Review of Automatic Image Orientation Methods

Mike Ghesquiere
University of Western Ontario
mghesqui@uwo.ca

1 ABSTRACT

In [1] Vailaya et al present "an algorithm for automatic image orientation using a Bayesian learning framework". Using Kohonen's learning vector quantization (LVQ) as well as comparing principal component analysis (PCA) and Linear Discriminant Analysis (LDA) for feature extraction. Performance was then compared to several well-known classifiers. This paper is summarized and its method independently implemented for several datasets. Finally, several other citing [1] are consulted. Modest results are achieved using the initially proposed algorithm on several unaltered datasets.

2 INTRODUCTION

With consumer image capture devices becoming increasingly prolific, personal image collections are growing quite rapidly. As such there is an increasing demand for automated, digital photo library management. One common problem are images incorrectly oriented. Whether this is due to a photo being placed on a scanner bed incorrectly or from a portrait-oriented image from a camera without a directional sensor. This can be a tedious process for the user which we aim to automate.

We assume that an image can be rotated to one of four orientations: 0˚, 90˚, 180˚, or 270˚. That is, it has been correctly aligned with the scanner bed. Thus the problem becomes a four-class classification problem. There are two main approaches to learning. One can look for high-level semantic features such as faces and use these to orient the image. This of course, would in this example require the image to contain a face. The other approach is to find low-level patterns that correlate with image orientation. The latter approach was taken for this paper. Generally high-level semantics for image orientation are not used alone but in conjunction with low-level features [2].

Following the impressive results of [1], an identical pipeline was attempted as follows, first, features must be selected from the image (Section 3). This gives a set of high-dimensional vectors of which only the most salient are extracted (discussed in Section 4). Several different classifiers are considered. Discussion on tuning each classifier occurs in Section 5 before comparing and discussing results in Section 6. Section 7 will cover several variations that other authors have tried to improve orientation before concluding in Section 8.

3 PREPROCESSING

There are many ways to select features from an image. For the problem at hand, any feature chosen must be rotation dependant. For this reason local features are much preferred over global features. Thus, images were broken down into $N \times N$ blocks and then features were extracted from each of the N^2 blocks. This approach was consistent between all papers surveyed. However, there was some variance in the choice of N as can be seen in Table 1. From these blocks, [1] extracted color moments (CM), that is, mean and variance, of each channel in LUV color space (total of 6 features per block). This can be thought of as an $N \times N$ thumbnail of the image (from mean) that additionally contains some texture information (from variance).

Table 1 N_{CM} and N_{EDH} used in cited works.

	N_{CM}	N_{EDH}	EDH bins	Length of feature vector
Vailaya et al. [1]	10	0	-	600
Luo et al. [2]	7	5	16+1	719
Takahashi et al. [3]	7	5	16+1	719
Cingovska et al. [4]	(8)[1]	(8)[1]	(8+1)[2]	720
Le Borgne et al. [5]	4	4	(4+1)[2]	176
Liu et al. [6]	5	5	12+1	475

[1] Inner 16 blocks were discarded leaving 48 blocks total.
[2] Although not explicitly specified, it is presumed that the 9 and 5 bins (quoted in [4] and [6] respectively), refers to 4 and 8 bins of directional edge pixels plus one non-edge pixel bin.

In addition, edge direction histograms (EDH) for each block were concatenated to the feature vector in [2], [3], [4], [6], and [5]. This was generally made possible by using by using a smaller N_{CM}, reducing the size of the feature vector quadratically. For the most part $N_{CM} = N_{EDH}$ was standard, however [2] and [3] used $N_{CM} = N_{EDH} + 2$ as color moments were found to be the stronger indicator of orientation [2]. Another interesting technique used by [4] was to *remove* the center blocks leaving only the outer 48 blocks of the original 72. This was cited as being due to "the information in the center of the image is typically very diverse and will add no effective features in the feature vector; it will increase complexity and even degrade classification performance" [4].

4 FEATURE EXTRACTION

As previously mentioned, the unmodified feature vector has an extremely high dimensionality (up to 720 dimensions). To avoid the *curse of dimensionality*, using feature extraction to reduce this to only the most pertinent features will reduce training time, memory usage and size of required training set. Two well-known techniques were tried by [1]: linear discriminant analysis (LDA) and principal component analysis (PCA).

4.1 PRINCIPAL COMPONENT ANALYSIS (PCA)

Given d-dimensional vectors, PCA will select the m features with highest variance. m is a tunable parameter, but should generally be such that $m \ll d$. This was optimized by finding minimal m that still maintains over 95% of the original variance. That is:

$$\left(\frac{\sum_{i=1}^{m} \lambda_i}{\sum_{i=1}^{d} \lambda_i} \geq 0.95 \right)$$

They found that m could be reduced to 100 (from 600) while maintaining this. On independent datasets, I found that this could be reduced as low as 80 to 90 validating these results (Figure 1).

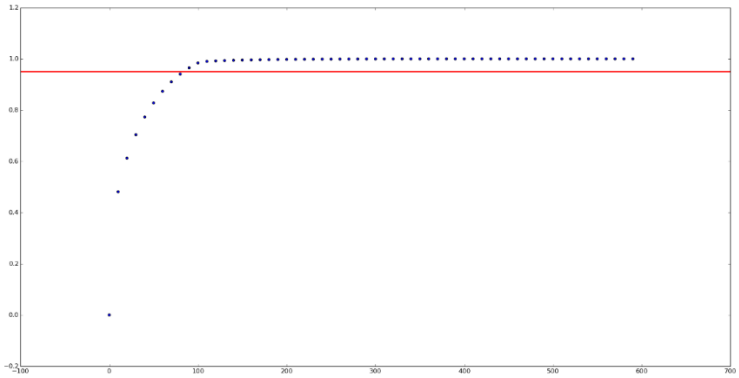


Figure 1: The VOC2007 dataset allowed for $m = 90$ *while maintaining a variance ratio* > 0.95.

Fig 1(a) shows effectively no distinction between the four classes after PCA. This is to be expected as PCA is unsupervised and should not be expected to find exactly the four clusters corresponding to image orientations.

4.2 Linear Discriminant Analysis

LDA on the other hand, has no parameters to be tuned. It will take the labels into account reduce the d-dimensional space to $(c-1)$-dimensions where c is the number of unique labels (i.e. 4). This proved very successful for the dataset used in [1]. Unlike PCA, when labels are included in the feature extraction process, as seen in Fig 1(b-d), the 4 orientations form near linearly separable partitions of the data. Although the authors praised LVQ for its classification, much of the heavy lifting appears to simply come from LDA.

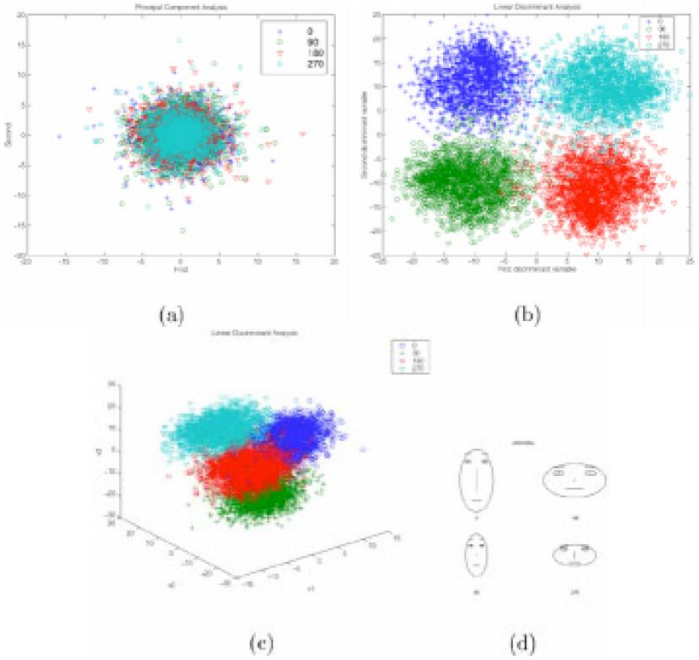


Figure 2 Taken from [1] (figure 4). Two-dimensional representation of data using (a) PCA, and (b) LDA as well as (c) the full 3-D representation of LDA. (d) shows the Chernoff face representation of (c).

This distinctness induced by LDA alone also draws some suspicion regarding the dataset chosen by the authors. The dataset used was a collection of 16 334 images taken from Corel stock photo library. In [2], the authors discussed their hand pruning of the training set, however [1] does not specify whether the dataset was naturally clean (professional photographs) or whether some pruning was performed.

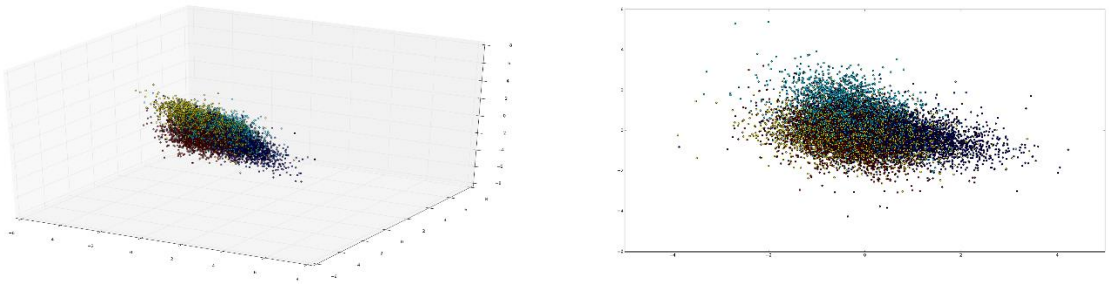


Figure 3: 3-D and 2-D scatter plots of the VOC2007 dataset after LDA transformation. There is some correlation, however, no clustering.

5 CLASSIFIER TUNING

Some classifiers require additional tuning before they can be reasonably compared against each other. Several classifiers are considered below.

5.1 *k* NEAREST NEIGHBOUR (*k*-NN) CLASSIFIER

The k nearest neighbour (k-NN) classifier is a very straightforward classifier yet generally gives good results in practice. In addition to choosing k, it must be decided whether to use voting (uniform) or distance weighting. Furthermore, k -NN is subject to relative feature scaling so generally normalization is used.

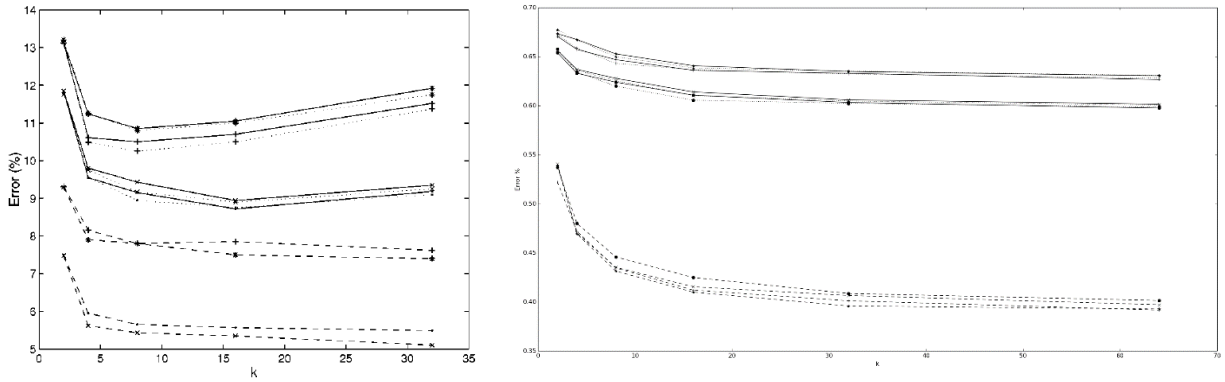


Figure 4: Error rates given by [1] (left) and by IndoorScenes (right). Plots are as follows: voting (uniform weighted) $(*)$, distance weighted $(+)$, voting (uniform weighted) with normalized data $(\times)$, and distance weighted with normalized data $(\cdot)$. Before feature extraction (solid lines), using PCA (dotted lines), and using LDA (dashed lines).

As seen in Figure 4, similar results (albeit with much higher error rates) were obtained independently as were presented in [1]. PCA provides no improvement in classification, however, it must not be discounted that it greatly reduces the size of the vector space giving an increase in performance. LDA is the clear winner for feature extraction. Reported results showed an increase in performance after normalizing the data, however seemed to have no impact in my testing. Parameters used for later comparisons were not explicitly stated in [1] but a 94.75% testing accuracy (100% training accuracy) for k-NN with LDA was reported in the final classifier comparison. This seems to imply that $k \geq 8$ with uniform weighting and normalized data was used. [5] reported success with $k = 5$ although also noted "the choice of classifier did not make a significant difference. However, I found k-NN to perform optimally with $k \sim 50$

Table 2: Traing and testing error rates for k-NN with varying k. Uniform voting was used on the IndoorScenes dataset to obtain the numbers below.

	without LDA		with LDA	
	Training	Testing	Training	Testing
k-NN ($k = 1$)	100%	33.6%	100%	45.0%
k-NN ($k = 5$)	57.4%	35.7%	67.7%	49.2%
k-NN ($k = 10$)	51.7%	37.9%	63.7%	52.8%
k-NN ($k = 50$)	44.5%	40.0%	61.5%	55.8%
k-NN ($k = 100$)	43.3%	39.8%	62.1%	55.0%
k-NN ($k = 500$)	41.4%	38.8%	61.1%	55.2%

5.2 SUPPORT VECTOR MACHINE (SVM) CLASSIFIER

Another commonly used classifier is the support vector machine (SVM) classifier. Tuning of SVM was discussed by [1] very briefly. They tried linear, polynomial, radial basis function (RBF) and sigmoid kernels but found degree 3 polynomial to give the best results. From figures in [5], negligible differences in performance between polynomial (unknown degree) and sigmoid kernels. In my own testing, this was not the case. Sigmoid kernels consistently gave significantly lower accuracy by 10-20%. On the other hand, although degree 3 polynomial kernels gave the highest accuracies among polynomials, RBF and linear kernels slightly outperformed by 1-3%. RBF kernel accuracy was not significantly affected by choice of gamma. These trends held across all datasets with the exception of the "easy dataset" where RBF kernels gave near chance predictions. It is unclear whether this was simply caused by a smaller training set. For this reason, I chose to use linear kernels as even in the worst-case, it gave within 0.7% of the accuracy of the best SVM classifier.

Table 3 Comparing testing accuracy of linear (lin), polynomial (poly), radial basis function (rbf) and sigmoid (sigm) kernels on each dataset. Training accuracies are omitted.

	VOC2007	**IndoorScenes**	**VOC2012**	**"Easy"**
lin	40.5%	55.5%	45.4%	65.6%
poly ($\boldsymbol{d = 2}$)	36.3%	40.5%	37.2%	44.0%
poly ($\boldsymbol{d = 3}$)	39.8%	53.9%	43.8%	61.6%
poly ($\boldsymbol{d = 4}$)	33.4%	37.0%	34.4%	38.4%
poly ($\boldsymbol{d = 5}$)	38.9%	47.3%	38.5%	58.4%
poly ($\boldsymbol{d = 6}$)	31.4%	36.9%	31.1%	43.2%
rbf ($\boldsymbol{\gamma = 2^0}$)	40.5%	55.6%	46.1%	24.0%
rbf ($\boldsymbol{\gamma = 2^{-1}}$)	40.4%	55.7%	45.9%	24.8%
rbf ($\boldsymbol{\gamma = 2^{-2}}$)	40.9%	55.6%	46.0%	25.6%
rbf ($\boldsymbol{\gamma = 2^{-3}}$)	40.5%	55.4%	45.8%	32.0%
rbf ($\boldsymbol{\gamma = 0}$)	41.1%	55.7%	46.0%	25.6%
sigm ($\boldsymbol{\gamma = 2^0}$)	30.2%	36.3%	29.2%	36.8%
sigm ($\boldsymbol{\gamma = 2^{-1}}$)	30.9%	36.5%	32.0%	41.6%
sigm ($\boldsymbol{\gamma = 2^{-2}}$)	33.4%	39.2%	34.5%	41.6%
sigm ($\boldsymbol{\gamma = 2^{-3}}$)	35.1%	43.4%	37.9%	41.6%
sigm ($\boldsymbol{\gamma = 0}$)	25.8%	25.8%	25.3%	24.0%

5.3 LEARNING VECTOR QUANTIZATION (LVQ)

Kohonen's learning vector quantization (LVQ) [7], [8] is a supervised classification model which aims to find a set of *codebook vectors* which model the data. These codebook (or reference) vectors generally denoted $m_1, \ldots, m_q$ with q being the *codebook size* or *length*. Each m_i will have a class associated with it although each class may (and it is generally to) have several codebooks associated with it. Kohonen has proposed several variations of the LVQ algorithm [8] with subtle differences in how codebook vectors are treated. In the original algorithm (LVQ1), classification is done in a nearest neighbour fashion. Implementation contained with [9], also included LVQ2.1, LVQ3, and OLVQ1. It is unclear which algorithm was used by [1], however, I chose to use OLVQ1 for its faster convergence [9].

$$\hat{q} = \arg\min_{q} \left\{ -\sum_{j=1}^{n} \log f\left(\boldsymbol{y}^{(i)}(j) \middle| \omega, \boldsymbol{\theta}_{(q)}\right) + \frac{q}{2}\log n + \frac{\dim\left(\boldsymbol{y}^{(i)}\right)}{2} \sum_{j=1}^{q} \log\left(m_j^{(i)} n\right) \right\} \quad (1)$$

To select optimal codebook length $\hat{q}$, [1] used the above *modified minimum description length* (MMDL) criterion was used. MMDL can be broken down into the sum of 3 terms. The first term is negative log-likelihood over n samples. The more accurate a classifier f is at predicting class ω from a sample $\boldsymbol{y}_j^{(i)}$, the smaller this term will become. The second term the authors claim "accounts for the weights $m_j^{(i)}$" [1] although this is not clear. The third term relates the proportion of samples $m_j^{(i)}$ corresponding to each codebook. This will be minimized for "imbalanced" codebooks where there is a large variance in proportion of samples covered by each codebook vector. Again, it is unclear why this is desirable.

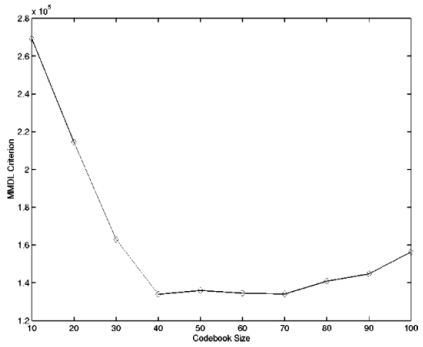


Figure 5: Taken from [1], MMDL was found to be minimized for $q \approx 40$ *(ten codebook vectors per class).*

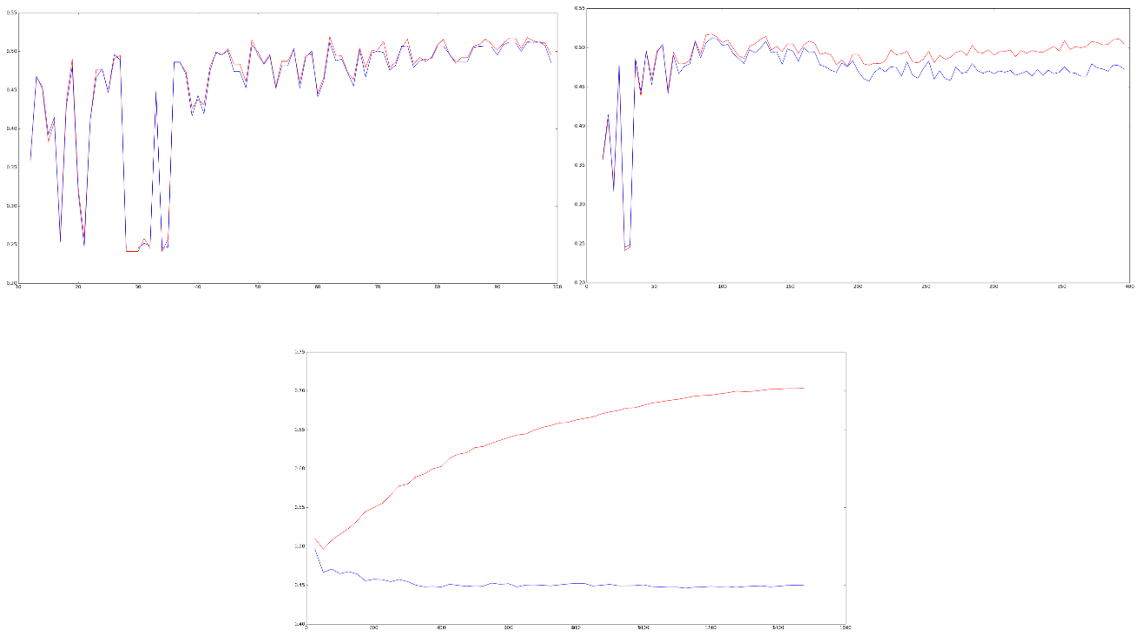


Figure 6: Training accuracy (red) and testing accuracy (blue) vs. codebook size over increasing scales. Whereas [1] found $q \sim 40$ *to be optimal,* $q \sim 100$ *(25 codebooks per class) appears to be optimal.*

5.4 OTHER CLASSIFIERS

Many other classifiers have been attempted for this problem; hierarchical discriminating regression (HDR) trees and a mixture of Gaussians were attempted by [1]. Little information was found on the HDR algorithm let alone an implementation. As this was used only for a comparative standard, I did not investigate it deeply. Good results were also achieved with AdaBoosting in [6] so this was used for comparison instead. Performance of AdaBoost can be increased by increasing the number of estimators with less risk of overfitting compared to other classifiers. As can be seen in Table 4, although LDA increases performance when fewer estimators are used, AdaBoosting with many estimators can actually achieve higher accuracy *without* LDA. However, this comes at the cost of performance. If features have been extracted through LDA, 10 estimators are sufficient, otherwise, at least 50 estimators is recommended.

Table 4: Accuracy of AdaBoost with varying number of estimators on the IndoorScenes dataset.

	without LDA		with LDA	
	Training	Testing	Training	Testing
AdaBoost (5)	40.9%	40.0%	58.1%	50.8%
AdaBoost (10)	48.0%	46.6%	59.1%	51.6%
AdaBoost (25)	52.3%	49.9%	60.7%	51.4%
AdaBoost (50)	54.3%	51.9%	60.8%	51.4%
AdaBoost (100)	56.7%	52.1%	61.3%	51.6%
AdaBoost (250)	59.9%	52.1%	61.6%	51.8%

6 RESULTS

There was a large variance in reported results. SVM was considered in all papers surveyed so it provides a good standard of comparison. Although [1] claims an accuracy over 96% on their testing dataset this is considerably higher than the 68.8% for CM only on consumer images reported by [2]. [6] and [5] reported as high as ~78% for basic SVM. Of course, each had differences in processing (as discussed in Section 3), however it gives a reasonable comparison in relative 'difficulty' of datasets. In my own testing the following accuracies were found for each dataset:

Table 5 SVM Accuracies between datasets.

	Training	Testing
VOC_2007 (4952)	61.9%	42.9%
VOC_2012 (17125)	52.7%	45.1%
IndoorScenes (13663)	61.3%	55.4%
"easy" (500)	100%	54.4%

These results struck me as rather low even compared to the poorest surveyed results. Training set accuracy seems to be correlated with dataset size which should not be surprising (half of the dataset was used for training). Relative testing rates are also in line with anticipated results. The VOC contest datasets contain a large range of images many of which cannot be reasonably expected to be correctly classified (Figure 6). Selecting only the 'easiest' images from this set resulted in testing accuracy similar to the, also constrained, set of IndoorScenes.

a)  b)  c)  d) 

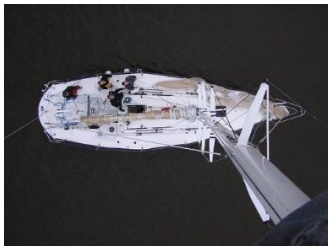
e)  f)  g) 

Figure 7 Examples of images contained in both VOC_2007 and VOC_2012 which cannot reasonably be expected to be predicted by a CM-based image orientation classifier. Either due to unifom background (g), deceptive perspectives (c), relying on higher-order semantic knowledge (b, f) or even being inherently ambiguous to a human observer (a, d, e).

Based on results from Figure 4, PCA achieved no improvement to classification over the unreduced features. Thus only with and without LDA are considered for comparing classifiers. Interestingly, the best performing classifiers from [1] (LVQ and mixture of Gaussians), turned out have the lowest accuracy of all properly tuned classifiers tested for this dataset (50.2% and 44.0% respectively). SVM, and k-NN both attained similar accuracies.

Table 6: Comparison of classifiers on the IndoorScenes dataset

	without LDA		with LDA	
	Training	Testing	Training	Testing
k-NN ($k = 50$)	44.5%	40.0%	61.5%	55.8%
SVM (linear kernel)	100%	41.5%	61.3%	55.4%
LVQ ($q = 100$)	-	-	50.6%	50.2%
AdaBoost (50)	54.3%	51.9%	60.8%	51.4%
Mixture of Gaussian	-	-	49.6%	44.0%

7 EXTENSIONS

Attempts to extend this framework have generally focussed on one of two areas. Incorporating high level semantic cues or trying to optimize user satisfaction

7.1 SEMANTIC CUES

To determine how humans can easily orient images, the authors of [2] asked participants to orient a set of mostly incorrectly oriented images as well as specifying what semantic cues they used. From this, they found that *sky*, *grass* and *people* accounted for more than 70% of the correct observations. By training several models, each for specific features (e.g. face detection, sky detection) the results can be combined together for better accuracy. This has been attempted in both a hierarchical fashion [4] and using a Bayesian network [2] to good success.

7.2 User Satisfaction

Raw accuracy may not always be the best score for comparing algorithms however. Users tend to be less happy when an image is 'unjustly' rotated from a correct orientation than they are for incorrectly oriented images which are not detected [4]. For this reason, classifiers with low false positive rates are to be preferred. One approach is to pre-filter images which are likely to become false positives and simply not even attempt to correct them. This was found to decrease user's annoyance while using the system [3].

8 Conclusion

Image orientation appears to be a problem heavily dependent on the dataset being used. Although very high accuracy was originally achieved in [1], this was on a set of professional Corel photos. Subsequent papers have quoted much more reasonable results of 82.5% [2], 79.2% [4] and 76.8% [5] using more complex architectures. By comparison, 55.8% accuracy is quite modest.

There are several obvious areas for improvement. Firstly, incorporating EDH into the feature vector I believe would give a large boost to performance. Especially for the indoor scene dataset, it is not difficult to see that vertical lines would be strongly correlated. Indoor images are rarely taken orthogonal to a wall (potentially confusing horizontal edges along floors and ceilings) but nearly always contain at least one strong vertical edge where two walls meet.

Secondly, the training and testing sets were not consistent between all tests. Fixing the training set would allow for hand-pruning of noisy samples as well as more consistent results between runs. This task of hand-pruning a large dataset (ideally total dataset size of 10,000+ images) is admittedly tedious and daunting.

Finally, incorporating specialized detectors for sky/faces or classifiers for dark/light images presents a large area to be investigated. Face detection inarguably provides a good improvement. There are inexhaustibly more possible features that may or may not be strongly correlated with orientation. Detecting image class is another interesting direction. By first detecting whether an image is say, indoor vs. outdoor, a system could then switch between classifiers that were each trained on only one class of image.

9 References

[1] A. Vailaya, H. Zhang, C. Yang, F.-I. Liu and A. K. Jain, "Automatic Image Orientation," *IEEE Transactions on Image Processing,* vol. 11, no. 7, pp. 746-755, 2002.

[2] J. Luo and M. Boutell, "Automatic Image Orientation Detection via Confidence-Based Integration of Low-Level and Semantic Cues," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 27, no. 5, pp. 715-726, 2005.

[3] Y. Takahashi and R. Oami, "Relevant Image Pre-filtering in Automatic Image Orientation Correction," in *IEEE International Conference on Image Processing*, 2010.

[4] I. Cingovska, Z. Ivanovski and F. Martin, "Automatic Image Orientation Detection with Prior Hierarchical Content-Based Classification," in *IEEE Conference on Image Processing*, 2011.

[5] H. Le Borgne and O. E. Noel, "Pre-classification for Automatic Image Orientation," in *IEEE Conference on Acoustics, Speech and Signal Processing*, 2006.

[6] X. Liu, L. Zhang, M. Li, H. Zhang and D. Wang, "Boosting image classification with LDA-based feature combination for digital photograph management," *Journal of the Pattern Recognition Society,* vol. 38, no. 6, pp. 887-901, 2005.

[7] T. Kohonen, "Learning Vector Quantization," *Neural Networks,* vol. 1, no. 1, p. 303, 1988.

[8] T. Kohonen, "The self-organizing map," *Proceedings of the IEEE,* vol. 78, no. 9, pp. 1464-1480, 1990.

[9] T. Kohonen, J. Kangras, J. Laaksonen and K. Torkkola, "LVQ PAK: A program package for the correct application of Learning Vector Quantization algorithms," 1992.

[10] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot and E. Duchesnay, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research,* vol. 12, pp. 2825-2830, 2011.

[11] G. Bradski, "OpenCV Library," *Dr. Dobb's Journal of Software Tools,* 2000.

[12] M. Everingham, L. van Gool, C. K. I. Williams, J. Winn and A. Zisserman, "The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results," [Online]. Available: http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html.

[13] M. Everingham, L. van Gool, C. K. I. Williams, J. Winn and A. Zisserman, "The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results," [Online]. Available: http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[14] A. Quattoni and A. Torralba, "Recognizing Indoor Scenes," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.