VQ based Image Retrieval using Color and Position Features

Ajay H. Daptardar*and James A. Storer
Computer Science Department, Brandeis University,
Waltham, MA

Abstract

Two methods for incorporating position features along with the more traditional color features are introduced for the content-based image retrieval task. Both methods use vector quantization (VQ) to compute image similarity by encoding query image features using database image codebooks. Database images that encode the query at low distortion are considered similar to the query. The first method trains VQ codebooks separately for color and position features. Separate training alleviates the problem of feature weighting during the training phase and allows different weightings while querying. The second partitions images into regions and trains tree structured codebooks (TSVQ) for these regions. During the query phase, the number of codewords for each region is dynamically determined. The pruned TSVQ codebooks are then used to encode query image regions. Both of these approaches, especially the second, have significantly lower complexity than previous techniques that employ VQ, and experiments performed on the COREL image database indicate that the performance of our methods is comparable (slightly better than) these previous methods.

1 Background

With the recent proliferation of digital images, there is a need for information systems that can organize and understand images. In the query-by-example model (Eakins and Graham [3]), a user presents the system with a query image and the system responds by retrieving a set of database images with (visually) similar content. Relying on the discriminative power of color features, histograms measuring color volumes have been efficiently and effectively for this task (Swain and Ballard [10], Faloutsos et al. [4], Flickner et al. [5]). However, a drawback of global color histograms is that if images with different statistical properties (classes) are later added to the

*Contact: ajay@brandeis.edu

database, the histogram colors (labels) may need to be recomputed to maintain retrieval accuracy. When dealing with varying databases (the web, for example), it can be advantageous to use models generated individually from each image.

Probabilistic models are favorable in this setting since they are provide minimum probability of error retrieval. Vasconcelos [11] uses finite Gaussian mixture models to represent image densities and uses Bayes classifiers for retrieval, i.e the most similar image (class) is the one that maximizes the posterior probability of the database image (class) given the query image. Jeong and Gray use finite Gaussian mixture models where an image's similarity is explicitly computed as the probability of the query features given a database model [8]. Their method performs better than Vasconcelos' asymptotic likelihood approximation (ALA), method which uses distributional distances as similarity measures. We introduced similar approach using VQ codebooks [1], where the similarity was computed by determining the how well a database image's codebook can represent a query image. Although this method is not explicitly probabilistic, it compares well with more complex Gaussian mixture models and is computationally more favorable.

In our previous work [2], we introduced a simple method to combine position features in a VQ framework. Position features were directly incorporated into the training of codebooks and the weighting for the color and position features was determined empirically. It was found that by using position features for retrieval, there was an overall modest improvement in performance for a minor increase in complexity (feature vector dimension increase by 2), while at the same time providing "insurance" where it makes little difference for many classes but can yield significant improvements for some.

In this paper, we present two new methods for incorporating position features into the retrieval framework. The first method trains codebooks for color and position features separately, thereby not requiring feature weighting during the training phase. Our second method partitions images into 9 regions and trains VQ codebooks for each on them. While maintaining a competitive level of retrieval accuracy, it works with greatly reduced complexity (lower than our previous work and an order of magnitude or more lower than previous methods employing Gaussian mixture models). This second method can be viewed as related to the work by Stricker and Dimai [9] where images were represented by fuzzy, partially overlapping regions; feature vectors consisting of the first three color moments in the HSV color space are extracted from each region and retrieval is performed by essentially computing the sum of weighted differences between color feature moments.

2 Computing Image Similarity

In our methods we use VQ codebooks as generative models for the image observations. Vector quantization is a technique generally used for signal compression [6]. Given a set of training vectors, a training algorithm determines a set of codewords that constitute a codebook. Compression is achieved in representing a source vector by transmitting the index of the codeword that is closest (in the MSE sense) to it.

We use VQ codebooks to represent images by training a codebook for every database image. Image similarity is determined by how well a given database image's codebook encodes a query image. Specifically, given a query image A, and a database image B, our system computes a simple retrieval distance which is defined to be the mean squared error when A is encoded with B's codebook:

$$d(A,B) = \frac{1}{N}\sum_{i=1}^{N} \operatorname*{argmin}_{j} \|(a_i - b_j)\|_{\mathbf{W}} \tag{1}$$

where $a_i, b_j \in \mathbb{R}^k$, $\{b_j\}_{j=1}^{M}$ is the codebook for image B, $\{a_i\}_{i=1}^{N}$ are the feature vectors from query image A, $\mathbf{W} \in \mathbb{R}^{k\times k}$ is a weight matrix and $\|x-y\|_{\mathbf{W}} = (x-y)^t\mathbf{W}(x-y)$.

In our previous work [2], we extended this basic approach by incorporating the XY-coordinates into the feature vectors. Full search VQ codebooks were trained jointly on color and position features using the GLA and the feature weightings and codebook sizes were determined empirically. Although this approach performed comparably to previous methods employing VQ at lower complexity, it is still more complex than the methods we will present here.

2.1 Separate Training with Color-Position Codebooks

It is not clear what effect joint training of disparate features has on retrieval accuracy, thus in order to avoid the feature weighting problem during the training phase, color and position features can be trained separately. Color-position codebooks consist of separate codebooks for different feature types. We use the Lloyd clustering algorithm to first train a color only codebook. The feature vectors are then mapped, using only their color components, to entries in this codebook. Using the position components of these feature vectors, a position codebook is trained for each color codeword (so each color codeword has a position codebook associated with it). When encoding a given a query vector, the best color position match is the one that jointly minimizes the weighted color and position distortion:

$$d(A,B) = \frac{1}{N}\sum_{i=1}^{N} \operatorname*{argmin}_{j,k} \left\{ \|(a_i^{(c)} - b_j^{(p)})\|_{\mathbf{W}^{(c)}} + \|(a_i^{(p)} - b_{jk}^{(p)})\|_{\mathbf{W}^{(p)}} \right\} \tag{2}$$

where $x_i^{(c)}$ and $x_i^{(p)}$ are the color and position features respectively for feature vector x, $\mathbf{W}^{(c)}$ and $\mathbf{W}^{(p)}$ are the color and position weights respectively and $k \in \{1, \ldots M_j^{(p)}\}$ with $M_j^{(p)}$ the number of position codewords associated with color j. In terms of complexity, this method is cheaper than having an unstructured VQ codebook of size $M_1 M_2$ with M_1 colors and M_2 positions per color.

2.2 Region VQ

Our second method partitions the image into regions and computes similarity between corresponding regions. Separate VQ codebooks trained on color feature vectors are

associated with each region of each database image. To compare a query image with a database image, the total similarity score is the sum of the scores for each region, where the score for a region is the MSE when the query image features for that region are encoded using a database image's VQ codebook for that region:

$$d(A,B) = \sum_{i=1}^{r} \frac{w_i}{|R_i|} \sum_{j=1}^{|R_i|} \underset{k}{\operatorname{argmin}} \|a_{ij} - b_{ik}\| \tag{3}$$

where r is the number of regions in the images, $|R_i|$ is the number of feature vectors in region R_i, $\{b_{ik}\}_{k=1}^{M_i}$ is image B's (color only) codebook of size M_i for region R_i and w_i is the weight to be assigned to region R_i. The size of the region codebook M_i is varied by query and region by setting a parameter called the *query threshold*. This parameter will determine how many codewords to use when encoding a query. Since the database codebook size needs to be varied based on the query image statistics, we use tree-structured codebooks (TSVQ) for the database images. After partitioning the image into regions, we train large TSVQ codebooks by successively splitting nodes with the largest distortion until we arrive at the desired number of leaf codewords. The following steps summarize the querying process for a query image Q, database image D and query threshold T.

- Extract N regions from the query $Q = \{q_i\}_{i=1}^{N}$ and train TSVQ codebooks for each region $C(q_i)$ such that the average distortion for each region is less than the query threshold T.
- Prune database image's region codebooks $C(d_i)$ to size $|C(q_i)|$. The pruning algorithm merges siblings u and v with parent w such that $D(w)-(D(u)+D(v))$ is minimal amongst all siblings. Where $D(n)$ is the (training) distortion at node n and is stored in the codebook.
- Encode query features using this pruned codebook using full search VQ as defined in Eq. 3.

Figure 1 shows two sample database images; to the right of each image is a table showing the number of entires in the codebook for the corresponding region. A key advantage of this approach is additional reduction in complexity. Position information is implied and not longer explicitly part of the computation. In addition, when a region is processed, smaller codebooks are being searched. In fact, as we shall see, setting the query threshold so that average codebook size is less than 2 yields performance in practice that is still comparable (slightly better) than previous methods.

2.3 MDIR

We compare the two methods presented in this work with the minimum distortion image retrieval (MDIR) method of Jeong and Gray, which is based on Gauss mixture VQ, and is currently the most competitive probabilistic image retrieval method [8]. MDIR uses finite Gaussian mixture models (GMM) trained using GMVQ clustering

2	2	1
19	15	4
4	4	3

3	9	7
12	18	10
10	5	6

Figure 1: Sample database images with corresponding region codebook sizes for a query threshold of 1500.

(Gray [7]) to represent images. The best match for a query image is the database GMM that minimizes the distortion when the query features are encoded using that GMM:

$$d(A, B) = \sum_{i=1}^{N} \operatorname*{argmin}_{j} \rho(a_i, \pi_j, g_j) \tag{4}$$

where $\{\pi_j, g_j = \mathcal{N}(\mu_j, \Sigma_j)\}_{j=1}^{M}$ is the finite Gaussian mixture model with priors π_j trained from image B and $\rho(a_i, \pi_j, g_j) = (d_{\mathrm{LL}}(a_i, g_j) - \ln \pi_j)$ is the penalized log-likelihood.

3 Experiments

For our experiments, we used the standard COREL database (also used by Wang et al. [12], Jeong and Gray [8]), which consists of 1500 JPEG images, organized into 15 classes of 100 images each. Database images are either 256×384 or 384×256 in size. For convenience, the images were cropped to a central 256×256 region and then scaled down to 128×128 using the Convert[1] program. Figure 2 shows sample images from the database.

3.1 Image features

We employed relatively simple features for experiments. Database images were transformed from the RGB to the nonlinear CIE-LUV color space in which the Euclidean distance between color vectors corresponds more closely with human perception (Wyszecki and Stiles [13]). We used a window of size 2×2 for feature extraction by sliding it across the image in a raster tiling fashion. Each feature vector comprises of the mean and the variance for each color channel within that block along with its position (XY-coordinates). The result is an 8-dimensional feature vector: $(\mu_L, \mu_U, \mu_V, \sigma 2_L, \sigma 2_U, \sigma 2_V, x, y)^t$.

[1] `http://www.imagemagick.org/`

Figure 2: Samples database images from each class in raster order: *Africans, Elephants, Caves, Beach, Roses, Postcards, Architecture, Horses, Sunsets, Buses, Mountains, Tigers, Dinosaurs, Foods, Women.*

3.2 Results

The same set of 210 query images as used in Jeong and Gray [8] and in our previous work [2] was used in all experiments reported here. Standard *precision* vs. *recall* curves were used as performance metrics. Precision is defined as the fraction of images retrieved that are relevant (belong to the same class as the query) and recall is the fraction of the relevant images that are retrieved: $p = a/(a+b)$ and $r = a/(a+c)$ where a is the number of relevant images that are retrieved, b, the number of irrelevant items retrieved and c, the number of relevant items not retrieved. Precision and recall are shown on a single graph so that they may show the change in precision as the recall increases. Since the precision typically drops as the recall increases, a retrieval system is said to be more effective if it has higher precision at the same recall values.

To test our first method using color-position codebooks, we trained codebooks with 8 colors and 8 positions per color. Figure 3 shows the precision recall plots of color-position codebooks with our previous work of jointly trained codebooks consisting of a total of 8 codewords and MDIR also consisting of 8 component Gaussian mixtures. It can be seen from this plot that in the recall range of 10% to 50% (arguably the most important region for image retrieval systems), color-position codebooks improves precision by almost 20% at best. The influence of position weighting is shown in table 1. Here we show the *total precision*, which is simply the precision sum at all recall values: $P_t = \sum_{r=1}^{100} P_r$ where P_r is the precision at recall r, for various position weights. We have used simple scalar weights of type $w\mathbf{I}$, with $\mathbf{I}$ being the identity matrix of the appropriate dimensions for position features. Although the various position weightings fractionally improve total precision, the method is still

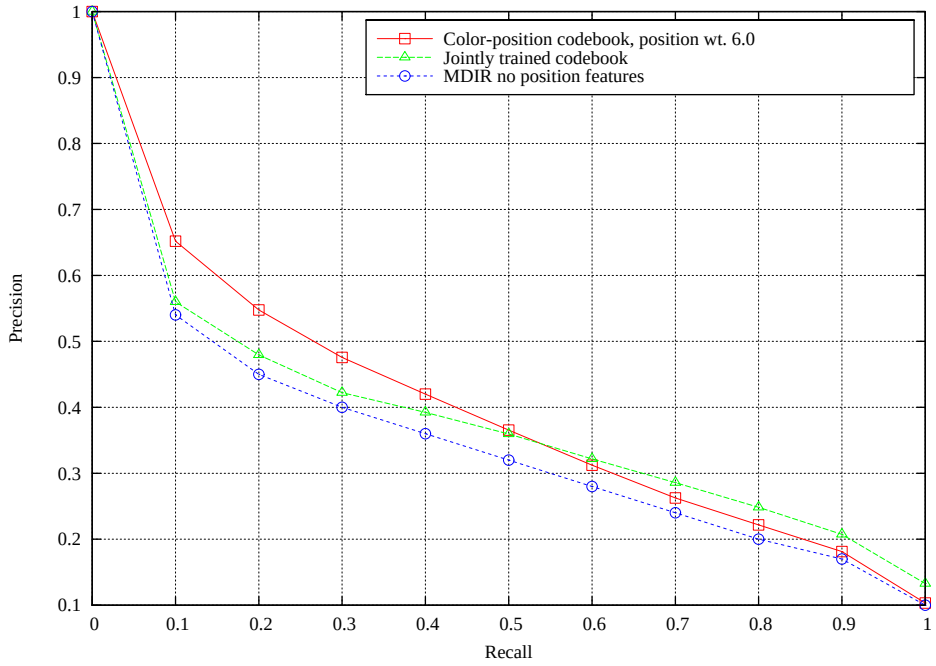

Figure 3: Precision vs. Recall for color-position codebooks.

advantageous since the influence of weightings is removed from the codebook training phase.

For region VQ experiments, we partitioned each database image into 9 regions where the 4 corner regions were of size 43×43. For each region, a TSVQ codebook was trained with 32 leaf codewords. Tests were performed at various query thresh-

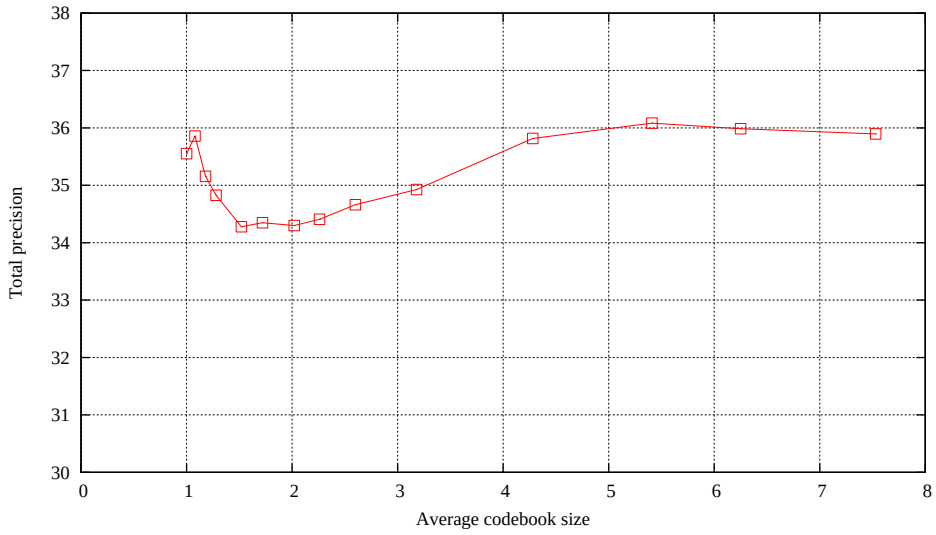

Figure 4: Total precision vs. Average codebook size for region VQ

Color Weight	Position Weight	Total Precision
1	6	38.90
1	5	38.88
1	7	38.85
1	8	38.73
1	10	38.40
1	2	38.03

Table 1: Total precision for color-position codebook with 8 colors and 8 positions per color with different positional weightings.

olds. Figure 4 plots the total precision verses the average region codebook size which is computed as the average codebook size for all 210 queries for all 9 regions. The best performance occurs at an average codebook size of 5.41 corresponding to a query threshold of 1500. Figure 5 shows the precision recall performance of region VQ compared with jointly trained codebooks and MDIR. Region VQ with average codebook size of 1.08 performs almost as well as with average codebook size 5.41. However notice that over the recall range of 10%-40%, the latter improves precision by around 3%. Also, region VQ for both average codebook sizes shows an improvement both, in terms of precision and complexity over the other methods for recall up to 40%.

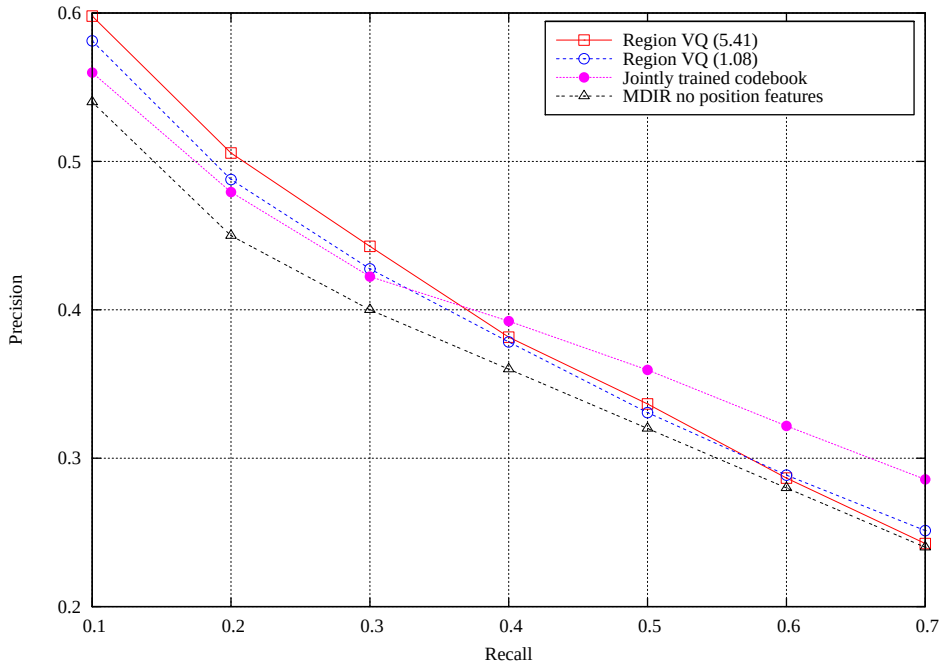

Figure 5: Precision vs. Recall for region VQ. The numbers in parenthesis are the average codebook sizes

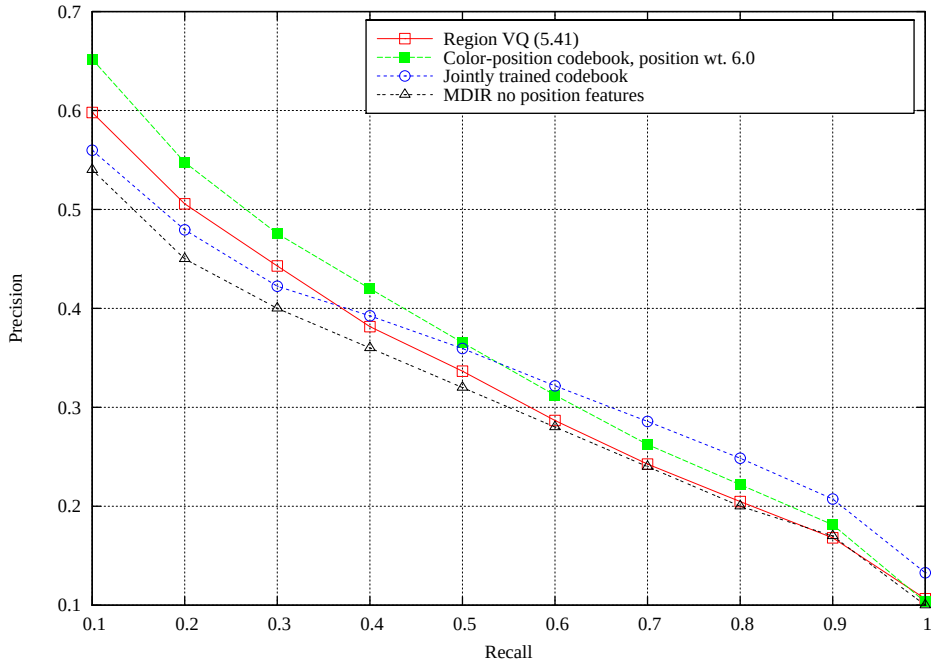

Figure 6: Precision vs. Recall plots

3.3 Complexity

In this section we perform a simple complexity analysis of the various methods referred to in this work. As an example, consider the cost (multiplications) of searching for the best codeword for a given 8-dimensional source vector. A color-position codebook with 8 colors and 8 positions per color, requires $8(6 + 2 \times 8) = 176$ operations while an equivalent jointly trained codebook with 64, codewords requires $8 \times 64 = 512$ operations. Region VQ affords even more reduction in complexity. Indeed, for an average of 8 codewords per region, $6 \times 8 = 48$ multiplications suffice. On the other hand, MDIR with 64 components per Gaussian mixture and full covariances requires $8 \times 8 \times 64 = 4096$ multiplications.

4 Conclusion and Future Work

We have presented two low complexity alternatives for incorporating position features into the VQ based image retrieval framework. Both methods perform comparably or better than previous methods while offering significantly lower complexity. Figure 6 shows the precision recall plots comparing these We see here that for recall values up to 50%, the new methods show improved precision, up to 20% at best. Further investigation in terms of region weighting and soft partitioning of images will help better understand region based VQ methods. Also, it would be interesting to replace region VQ codebooks with Gaussian mixtures especially when the number of codewords is very low.

References

[1] Ajay H. Daptardar and James A. Storer. Content-based image retrieval via vector quantization. In *ISVC*, volume 3804 of *Lecture Notes in Computer Science*, pages 502–509. Springer, 2005.

[2] Ajay H. Daptardar and James A. Storer. Reduced complexity content-based image retrieval using vector quantization. In *Data Compression Conference*, pages 342–351. IEEE Computer Society, 2006.

[3] J. P. Eakins and M. E. Graham. Content-based image retrieval. Technical report, JISC Technology Applications Programme, 1999.

[4] Christos Faloutsos, Ron Barber, Myron Flickner, Jim Hafner, Wayne Niblack, Dragutin Petkovic, and William Equitz. Efficient and effective querying by image content. *Journal of Intelligent Information Systems*, 3(3-4):231–262, 1994.

[5] Myron Flickner, Harpreet Sawhney, Wayne Niblack, Jonathan Ashley, Qian Huang, Byron Dom, Monika Gorkani, Jim Hafner, Denis Lee, Dragutin Petkovic, David Steele, and Peter Yanker. Query by image and video content: The QBIC system. *IEEE Computer*, 28(9):23–32, 1995.

[6] Allen Gersho and Robert M. Gray. *Vector Quantization and Signal Compression*. Kluwer Academic Publishers, 1992.

[7] Robert M. Gray. Gauss mixture vector quantization. In *Proceedings of IEEE ICASSP*, volume 3, pages 1769–1772, 2001.

[8] Sangoh Jeong and Robert M. Gray. Minimum distortion color image retrieval based on Lloyd-clustered Gauss mixtures. In *Data Compression Conference*, pages 279–288. IEEE Computer Society, 2005.

[9] Markus Stricker and Alexander Dimai. Color indexing with weak spatial constraints. In *Storage and Retrieval for Image and Video Databases IV*, volume 2670, February 1996.

[10] Michael J. Swain and Dana H. Ballard. Color indexing. *International Journal of Computer Vision*, 7(1):11–32, 1991.

[11] Nuno Vasconcelos. Minimum probability of error image retrieval. *IEEE Transactions on Signal Processing*, 52(8):2322–2336, 2004.

[12] James. Z. Wang, Jia Li, and Gio Wiederhold. Simplicity: Semantics-sensitive integrated matching for picture libraries. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(9):947–963, 2001.

[13] Günther Wyszecki and W.S. Stiles. *Color Science: Concepts and Methods, Quantitative Data and Formulae*. Wiley-Interscience, 2 edition, 2000.