Towards Versatile Document Analysis Systems

Henry S. Baird and Matthew R. Casey

Computer Science & Engineering Dept
Lehigh University
19 Memorial Dr West
Bethlehem, PA 18017 USA
E-mail: baird@cse.lehigh.edu, mrc8@lehigh.edu

Abstract. The research goal of highly versatile document analysis systems, capable of performing useful functions on the great majority of document images, seems to be receding, even in the face of decades of research. One family of nearly universally applicable capabilities includes document image content extraction tools able to locate regions containing handwriting, machine-print text, graphics, line-art, logos, photographs, noise, etc. To solve this problem in its full generality requires coping with a vast diversity of document and image types. The severity of the methodological problems is suggested by the lack of agreement within the R&D community on even what is meant by a representative set of samples in this context. Even when this is agreed, it is often not clear how sufficiently large sets for training and testing can be collected and ground truthed. Perhaps this can be alleviated by discovering a principled way to amplify sample sets using synthetic variations. We will then need classification methodologies capable of learning automatically from these huge sample sets in spite of their poorly parameterized—or unparameterizable—distributions. Perhaps fast expected-time approximate k-nearest neighbors classifiers are a good solution, even if they tend to require enormous data structures: hashed k-d trees seem promising. We discuss these issues and report recent progress towards their resolution.

Keywords: *versatile document analysis systems, DAS methodology, document image content extraction, classification, k Nearest Neighbors, k-d trees, CART, spatial data structures, computational geometry, hashing*

1 Introduction

Computer vision systems, and document analysis systems in particular, are notoriously either overspecialized and fragile, or ruinously expensive (*e.g.* national postal code readers)[1]. The research goal of highly versatile document analysis systems, capable of performing useful functions across the great majority of document images, seems to have receded, even in the face of decades of research [2][3]. Certainly users—*e.g.* in the digital libraries, web search, and intelligence communities—avidly desire such a technology. But is this goal too ambitious: a mirage,

[0] [To appear in *Proc., 7th IAPR Document Analysis Workshop (DAS'06)*, Nelson, New Zealand, February 12-15, 2006.]

so ill-defined or open-ended that it is illusory? Or, can some fresh research agenda accelerate our approach to it? This paper takes seriously the potential of such an agenda.

Thus we aim at a *versatility first* research strategy: that is, we attempt to invent document image analysis capabilities that will work reliably across the broadest possible range of cases. This diversity embraces documents and images that: are carefully prepared as well as hastily sketched; are written in many languages, typefaces, typesizes, and writing styles; obey a myriad of geometric and logical layout conventions; possess a wide range of printing and imaging qualities; are expressed as full color, grey-level, and black-and-white; may be acquired by geometrically accurate flat-bed scanners, or snapped with hand-held cameras under accidental lighting conditions; are of any size or resolution (digitizing spatial sampling rate); and are presented in many image file formats (TIFF, JPEG, PNG, etc), both lossless and lossy.

Now not every useful functionality is called for on all or even most documents: *e.g.* word-spotting in Amharic is surely a niche application which will be perfected (if it ever is) by special-purpose engineering. So for the purposes of this paper we will restrict our attention to a set of basic capabilities that are, arguably, most broadly useful in that they have been applied very early in the document-processing pipeline. These are the *document image content extraction* tools, able to locate and characterize regions containing handwriting, machine-print text, graphics, line-art, logos, photographs, noise, and other "content types." By "locate" we mean describe the regions in the image where each content type dominates. By "characterize' we mean report estimates of a few basic properties of the regions: *e.g.* for machine-print, report skew angle, number of lines of text, number of words, and perhaps type size and type family. Although high accuracy is always appreciated, for broadly applicable tools such as these are, a lower threshold of competency may be sufficient: in pariticulat, trhe confidence of a decision or of the value of a property may be as important as accuracy to downstream processing.

Even for this narrowly characterized set of tasks, the severity of our methodological problems is suggested by the lack of agreement within the R&D community on what is meant by a representative set of samples. The gold standard for this in our field is a manually collected set of samples with a published rationale for selection and a detailed protocol for assigning ground truth (*e.g.* UWash CD-ROM DBs[4]). Certainly these methodologies are essential: they have assured long and useful lives for their resulting data bases, and have strongly benefited DAS research. But their costs are dauntingly high—approximately $2M to compile the 3-CDROM UWash DBs—and they are as well known for their omissions as their coverage—*e.g.* the UWash DBs contain only English and Japanese documents. If a document image is seen, later, what does it mean to claim that the training set was representative of it? If we can't make this claim, how great an indictment is it if a classifier, trained on that set, fails on this new sample? What do we say if the user claims that he or she expected our tool to succeed on this sample, because "it was like the training data"? How can sample sets with far broader coverage be affordably collected and ground-truthed?

In Section 8 we propose that this obstacle can be alleviated, if not overcome, by developing a principled way to amplify sample sets using synthetic variations. A particular form of this, which we call *model-based interpolation* may combine the authenticity of "real" samples with the broad and systematic coverage of synthetic samples.

The data sets we need to provide coverage will certainly be vast: far larger than our current databases. To make good use of them, we will need classification methodologies capable of learning automatically from such huge sample sets in spite of their poorly parameterized—or unparameterizable—distributions. In Section 4 we propose that it may be possible, due to recent advances in memory, disk storage, and data structures technology, to design fast expected-time approximate k-nearest neighbors classifiers. The key insight here is that fast *expected time* performance is a good match for document image analysis problems, since among pages images within a document, and even more strongly among pixels within a single image, we can expect to see

the effects of style consistency[5][6]—what Prateek Sarkar called *isogeny*—so that we can hope that the enormous data structures that high-dimensional search trees (*e.g.* non-adaptive k-d trees) tend to require may be manageable using modern paging techniques.

2 A Research Agenda for Versatile Document Analysis

We are investigating strongly versatile algorithms for the **document image content extraction** problem:

> *Given* an image of a document,
> *find* regions containing handwriting, machine-print text, graphics, line-art, logos, photographs, noise, etc.

To solve this problem in full generality—given the great diversity of document and image types—we have decided to emphasize, in our research strategy:

1. *versatility first*: concentrate first on designing methods that work across the broadest possible variety of cases (document and image types);
2. *voracious classifiers*: trainable on billions of samples in reasonable time;
3. *extremely high speed classification*: ideally, nearly at I/O rates (as fast as the images can be read);
4. *amplification*: use real ground-truthed training samples as 'seeds' for massive synthetic generation of pseudorandomly perturbed samples for use in supplementary training;
5. *confidence before accuracy*: good estimates of the confidence of decisions is important; high accuracies are desirable but even modest accuracy can be useful;
6. *near-infinite space*: *i.e.* design for best performance in a near-term future when main memory will be orders of magnitude larger and faster than today; and
7. *data-driven design*: we won't invest much effort in making what are, ultimately, arbitrary engineering decisions such as choice of preprocessors and features; instead, we'll try to allow the training data to determine these automatically as far we can;

3 Document Image Content

Types of document images we wish to process include color, grey-level, and black-and-white; also, any size or resolution (digitizing spatial sampling rate); and in any of a wide range of file formats (TIFF, JPEG, PNG, etc). We choose to convert all image file formats in a three-channel color PNG file in the HSL (Hue, Saturation, and Luminance) color space; this has the advantage that black-and-white and greylevel images convert into HSL images with fixed values for Hue and Saturation permitting easier comparison with color images using simple metrics.

We are gathering sample page images containing the following types of content: handwriting, machine print, line art, photos, math notation, maps, engineering drawings, chemical drawings, "junk" (e.g. margin and gutter noise),and blank. Sources for these include: the Univ. Washington CD-ROM Data Bases, the UNLV Data Sets, various Library of Congress digital library sites such as the American Memory Project, the ISI Kolkata mathematics notation collection, and Lehigh University digital library sites[1] We can exploit any pre-existing ground-truth metadata with rectangular zones; for other images we have developed an interactive ground-truthing tool. We have begun a systematic statistical sample and analysis of the presence of these content types within these data bases.

[1] Details are reported in [7].

4 Statistical Framework for Content Classification

To avoid overspecialization, we choose to classify individual pixels, not extended regions of arbitrarily chosen size and shape. Thus each training and test sample is a pixel in a document image, and each pixel owns d features which are scalar properties extracted by image processing in the vicinity of that pixel.

Our statistical sample space (the "universe") $U = \mathbf{R}^d$, the multidimensional reals. If all we are told about a classification problem is a training set—if we lack, for example, analytic models for the processes that generated the samples—then no reliable and rapidly computable way is known, grounded in pattern recognition theory, for automatically choosing a small number of sufficiently discriminative features that will support highly accurate classification. Thus in practice we will be forced to explore, in a trial and error way, a large space of promising features suggested by insight, experiment, and the literature. Given a set of features, methods are known to select a good subset of them: but it is far easier to propose new features than to select the best ones. For these reasons we expect that the number of features will be large, say $d \approx 100$. A member of the sample space is called $\mathbf{x}, \mathbf{y}, \mathbf{z}$, etc. All samples belong to one of m classes $\mathbf{C} = \{c_j\}_{j=1,\ldots,m}$. The number of content type classes m will, we expect, be ≈ 10. Within the $\mathbf{R}^d$ space we choose to use the Infinity norm (or L_∞ norm) $\|\mathbf{x} - \mathbf{y}\|_\infty \equiv max_{i=1}^{d}\{|x_i - y_i|\}$ to capture the notion of feature similarity. Many pattern recognition researchers believe that the choice of metric has less influence on the accuracy of a classifier than the size and representativeness of the training data[8]. Training data consists of a set of samples $\mathbf{T} \subset \mathbf{R}^d$ labeled with their classes. We expect the number of training samples $n \equiv |\mathbf{T}| \approx 1$ billion. We first discuss the problem of classifying a single test sample **x** but eventually we expect to batch-classify large sets **X** of test samples. Note that in our context $c \ll d \ll n$: this will seriously constrain our engineering options, as we will see.

5 A Nearly Ideal Classifier in this Context

Classification problem

Given $\mathbf{T}$, and a previously unseen sample **x**,
find the most probable class c for **x**.

Adopting a Bayesian approach, we choose $c_{max} = argmax_{c_j \in \mathbf{C}}\{P(c_j|\mathbf{x}, \mathbf{T})\}$ where $P(c_j|\mathbf{x}, \mathbf{T})$ denotes the posterior probability of class c_j given the new observation **x** and the prior knowledge expressed in training data $\mathbf{T}$.The *k-nearest neighbors*(kNN) algorithm is of course:

K Nearest Neighbors algorithm

INPUT: $\mathbf{T}$, $k \in \mathbf{N}^+$, and **x**
OUTPUT: $c \in \mathbf{C}$
Step 1. within $\mathbf{T}$, find k nearest neighbors of **x**
that is, find a set $Y \subset \mathbf{T}$, $|Y| = k$, s.t.
$\forall \mathbf{y} \in Y \ \forall \mathbf{z} \in T - Y \ \|\mathbf{x} - \mathbf{y}\| \leq \|\mathbf{x} - \mathbf{z}\|$
Step 2. within Y, use the ground-truth classes assigned to samples in $\mathbf{T}$ find the most frequently occurring class, breaking ties at random.

For any particular set of features, assuming that the training data is representative, the kNN algorithm enjoys a valuable theoretical property: as the size of the training set increases, its error rate approaches to no worse than twice the Bayes error[9], which is the minimum achievable by any classifier given the same information. This remarkable performance is due, fundamentally, to its

not committing to a specific parametric model for the per-class distributions. This algorithm also generalizes directly to more than two classes (unlike several competing methods such as neural nets and classification trees). Finally, it has has often been competitive in accuracy (if rarely in speed) with more recently developed classifiers including support-vector machines. For these reasons, we consider kNN would be a nearly ideal classifier for this problem, were it were not for the high cost in both time and space of naive implementations and the difficulty of crafting algorithms that approximate its performance in high dimensions.

We have implemented this (for $k = 5$) and, in early small scale experiments, have seen it yield pixel classification rates on the content extraction problem greater than 98% correct. Interestingly, even when per-pixel classification error is as high as 23%, the principal regions and their dominant content types are evident to the eye and (we expect) can be extracted robustly by simple image post-processing[7].

6 Approximations to kNN

Radius Search problem

Given $\mathbf{T}$, $r \in \mathbf{R}$, and $\mathbf{x}$,
find all points of $\mathbf{T}$ within radius r of $\mathbf{x}$: that is, the set $Y_r \equiv \{ \mathbf{y} \in \mathbf{T} \ s.t. \ \|\mathbf{x}-\mathbf{y}\| \leq r \}$

If $r \ll s$, we can expect that finding Y_r will be easier than finding Y_s. (Of course, generally $|Y_r| < |Y_s|$ also, in which case reporting the result of the search will take *longer*. But we choose to neglect runtimes that depend on the size of the output, for reasons that will become clear later.) If $|Y_r| = k$, then radius search is equivalent to kNN. If $|Y_r| \approx k$, then we will say that radius search *approximates* kNN. (Of course, this notion of approximation does not address the key issue, which is how much less accurate radius search classification is than kNN classification.)

It might be cheaper to compute approximations to radius search.

Nested Radius Search problem

Given $\mathbf{T}$, $\{r_i\}_{i=1,\ldots,s} \in \mathbf{R}$, $r_1 < r_2 < \ldots < r_s$, and $\mathbf{x}$,
find sets $\subset \mathbf{T}$ within radii r_i of $\mathbf{x}$: that is,
$Y_{r_i} \equiv \{ \mathbf{y} \in \mathbf{T} \ s.t. \ \|\mathbf{x} - \mathbf{y}\| \leq r_i \}$

Note that $Y_{r_1} \subseteq Y_{r_2} \subseteq \ldots \subseteq Y_{r_s}$. If $r_k < r_l$, then it may be possible, through judicious choice of data structures and algorithms, to compute Y_{r_l} faster than Y_{r_k}.

Approximate Radius Search problem

Given $\mathbf{T}$, $r \in \mathbf{R}$, $\epsilon \in \mathbf{R}$ and $\mathbf{x}$,
find sets Y^L and $Y^U \subset \mathbf{T}$ such that
$Y_{r-\epsilon} \subseteq Y^L \subseteq Y_r \subseteq Y^U \subseteq Y_{r+\epsilon}$

That is, Y^L and Y^U approximate Y_r within ϵ. As $\epsilon \to 0$, $Y^L \to Y_r$ (from below) and $Y^U \to Y_r$ (from above), and so Approximate Radius Search converges to Radius Search. When ϵ is large, it should be easier to solve Approximate Radius Search than to solve Radius Search.

How can Approximate Radius Search be used for classification? Well, if it happens that $|Y^L| = |Y^U|$, then $Y^L = Y^U = Y_r$ and, exactly as in kNN, we can choose the most frequently occurring class in that set. Lets call the frequency of occurrence of each class c_j in Y^L "f_j^L" and similarly, in Y^U, "f_j^U". Let the most frequently occurring classes in these sets be $maxc^L$ and $maxc^U$; if these happen to be the same class, we will of course choose that class. But if they differ, then we have several policy choices:

- we can compute average frequencies for each c_j, *e.g.* $f_j^{aver} \equiv (f_j^L + f_j^U)/2$, and then choose the class with the highest average frequency;
- we can use the value k from kNN to assist in the decision, for example, if $|Y^L|$ is closer to k than Y^U, we choose the most frequently occurring class in Y^L (and vice versa).

Thus Approximate Radius Search can be used as a basis for classification and offers a range of engineering tradeoffs between accuracy and speed.

7 Approximate Radius Search under the Infinity Norm

With the Infinity Norm as our metric, then Radius Search becomes a multidimensional range search (or "region query") in which the search regions are hypercubes of radius r centered on the query sample $\mathbf{x}$. Multidimensional range searching is an intensively explored topic[10][11] by researchers in the computational geometry, machine learning, pattern recognition, and graphics research communities. However, special requirements of pattern classification may lead us in fresh directions. For example, in the literature, range searching is commonly discussed as a generalization of *point searching*, which in turn is characterized as a generalization of dictionaries to multidimensional data. Dictionaries are classically designed to support three basic operations: *insert* a single data item, *find* an item (returning its associated metadata, or returning 'not found'), and *delete* an item. Data structures and algorithms for dictionaries are thus *dynamic*: designed to process an arbitrary sequence of these three operations efficiently

By contrast, in classification, insertions are performed all at once in an offline batch operation (the "training stage") in which all the data (training samples) are simultaneously available. Deletion never occurs (although we may choose to prune ("edit) them or summarize sets of data points statistically rather than storing all of them explicitly). Thus the data structure is *static*. Finding is the only operation which is online and must run fast; and it may also be possible to batch a large set of find operations and so exploit the similarity of sequences of queries for points that are isogenous. Thus much of the literature on dynamic multidimensional search is not directly relevant to classification, and may benefit from a fresh perspective.

Let us review techniques for multidimensional search that may assist us in choosing, adapting, or crafting new data structures and algorithms suitable for classification.

7.1 Adaptive k-d Trees

One multidimensional search technique with broad applicability is Bentley's *k*-d trees[12]. The variant of k-d trees most relevant to classification seems to be the *adaptive k*-d tree[13]. Briefly, these partition the set of points recursively in stages: at each stage one of the partitions is divided into two subpartitions; we will assume that it is possible to choose cuts that achieve *balance*, that is, to divide into subpartitions containing roughly the same number of points. Division is by cutting along one of the dimensions $i \in \{1, \ldots, d\}$, *i.e.* by choosing a threshold value and assigning each of the partition's points $\mathbf{x}$ to subpartition (a) or (b) according to whether its $\mathbf{x}_i$ component value is (a) less than, or (b) greater than or equal to the threshold. (In some implementations, the threshold corresponds to a component value for one of the points, which is then stored in the interior node of the search tree; but will we assume here that all points are stored in leaf nodes). Generally, at each stage a different dimension is cut: one simple strategy is to cycle (and if necessary recyle) through the dimensions in a fixed order; another strategy is to cut the currently most populous partition. Cutting proceeds until all partitions contain few enough points to invite a final fast sequential search. The final partitions are generally hyperrectangles (not always hypercubes) with orthogonal sides (parallel to the coordinate axes of $\mathbf{R}^d$).

Balancing each cut ensures that find operations execute in $\Theta(\log n)$ time in the worst case. Consistent with a guarantee of such logarithmic-time finds, Bentley's k-d tree construction achieves an asymptotically minimum number of cuts and thus a minimum number of partitions (close to n/p, where p is the number of points in the final partitions; note that, in our context where $p = 10$ this is still huge, $\approx 10^8$). The threshold value chosen for each cut depends on the distribution of points within the partition to be cut, so the thresholds are not *independently predictable*: that is, none (after the first) can be computed without knowledge of the cut thresholds in some earlier stages. Further, given a previously unseen **x** it is not possible to compute the d upper and lower bounds of its k-d hyperrectangle (within which it lies) without traversing the k-d tree. Thus locating **x**'s k-d hyperrectangle *requires* $\Theta(\ \log n\ )$ time.

The pruning power of k-d trees speeds up range searches. Given a query point **x** and a radius r, defining a search hypercube, it is straightforward to generalize the find algorithm to explore all k-d tree nodes whose subtrees overlap the search hypercube. The asymptotic runtime of such variants has been studied: [14] reports that the worst-case number of tree nodes explored is $\Theta(\ d\, n^{1-1/d}\ )$. In our context, this may (or may not) promise much improvement over brute-force search, since (neglecting the multiplicative constant) $d\, n^{1-1/d} \approx 100(10^9)^{0.99} = 100(10^{8.91}) = 10^{10.91} \gg 10^8 \approx n$. Of course this may be a pessimistic bound for several reasons. Whether or not k-d tree range searches are efficient for classification may ultimately be decidable only by experiment.

7.2 Multidimensional Tries

An alternative data structure that assists multidimensional search is a recursive partitioning of the space using *fixed cuts*. Suppose that for each dimension $i \in \{1, \ldots, d\}$, lower and upper bounds L_i and U_i on the values of the components $\mathbf{x}_i$ are known. Then one may choose to place cuts at midpoints of these ranges, *i.e.* at $(L_i + U_i)/2$, and later, when cutting those partitions, recursively cut at the midpoint of the (now smaller) ranges. A search trie can be constructed in a manner exactly analogous to k-d trees with the exception that the distribution of the data is ignored in choosing cut thresholds. It will no longer be possible to guarantee balanced cuts and thus most of the time and space optimality properties of k-d trees are lost. However, there are gains: for example, the values of the cut thresholds can be predicted (they all are of course predetermined by $\{L_i, U_i\}_{i=1,\ldots,d}$). As a consequence, if the total number of cuts r is known, the hyperrectangle within which any query (test) sample **x** lies can be computed in $\Omega(r)$ time, and in some realistic models of computation in $\Omega(1)$ time—in either case, extremely fast, faster than computing a single point-to-point distance using the metric. We will call these recursive midpoint-cut trie hyperrectangles *partitions* (they are often called 'bins' or 'cells' in the kNN literature).

Bit-Interleaving Partitions resulting from tries as above can be addressed using *bit-interleaving*. Let $< d_k, m_k >_{k=1,\ldots,r}$ be a sequence of cuts, where, at cut k, d_k is the dimension chosen to be cut and m_k is the midpoint of the partition chosen for that cut. Among the partitions of feature space that result, a test sample x will fall in a partition that can be described by a sequence of decisions $b_k = (x_{d_k} > m_k)$ taking on the value False ('0') or True ('1') as x lies below or above the cut respectively. Equivalently, any bit-sequence $< b_k >_k = 1, \ldots, r$ of length r determines the boundaries of some partition, and so can be thought of as an address for it. To summarize: given a test sample x and a sequence of r cuts, we can compute in $\Theta(r)$ time the bit-interleaved address of the partition within which x lies.

Now in this context—as in most if not all naturally occurring image pattern recognition problems—we expect that training and test data have a skewed (nonuniform) distribution in

feature space, and specifically when r is large that only a small fraction of the resulting small partitions will be occupied by *any* training data. If this assumption holds true in practice, only a few distinct bit-interleaved addresses will occur in even a very large training set, and so it may be possible to use a dictionary data structure to manage them. The feasibility of this approach depends crucially of course on how many distinct values of bit-interleaved addresses occur in practice.

Experiments on our document images where $d = 15$, using $n = 10,000,000$ training samples, varying the number of bits $r < 75$ suggest that the number of occupied partitions, as a function of the length of their bit-interleaved address, is asymptotically roughly cubic. For $r = 50$ the absolute number observed was about 2,100,000, which is of course easily manageable by single-stage hashing where the hash table is contained within main memory.

Also, we have systematically tested the accuracy and speedups of hashed kD trees, where $d = 15$ (again), $n = 1,565,695$ training samples and $254,181$ test samples. When a brute-force 5-NN program is run, it achieves a per-pixel correct classification rate of 78% (and the results look very good to the eye); of course, it runs slowly, requiring about 400 billion distance calculations. By using bit-interleaved address of length 40 bits, hashing speeds up the calculation by a factor of 99.7 for a slight drop in per-pixel accuracy, to 68%. The range of tradeoffs between speed-up and accuracy is shown in Figure1.

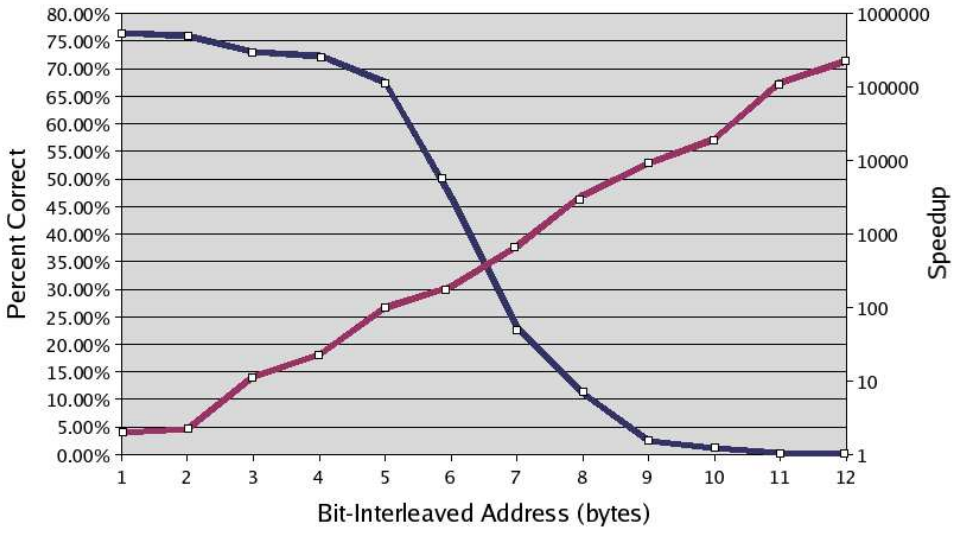

Fig. 1. Tradeoffs between accuracy (percent of pixels correctly classified, labeled on the left scale) and speed-up (factor of decrease in the number of distances computed, compared to brute-force; labeled on the right scale, using a log-scale), as a function of the length of the bit-interleaved address (shown in units of 8-bit bytes).

Accuracy remains high (above 60%) until somewhere between 40 bits (5 bytes) and 48 bits; thereafter it falls rapidly. The speed-up factor improves exponentially across the range.

In the above data, errors are of two types. In the first type, we hash into a cell which contains some training samples, but 5NN does not do as well as expected: this means of course that not all of the five nearest neighbors lie within this hash cell. Errors of the second type occur when the cell contains *no data*: that is, no training points hashed into that cell. Errors of the second type are in the minority as long as the bit-interleaved address is less than about 50 bits; but above that, they rise rapidly and dominate the error count. Clearly these errors will be reduced by the addition of more training data. It is interesting to note that, if we do not count the second type of errors, then accuracy remains high (above 60%) until more than 64 bits are used, for a speed-up of 3401.

The rise and rapid dominance of errors of both types are to be expected when the volume of a hash cell falls below a certain threshold. We can roughly estimate this threshold as follows. We have observed that the distance from a probe point to the farthest of its five nearest neighbors is less than 15 for about 95% of the points. When we use at least 45 bits of bit-interleaved address, we have in effect guaranteed that each of the 15 dimensions have been cut three times, which gives hash cells that are $256/2^3 = 32$ on a side. This is at the boundary of the unpleasant case when the farthest of the five nearest neighbors is likely to fall outside the cell. This rough analysis seems to explain the rapid fall-off of accuracy above 48 bits.

One obvious next step is to dramatically increase the size of the training set and observe how fast the errors of the second type decrease, and the effect also on speed-up.

8 Amplifying Sample Sets

Just a brief note on some ideas for amplifying sets of "real" samples using synthetically generated samples that may answer the objection that synthetic samples are inappropriate for training and testing. The complex processes by which document images are generated—governed by choices of subject, meaning, language, text, illustrations, page layout styles, typefaces, type sizes, printing parameters, image digitization resolution, etc etc—are in many details subject to parameterization by more or less continuously varying numerical values: certainly type size and digitizing gutter widths, line spacing, digitizing resolution, and image degradation parameters; also, less obviously, parameters that characterize typeface families such as stroke width and serif length[15]. Given two "real" images collected and added to a sample set, it is often possible to estimate the values of these parameters. When there is a gap between the two values—e.g. digitizing resolutions of 200 and 400 dpi—we will argue that they span a range of values—including, say 300 dpi—that *could occur* in other sets of real samples. Thus any variation of either image that is generated synthetically by chosing values between the two, within the range that they span, should be admissible as a training or test sample. Given than one such parameter, convex combinations of their values should also be admissable. Synthetic images generated in the way we call *synthetically interpolated*. We have begun the investigation of interpolation tools for this purpose.

9 Discussion and Future Work

However, we may soon face other theoretical and practical obstacles to following this approach in building a fast approximate kNN algorithm. How can we ensure that a partition addressed by a bit-interleaved address contains **all** of the training samples that may fall within the kNN set, or an approximate-radius NN set? It is easy to see that, in the worst case, this extra offline preprocessing (and space requirements) may be exponential in d; even worse, this would require exponential space to store the redundant copies of the test samples. Again this point must be tested empirically.

In a personal communication, Jon Bentley has suggested to us that a hybrid of classification trees (CARTs[16]) and *k*-d trees may provide a circumvention of these potential problems. Specifically, the first few levels of a CART are compressed using a hashing scheme similar to the bit-interleaving trick.; but it is not yet clear how to make this work.

References

1. Pavlidis, T.: Thirty years at the pattern recognition front. In: King-Sun Fu Prize Lecture, 11th ICPR. (2000), address = Barcelona, Spain)
2. Nagy, G., Seth, S.: Modern optical character recognition (1996)
3. Nagy, G.: Twenty years of Document Image Analysis in PAMI. (IEEE Transactions on Pattern Analysis and Machine Intelligence)
4. I. T. Phillips, S.C., Haralick, R.M.: Cd-rom document database standard. (In: Proc., 2nd IAPR ICDAR), pages = 478–483, year = 1993)
5. Sarkar, P., Nagy, G.: Style consistent classification of isogenous patterns. IEEE Trans. on PAMI **27** (2005)
6. Veeramachaneni, S., Nagy, G.: Style context with second order statistics. IEEE Trans. on PAMI **27** (2005)
7. H. S. Baird, M. A. Moll. J. Nonnemaker, M.R.C., Delorenzo, D.L.: Versatile document image content extraction. In: Proc., SPIE/IS&T Document Recognition & Retrieval XII Conf., San Jose, CA (2006)
8. Ho, T.K., Baird, H.S.: Large-scale simulation studies in image pattern recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence **19** (1997) 1067–1079
9. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification, 2nd Edition. Wiley, New York (2001)
10. Samet, H.: The Design and Analysis of Spatial Data Structures. Addison-Wesley, Reading, Massachusetts (1990)
11. Bentley, J.L.: Multidimensional binary search trees used for associative searching. Communications of the ACM **18** (1975) 509–517
12. Bentley, J.L.: Multidimensional binary search trees used for associative searching. Commun. ACM **18** (1975) 509–517
13. Freidman, J.H., Bentley, J.L., Finkel, R.A.: An algorithm for finding best matches in logarithmic expected time. ACM Trans. Math. Softw. **3** (1977) 209–226
14. Lee, D.T., Wong, C.K.: Worst-case analysis for region and partial region searches in multidimensional binary search trees and balanced quad trees. Acta Inf. **9** (1977) 23–29
15. Knuth, D.E.: Computer Modern Typefaces. Addison Wesley, Reading, Massachusetts (1986)
16. Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J.: Classification and Regression Trees. Wadsworth& Brooks/Cole, Pacific Grove, CA (1984)