Term Paper on

Automatic Image Orientation Detection

Computer Vision – CS519

Cal Poly Pomona Winter ‘03

John Leung

Table of Contents:

1. Summary

My main interest in this research is coming from the fact that I have many pictures from friends and myself that I have taken over the years with my digital camera. Re-orientation of it is not only a time consuming task, but will also result in quality loss if the software did not implement a loss-less jpeg rotation. Thus I would like to find some good research papers on this and see if I can gather good ideas from them.

I have learned many terminologies and methods in dealing with classification in this research. For me to remember this terminologies and methods easier, I have composted a terminologies section so I can refer back to them easily in the future. The paper will start with the introduction to the problem and later sections will discuss ways to deal with it. Enjoy!

2. Introduction

Many tasks in computer vision and digital image processing require the knowledge of the image orientation. Examples of such tasks include image organization and storage, analysis and retrieval, image enhancement, and surface texture extraction, just to name a few. The problem is that during image capturing, when scenes or objects are vertically oriented, we normally tilt our cameras 90 degrees to the left or to the right to capture the full image. When we want to digitize or upload the images, we need to re-orient the images manually. This is a trivial task when dealing few images, but a very time consuming task when dealing with hundreds or thousands of images. Thus an automation of identifying the true image orientation is needed.

The methodology I tried to follow is to gather as much sources as possible so I can get a survey and compare the various ways to automate the image orientation process. However, after spending many hours of searching, not only did I found just three papers related to this topic, but also only one out of the three were actually useful. The first is *Automatic Image Orientation Detection* by Aditya Vailaya, Hong Jiang Zhang, and Anil Jain; this will be the paper I am presenting in this report. The first of the other two papers I found contain no technical information at all. The other one deals mainly with three-dimensional images, which I wanted to stay away from. The reason is that I wanted the report to be more in depth in two-dimensional methods rather than trying to cover too much and ending up not gaining any useful analysis in either. The

way I will work around this problem is to mainly present the idea and methods proposed from the first paper, and in the analysis section, I will resort to books and other resources for alternative methods or enhancements. Terminologies used which are not covered in the course or requires more explanation will also be covered in the background section to follow.

Below is a demonstration of what the proposed system can do when a sample image is presented in the four possible orientations:

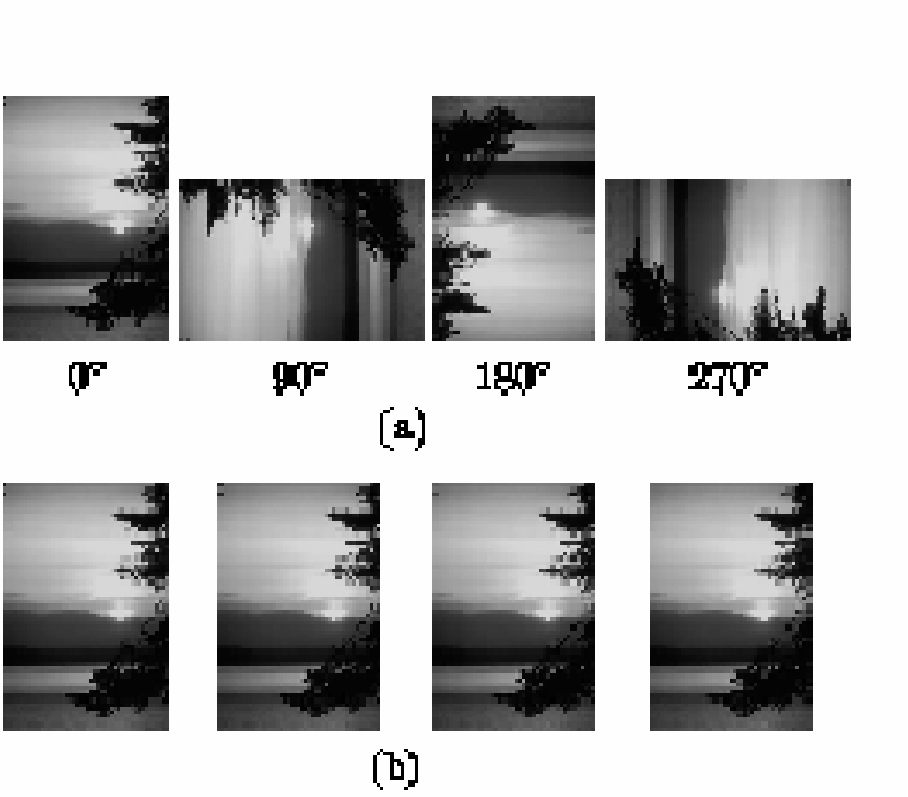

Fig.1. Sample images

(a) is the same input image oriented in the four possible orientations

(b) is the correct orientation detected by the system

3. Background

The automation of image orientation detection is a very difficult task. Humans use object recognition and contextual information to correctly identify the image orientation. For machines to find the orientation of images, sometimes recognition of ALL objects in the image is not necessary. For example, if the image contains people, chairs, or anything that is normally upright, we can use that to determine the orientation of our image (of course, there are always exceptions; we will either have to account for these exceptions or find out the probability of occurrence of these images and decide if it is better to just not account for them). However, the task is not so easy for many images, especially when we are dealing with images that contain objects that we have no knowledge of. This is why we still have a long way to go before achieving a full object recognition system. Thus not only do object recognition proposes a problem, but even if we can recognize the objects, we also need to know the context of the recognized objects. The following can best explain this:

Fig.2. Sample image

This image may even fool a human if not looked at more carefully. After inspecting more carefully, one can see that the background is a cloudy sky during sunset and the correct orientation of it is a 90 degrees rotation to the left. The tree is just the left side of the tree branches, not the tip of the tree. So how do we face both the problem of imperfect object recognition and also taking account of contextual information? The key is local feature extraction rather than object recognition.

3.1. Terminology

Color moments [3] – the first few low-order moments of each color channel. It is a compact representation of the color distribution of an image.

LUV – a color space system where L = intensity, U = blue with some green, V = red with some yellow.

HSV – a color space system where H = Hue, S = Saturation, V = Value.

MRSAR [4] is a dissimilarity measurement model for capturing random textures. It uses Mahalanobis distance and Euclidean mean to calculate the texture features and cluster centriods.

A priori probability [5] – it is based on the knowledge of the likelihood of successes of different events to predict the probability of the success of any given event.

Class-conditional probability density [6] – the probability that feature x arises for objects know to be in class C_C, denoted as $p(x|C_i)$.

Vector quantization [7] – A mapping of n-dimensional vectors in vector space into a *codebook,* which is a finite set m of vectors $Y = \{y_i$, for i = 1 to m$\}$. Each vector y_i is called a code vector or a *codeword*.

Maximum a posteriori (MAP) [8] - A posteriori means after (the fact), normally derived from experience or observation. MAP is class of nonlinear demodulation algorithm, which minimizes error probability in systems which use concatenated or interleaved coding.

4. Proposed Method

The authors of the paper proposed that rather than trying to identify and recognize the objects, find the local features instead. For outdoor images, they tend to have uniformity in spatial color distributions. For example, sky should be at the top and has more uniformity horizontally than vertically; grass is at the bottom and is typically green, just to name a few features. For indoor images, we sometimes have wall edges to help us and other information such as source of light is normally from the top. All these can be best obtained by local feature extraction rather than global extraction.

4.1. Feature Extraction

The specific method used is the represent the image in N x N blocks (the authors' actual implementation was 10x10). And from these local region blocks, the following four features are extracted:

1. Color moments in the LUV color space
2. Color histograms in the HSV color space
3. Edge direction histograms
4. MRSAR texture features

These features are then grouped into a feature vector set for each image. An interesting note is that the authors have found that out of the four features above, the color moments in the LUV color space yielded a much higher accuracy than the other three.

4.2. The Process

The extraction of these features is used for both the training process and the actual detection process. The following diagram is the proposed overall general process involved:

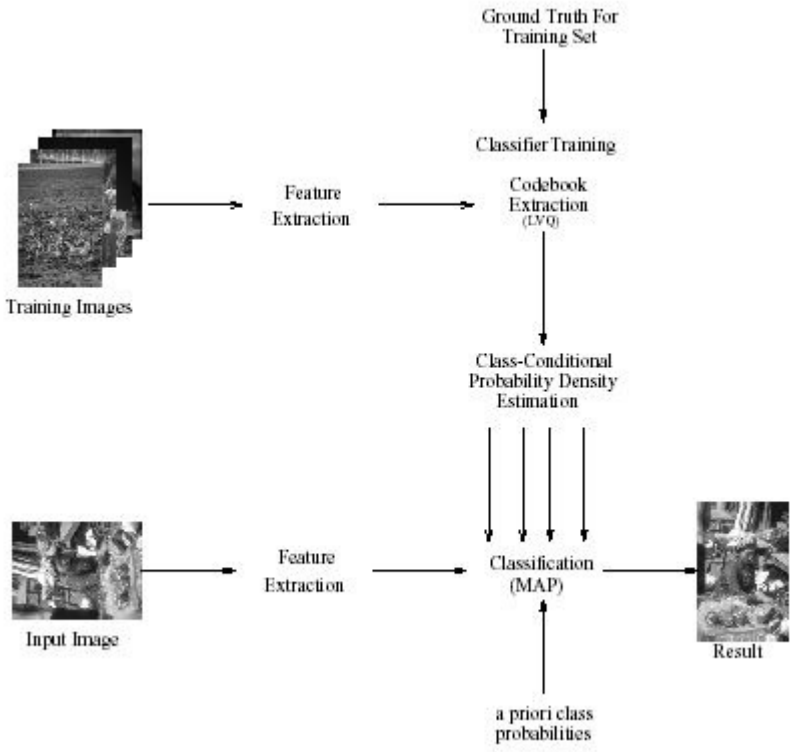

Fig.3. Detection system

This whole detection process can be best described as a two-part process. The first part is the training system. In it, we tell the system the orientation of features extracted from the training images, code them, and obtain the class-conditional probability density estimation for the actual detection part. The second part is the actual

detection. We extract features from the input image just as the training images and classify them based on both the class-conditional probability density estimation from the training system and a priori class probabilities. The result is correctly (hopefully) oriented images.

4.3. Training

The authors trained their Bayesian Classifier using 600 (100 blocks * {3 mean and 3 variance values of L, U, V components}) spatial color moment features. They noted that a key issue in using vector quantization for density estimation is the choice of the codebook size. Given a training set, the vector quantization approximated probability of the training set will increase as the dimension of the codebook grows. To solve this, the authors used the minimum description length (MDL) [9] principle. The MDL criterion states that the description of the code length to be minimized by the estimate must include not only the data code length, but also the code lengths of the parameters. Using MDL, the optimal codebook size is determined. The resulting criterion for the choice of the codebook size is then:

$$\hat{q} = \arg\min_q \{L(Y|\theta_{(q)}) + L(\theta_{(q)})\}$$

where the first term represents the data code length and the second term represents the code length of the parameters.

4.4. Classification

The system is based on four orientation classes $w_1, w_2, w_3,$ *and* $w_4,$ representing the four possible orientations. Classification is based on Bayes decision theory and is used for both the training system and the detection system (as shown in the detection system diagram above). Each image is represented by a feature vector extracted from the image (explained earlier). During the training phase, the class-conditional probability density function of the feature vector is estimated using vector quantization. The class-conditional density of a feature vector **y** given class *w,* each centered at a codebook vector is:

$$f_{\mathbf{Y}}(\boldsymbol{y} \mid \omega) \propto \sum_{j=1}^{q} m_j * \exp(-\|\boldsymbol{y} - \boldsymbol{v}_j\|^2/2)$$

where:

q is the codebook vectors extracted from the samples using vector quantization, and

m_j is the proportion of training samples assigned to $\mathbf{v}_{j..}$

During the detection stage, the Bayesian classifier is then defined using maximum a posteriori (MAP) criterion:

$$\hat{\omega} = \arg\max_{\omega \in \Omega} \{p(\omega \mid \boldsymbol{y})\} = \arg\max_{\omega \in \Omega} \{f_{\mathbf{Y}}(\boldsymbol{y} \mid \omega)\, p(\omega)\}$$

where:

p(*w*) represents the a priori class probability (also shown in the detection system diagram above) of the four orientation classes Ω.

4.5. Results

The following are sample images that are correctly oriented by the system:

Fig.4. Sample correct oriented images

And incorrectly oriented ones:

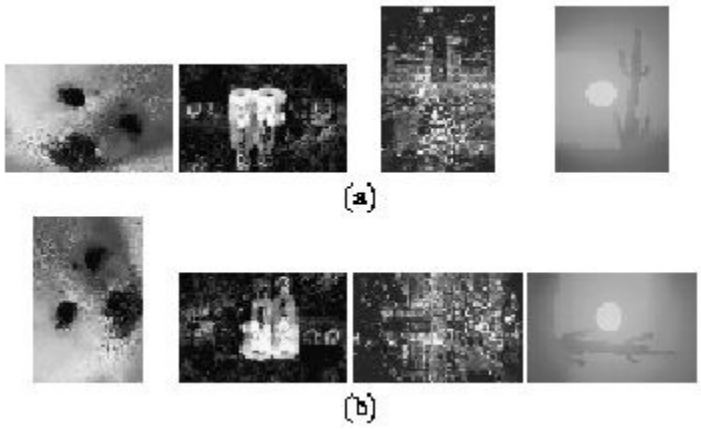

Fig.5. Sample incorrect oriented images

(a) Input images

(b) Incorrectly oriented outputs

5. Analysis

5.1. Results Analysis

The authors tested their system on two sets of images D1 and D2. D1 consists of 16,392 professional quality images from Corel stock photo library and D2 consists of 1,509 amateur photographs taken from a digital camera. When D1 is used for both the training data sets AND the test images, accuracy was over 96%, but drops to 87% on independent test data. The following diagram shows the actual results of various combinations of D1 and D2 used as training sets versus different sizes (number of images) of independent test images:

Training Set (Database:Size)	Independent Test Set (Database:Size)	Accuracy (%)	Number of Misclassifications
$D1$: 8,000	$D1$: 8,392	87.2	1,071
$D2$: 755	$D2$: 754	87.3	97
$D1$: 16,392	$D2$: 1,509	77.3	342
$D1+D2$: 8,755	$D1+D2$: 9,146	87.8	1,112

Fig.6. Test results

This seem a little confusing to me. If D1 for the test data is the same as the D1 used for training, then how can the authors' test results (96%) be so much different than the independent test results (87.2%, first row). If the D1 for test data IS different than the D1 used for training, then why did they use the same symbol? Other then this little mystery, the authors cited the reasons for the results (the third row versus the other rows) were

mainly due to the fact that D2 consists of amateur pictures taken under poor lighting conditions using a digital camera.

5.2. Reject Option

The authors noted that a reject option can be used that can be based on the a posteriori class probabilities. Basically, this means that the system can reject images whose maximum a posteriori probability is less than certain threshold. The result improved from 89.3% (no rejection) to 96.6% at 39.2% reject rate. (OK, so this is why their test results were different than the independent test results when BOTH tests used the same sets of data). I believe setting the reject rate this high (39.2%) is unacceptable because this will make the orientation detection system useless since the whole purpose of it is to automate the process. If 39.2% (this is almost half) of the images are rejected, it means that an operator will have to manually orient the reject images. Setting the reject rate to a single digit percentage should be fine; according to the chart below, this is a good compromise and will yield a few percentage improvements to the accuracy.

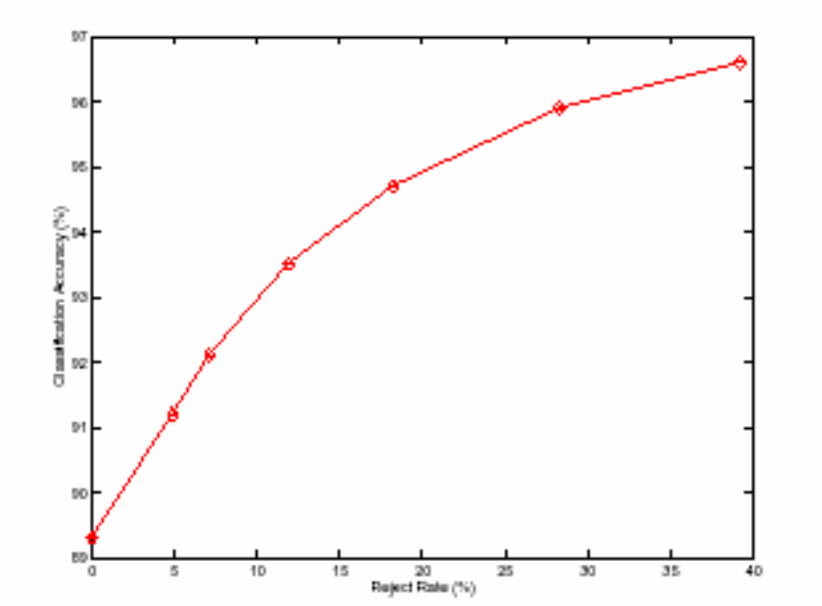

Fig.7. Classification accuracy vs. reject rate

5.3. Feature Selection

The authors also cited that using more features in the classification may deteriorate the detection. This phenomenon can be best explained according to Davies [6], that there is an optimum range of feature sets depending on the amount of training a classifier receives. If the number of training sets is increased (thereby increasing the patterns), then more features can be stored in the classification. However, if the number of features is increased to the extent that it out numbers the optimal ratio of training sets versus features, then the classification will deteriorate. The authors did address this issue, but not completely.

First, they only showed results using feature sizes of 50, 75, 100, 150, 200, then a big jump to 600 (600 is the number of features they implemented). It is true that the accuracy for features sizes between 200 to 600 will probably be the straight line they have plotted, but there is also a chance that there may be a little bump in there that the accuracy is the optimal. This is very possible since the accuracy between feature sets of 50 to 200 oscillates. Also, why did they stop at feature set of 600, why not test it with more? Below is the diagram showing the test results:

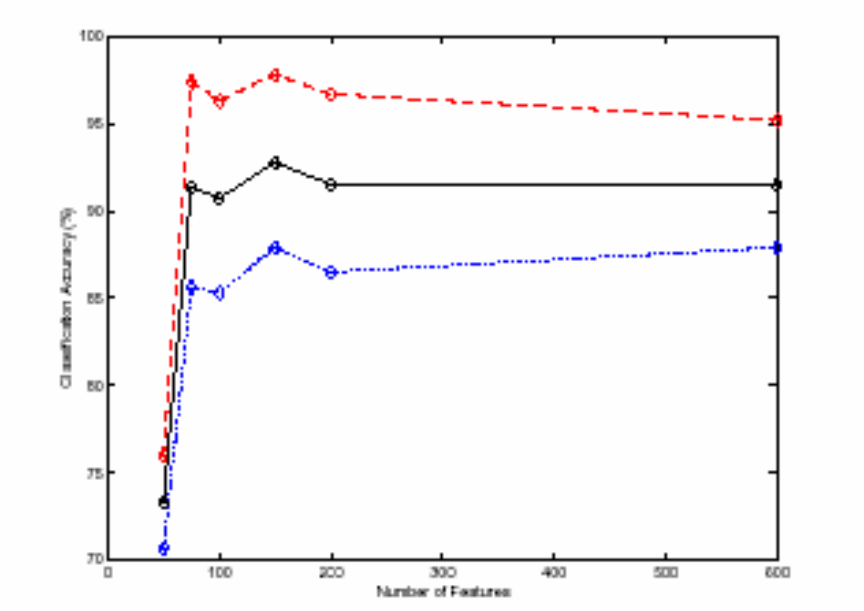

fig.8. Classification accuracy vs. number of features,

red = training set, black = test set, blue = entire database

6. Conclusion

As mentioned before, my main interest in this research is coming from the fact that I want to learn about methods to automate the orientation process. Realistically, the results shown were not too promising. One way I believe this detection system can drastically be improved if they have implemented a feedback system where it learns from its mistakes (at first, I thought the a priori class probabilities were such a feedback, but I was wrong). For example, the results of the incorrectly oriented or rejected image can be feed back to the system as training images and under a supervised method, the operator can tell the system which feature classifications were wrong. Of course, this will require new sets of feature to be decided and implemented and this can be a difficult task based on the predefined feature sets. If implemented however, after certain amount of feedbacks, this system should produce a very high accuracy autonomously (without further supervision).

Overall, I believe this is a very good orientation detection system. Further studies can be based on this to produce an even better system, making it a robust orientation system.

7. References:

[1] Aditya Vailaya, Hong Jiang Zhang, and Anil Jain. *Automatic Image Orientation Detection, 1999.* http://citeseer.nj.nec.com/cs

[2] Leif Haglund and David Fleet. *Stable Estimation of Image Orientation, 1994.* http://citeseer.nj.nec.com/cs

[3] Xiuqi Li, Shu-Ching Chen, Mei-Ling Shyu, and Borko Furht. *An Effective Content-based Visual Image Retrieval System, 2002.* http://citeseer.nj.nec.com/cs

[4] Rui Li and Wee Kheng Leow. *From Region Features to Semantic Labels: A Probabilistic Approach, 2002.* http://www.comp.nus.edu.sg/~leowwk/papers/mmm2003.ps.gz

[5] http://web.hku.hk/~hrnwlck/introstat/prob1.htm

[6] E. R. Davies. *Machine Vision, 1997.* Academic Press

[7] http://www.geocities.com/mohamedqasem/vectorquantization/vq.html

[8] Kang Wang. *Synchronization and Demodulation Algorithms for Continuous Dmodulation; Thesis, 2000.* http://cwc.ucsd.edu/~kwang/MS-thesis.ps

[9] Aditya Vailaya, Hong Jiang Zhang, and Anil Jain. *Automatic Image Orientation Detection, 1999.* http://citeseer.nj.nec.com/cs

[10] Emanuele Trucco. *Introductory Techniques for 3-D Computer Vision, 1998.* Prentice Hall

[11] R. C. Gonzalez and R. E. Woods. *Digital Image Processing, 2002.* Prentice Hall