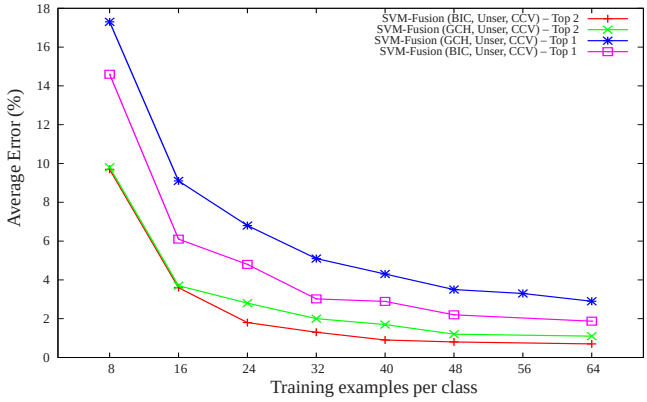


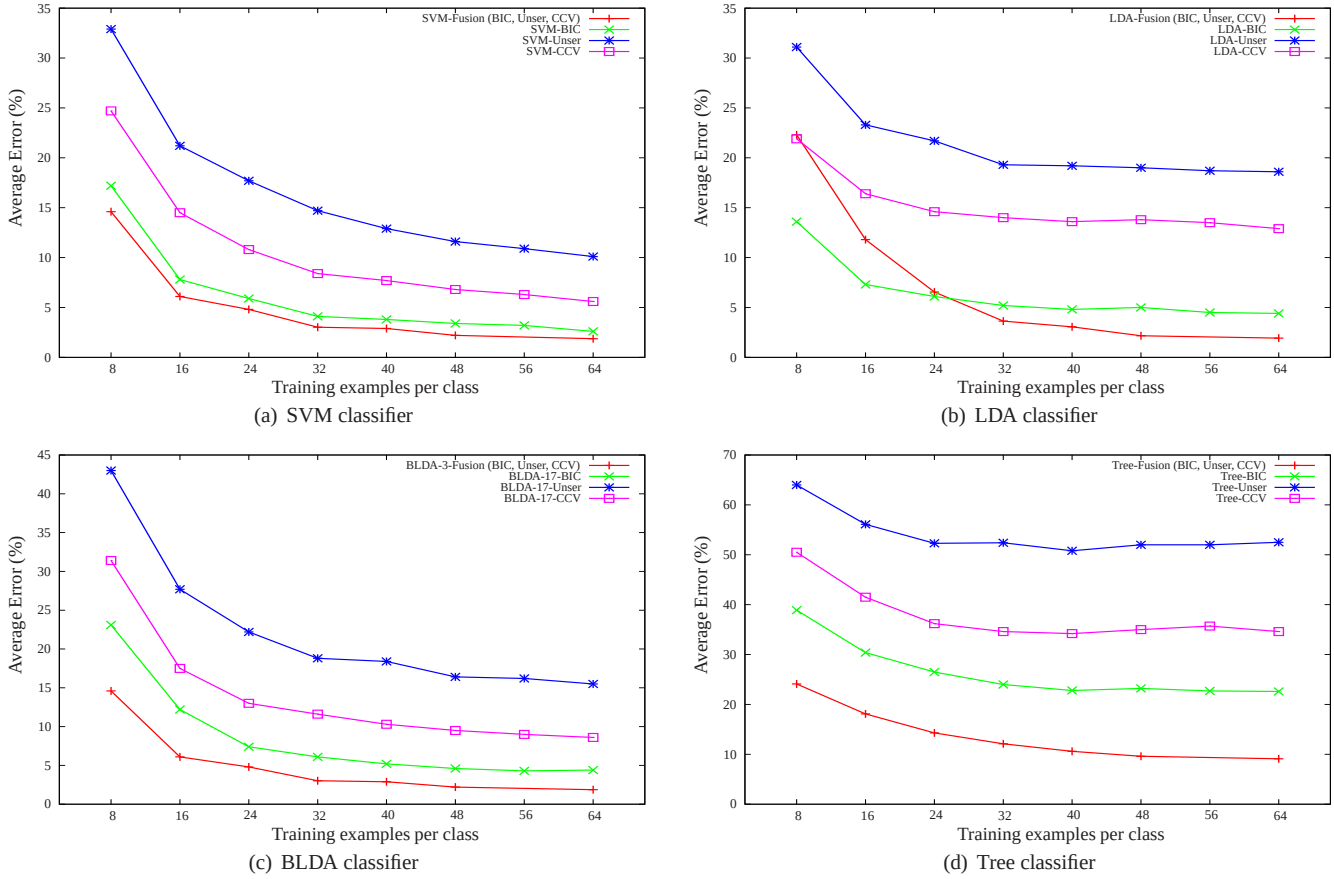
Figure 12: Top one, and two responses for SVM classifier with fusion.

5.2.2. Top two responses

Figure 12 portrays the results when the system is required to show the top 2 responses. In this case, the system provides the user the two most probable classes for a given input example considering the different classifiers and features used. Using SVM classifier and fusion of BIC, GCH, and Unser features, with 32 examples per class in the training, the average error is $\epsilon \leq 1\%$.

5.2.3. Average error per class

One important aspect when dealing with classification is the average expected accuracy per class. This information points out the classes that need more attention when solving the confusions. Figure 13 depicts the average expected error for each one of 15 classes. Clearly, *Fuji apple* is one class that needs particular attention. It yields the highest error when compared to the other classes. Another class that has an interesting er-



(a) SVM classifier (b) LDA classifier

(c) BLDA classifier (d) Tree classifier

Figure 10: Average error results for fusion of BIC, CCV, and Unser features considering SVM, LDA, BLDA, and Tree base learners.

ror behavior is *Onions*. After the error decreases when using up to 40 training examples it becomes higher as the number of training examples increases.

5.2.4. Average time

The average time to extract any of the features and perform classification using the introduced fusion technique is less than a second. However, the more examples in the training set the more time consuming are the combinations in the training stage. For instance, to train a multi-class classifier using the fusion approach introduced in this paper, with SVM base learner, 48 training examples per class, and the combination of the features BIC, CCV, and Unser, it is necessary about one hour in one 2.1GHz machine with 2GB of RAM.

6. Conclusions and Future Work

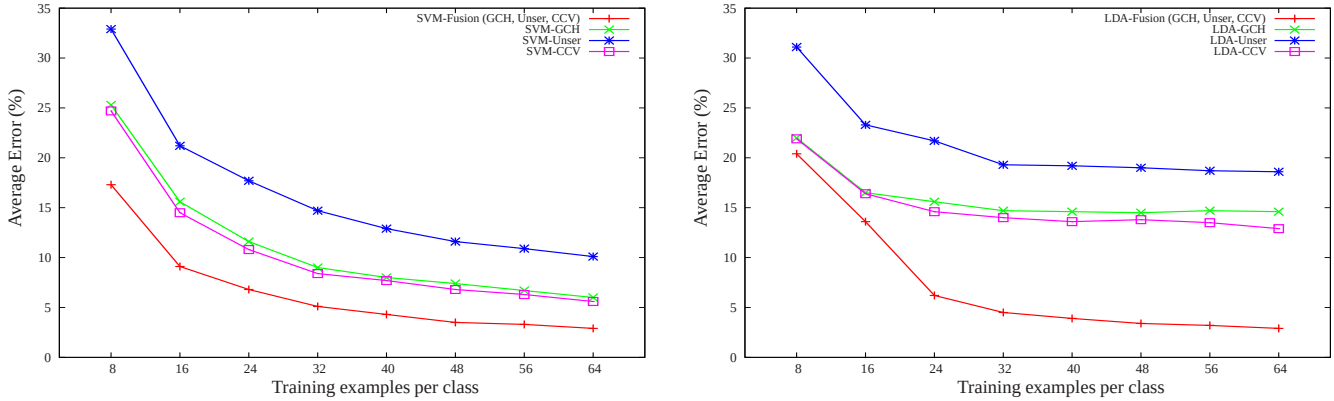
Oftentimes, when tackling complex classification problems, just one one feature descriptor is not enough to capture the classes' separability. Therefore, efficient and effective feature fusion policies may become necessary. Although normal feature fusion is quite effective for some problems, it can yield unexpected classification results when not properly normalized and preprocessed. Additionally, it has the drawback of increasing the dimensionality which might require more training data.

This paper approaches the multi-class classification as a set of binary problems in such a way one can assemble together diverse features and classifier approaches custom-tailored to parts of the problem. It presents a unified solution (Section 4) that can combine many features and classifiers. Such technique requires less training and performs better if compared with a naïve method, where all features are simply concatenated and fed independently to each classification algorithm.

The results show that the introduced solution is able to reduce the classification error in up to 15 percentage points with respect to the single features and classifiers. With the top one responses the solution yields a classification error $\epsilon \leq 2\%$ while with the top two responses such error is $\epsilon \leq 1\%$ using less than 40 examples per class.

A second contribution of this paper is the introduction to the community of a complete and well-documented fruit/vegetables image data set suitable for content-based image retrieval, object recognition, and image categorization tasks. We hope this data set will be used as a common comparison set for researchers working in this space.

Although we have showed that feature and classifier fusion can be worthwhile, it seems not to be advisable to combine weak features with high classification errors and features with low classification errors. In this case, most likely the system will not take advantage of such combination.



(a) SVM classifier (b) LDA classifier

Figure 11: Average error results for fusion of GCH, CCV, and Unser features considering SVM and LDA base learners.

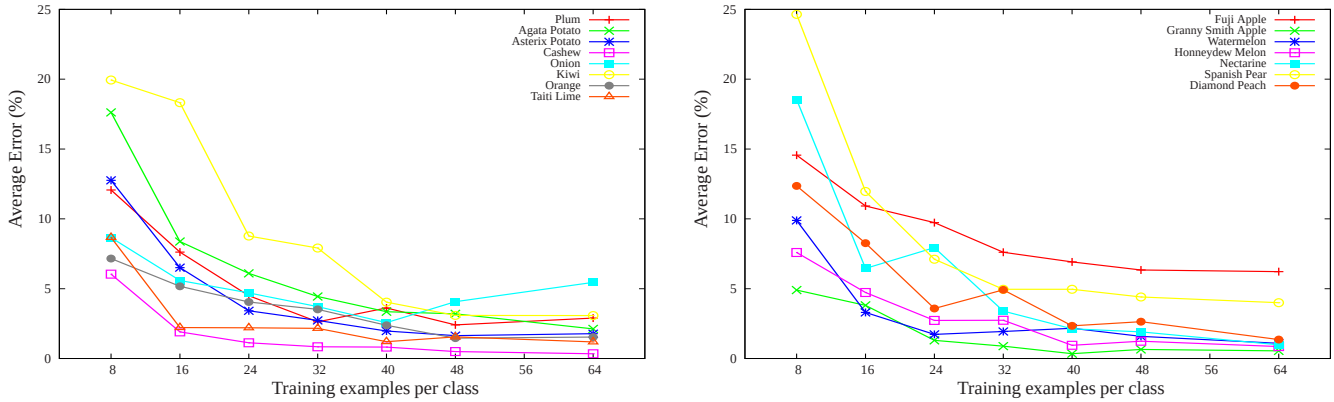


Figure 13: Average error per class using fusion of the features BIC, CCV, and Unser, for an SVM classifier.

The feature and classifier fusion based on binary base learners presented in this paper represents the basic framework for solving the more complex problem of determining not only the species of a produce but also its variety. Since it requires only partial training for the added features and classifiers, its extension is straightforward. Given that the introduced solution is general enough to be used in other problems, we hope it will endure beyond this paper.

Whether or not more complex approaches such as appearance-based descriptors provides good results for the classification is still an open problem. It would be unfair to conclude they do not help in the classification given that, their success is highly based on their patches representation. Such approaches are computational demanding and perhaps not advisable in some scenarios.

Further work includes the improvement of the fruits/vegetables representative patches, and the analysis of other appearance and texture image descriptors to point out produce varieties. Furthermore, we are interested in the incorporation of spatial constraints among the local descriptors.

In addition, we want to create the conditions for a semi-supervised approach that would lead to a continuous learning scenario, taking advantage of misclassified examples. In a semi-supervised scenario, the initial training stage can be simplified requiring only a few examples for each analyzed class.

Acknowledgments

The authors thank the people at the local produce distribution center for their patience and help. Finally, this research was funded by FAPESP (Award Number 2008/08681-9, 05/58103-3, 07/52015-0, and 08/54443-2) and CNPq (Award Number 309254/2007-8, 472402/2007-2, and 551007/2007-9).

References

Agarwal, S., Awan, A., Roth, D., November 2004. Learning to detect objects in images via a sparse, part-based representation. TPAMI 26 (11), 1475–1490.

Anand, R., Mehrotra, K. G., Mohan, C. K., Ranka, S., January 1995. Efficient classification for multi-class problems using modular neural networks. IEEE TNN 6 (1), 117–124.

Berg, A., Berg, T., Malik, J., 2005. Shape matching and object recognition using low distortion correspondences. In: CVPR. Vol. 1. pp. 26–33.

Bishop, C. M., 2006. Pattern Recognition and Machine Learning, 1st Edition. Springer.

Bolle, R. M., Connell, J. H., Haas, N., Mohan, R., Taubin, G., 1996. Veggievision: A produce recognition system. In: WACV. Sarasota, USA, pp. 1–8.

Comaniciu, D., Meer, P., May 2002. Mean shift: A robust approach toward feature space analysis. IEEE TPAMI 24 (5), 603–619.
Cutzu, F., Hammoud, R., Leykin, A., March 2005. Distinguishing paintings from photographs. CVIU 100 (3), 249–273.
Dietterich, T. G., Bakiri, G., January 1996. Solving multi-class learning problems via error-correcting output codes. JAIR 2 (1), 263–286.
Fei-Fei, L., Fergus, R., Perona, P., April 2006. One-shot learning of object categories. IEEE TPAMI 28 (4), 594–611.
Gonzalez, R., Woods, R., 2007. Digital Image Processing, 3rd Edition. Prentice-Hall.
Grauman, K., Darrel, T., 2005. Efficient image matching with distributions of local invariant features. In: CVPR. pp. 627–634.
Heidemann, G., October 2004. Unsupervised image categorization. IVC 23 (10), 861–876.
Jurie, F., Triggs, B., 2005. Creating efficient codebooks for visual recognition. In: ICCV. Vol. 1. pp. 604–610.
Lyu, S., Farid, H., 2005. How realistic is photorealistic? IEEE Trans. on Signal Processing (TSP) 53 (2), 845–850.
Marszalek, M., Schmid, C., 2006. Spatial weighting for bag-of-features. In: CVPR. pp. 2118–2125.
Narasimhamurthy, A., December 2005. Theoretical bounds of majority voting performance for a binary classification problem. IEEE TPAMI 27 (12), 1988–1995.
Otsu, N., January 1979. A threshold selection method from gray level histogram. IEEE Trans. on Systems, Man and Cybernetics 9 (1), 66–66.
Pass, G., Zabih, R., Miller, J., 1997. Comparing images using color coherence vectors. In: ACMMM. pp. 1–14.
Rocha, A., Goldenstein, S., 2007. PR: More than Meets the Eye. In: ICCV. pp. 1–8.
Rocha, A., Goldenstein, S., 2009. Multi-class from Binary: Divide to Conquer. In: Visapp. pp. 1–8.
Serrano, N., Savakis, A., Luo, J., 2004. A computationally efficient approach to indoor/outdoor scene classification. In: ICPR. pp. 146–149.
Shi, J., Malik, J., August 2000. Normalized cuts and image segmentation. IEEE TPAMI 22 (8), 888–905.
Sivic, J., Russell, B., Efros, A., Zisserman, A., Freeman, W., 2005. Discovering objects and their location in images. In: ICCV. pp. 370–377.
Stehling, R., Nascimento, M., Falcão, A., 2002. A compact and efficient image retrieval approach based on border/interior pixel classification. In: CIKM. pp. 102–109.
Turk, M., Pentland, A., 1991. Eigenfaces for recognition. Journal of Cognitive Neuroscience 3 (1), 71–86.
Unser, M., January 1986. Sum and difference histograms for texture classification. IEEE TPAMI 8 (1), 118–125.
Weber, M., May 2000. Unsupervised learning of models for object recognition. Phd thesis, Caltech, Pasadena, US.