



<http://www.diva-portal.org>

Postprint

This is the accepted version of a paper presented at *2nd International Workshop on Historical Document Imaging and Processing*.

Citation for the original published paper:

Wahlberg, F., Brun, A. (2013)

Feature space denoising improves word spotting.

In: *Proc. 2nd International Workshop on Historical Document Imaging and Processing* (pp. 59-66).

New York: ACM Press

<http://dx.doi.org/10.1145/2501115.2501118>

N.B. When citing this work, cite the original published paper.

Permanent link to this version:

<http://urn.kb.se/resolve?urn=urn:nbn:se:uu:diva-206930>

Feature Space Denoising Improves Word Spotting

Fredrik Wahlberg
Centre for Image Analysis
Uppsala University
fredrik.wahlberg@it.uu.se

Anders Brun
Centre for Image Analysis
Uppsala University
anders.brun@it.uu.se

ABSTRACT

Some of the sliding window features commonly used in off-line handwritten text recognition are inherently noisy or sensitive to image noise. In this paper, we investigate the effects of several de-noising filters applied in the feature space and not in the image domain. The purpose is to target the intrinsic noise of these features, stemming from the complex shapes of handwritten characters. This noise is present even if the image has been captured without any kind of artefacts or noise. An evaluation, using a public database, is presented showing that the recognition of word-spotting can be improved considerably by using de-noising filters in the feature space.

Categories and Subject Descriptors

I.4.3 [Image processing and computer vision]: Enhancement—*Filtering*

General Terms

Keywords

OCR, handwritten text recognition, filtering

1. INTRODUCTION

In off-line handwritten text recognition of historical documents, the starting point is a photograph of a page of some historical manuscript. The historical texts still in existence were written hundreds (or even thousands) years ago. This creates problems such as degraded ink, rough handling over generations and geometrically distorted parchment due to moisture. Transcribing these historical texts, in part or in full, is immensely valuable to research in several humanist disciplines.

The problem of doing a full recognition of a book is today largely unsolved, with the exception of a small set of books where extensive training data has been created manually. To be able to search and index the treasures in our libraries

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.

HIP '13 August 24 2013, Washington, DC, USA

Copyright is held by the owner/author(s). Publication rights licensed to ACM.

ACM 978-1-4503-2115-0/13/08 ...\$15.00.

<http://dx.doi.org/10.1145/2501115.2501118>

today, word-spotting has been proposed as a feasible solution where only short text sequences are searched for.

Because of the linear structure of text (i.e. each line and letter is read before the next, with few exceptions), sliding window feature extraction has been employed both when spotting words and doing a full transcription. This feature extraction approach formulates the process of recognition as a sequence labeling problem. The sequence in this case consists of feature vectors, one for each pixel column in a text line. Some of these sliding window features appear to be noisy signals when we plot them as curves. This noise consists of 1) extrinsic noise from the degradation of the manuscript and image acquisition and 2) intrinsic noise stemming from the complexity of handwritten letters and the ability of these features to efficiently capture relevant invariances of the letter shapes. In both image- and signal processing, techniques exist to mitigate the effects of this noise. In this paper, we will add to this by showing that de-noising filters can be used in the sliding window feature space to improve recognition.

1.1 Previous Work

In [6], a collection of sliding window features for off-line text recognition were proposed. Also, a preprocessing scheme, called normalization, was introduced because the authors did not consider the feature extractors robust enough to use directly on the binarized image. By compensating for slant, text height and other writer dependent characteristics, the robustness could be considerably improved. In [7], some of the proposed features were evaluated and extended. These are still some of the most common sliding window features in use.

We have implemented several noise reducing filters, most are based on the papers below.

In [1], a popular approach to median filtering on vector valued signals was proposed. In a window of a set size, the vector being processed was replaced by the “middlemost” vector using the l_1 or l_2 norm. This was shown to produce a robust estimate of a received multichannel signal containing bit errors and impulse noise. The filter was not applied to an image but to a signal similar to a feature vector sequence.

In [11], a filtering technique very similar to a local weighted average was proposed. The important contribution was that differences in intensity were taken into account in the weight function. The result was a weighted averaging filter that

preserved gradients in the data yet still acted as a smoothing filter where no strong gradient occurred. Filtering was performed on a local patch around the element being processed. We have adapted the bilateral filter to our setting where signals were vector valued and spatial relations one-dimensional.

In [3], a filtering technique was proposed called non-local means where no spatial distance in the image was considered when doing a weighted average. The distance between two image patches was measured by only using intensity values in a local patch. Several image patches considered close, by some similarity measure, were then merged by a weighted average to create the filtered image patch. This was based on the assumption that at some level of detail (defined by the size of the local patch) a lot of similarity between different image parts can be found. The non-local means filter have successfully been applied in many situations including both on 2D and 3D data. All local patches were transformed into vectors in some “patch space.” Hence, the dimensionality of the original data does not matter as long as it can be processed into patch vectors. In our setting, this let us create patches from neighbouring feature vectors along a sequence and do filtering in the same way as on an image.

In [8], a word image matching algorithm based on dynamic time warping (DTW) was shown to be successful for word-spotting on the Washington letters. In their approach, the results are greatly improved by using heuristics for finding non-matches, before DTW, developed by [5]. A database of the Washington letters, with segmented words, was later made public in [4]. We wanted to be able to show the effects of removing noise with filtering on a word-spotting setup proven to work. We have done this without changing the feature extraction process, the word matching method or evaluation data from [8]. We have however removed the pruning rules from our re-implementation since they introduce false negatives.

2. METHOD

A common approach to handwritten text recognition is using sliding window features, formulating the recognition as a sequence matching problem. When looking at the generated feature vectors, some are very noise sensitive (even after normalization). When extracting the upper contour feature, some pixel above the text line incorrectly classified as foreground would cause the feature signal to “spike” before jumping back to the actual contour. Other features are sensitive in similar ways.

To mitigate the effect of noise, we have investigated several filtering techniques. In this paper, the filtering was performed in the feature domain, though most filters have an equivalent in the image domain. To evaluate the effects of noise filtering in the feature vector sequences, we re-implemented the work from [8] using DTW. DTW is very noise sensitive, letting us (as a starting point) investigate the filtering without the extra robustness and learning capabilities added by HMMs or Neural Networks.

2.1 Sliding window features

Sliding window feature extraction, in handwriting recognition, creates a sequence with feature vectors along the text

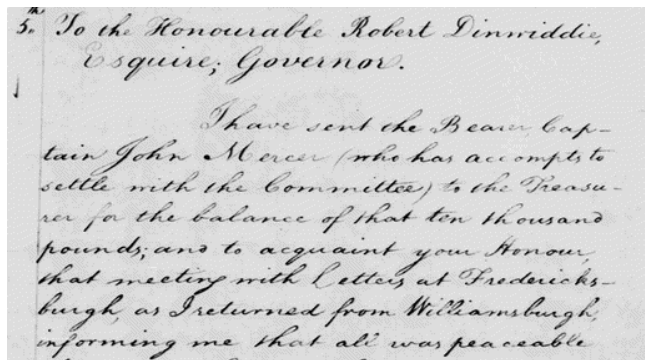


Figure 1: A part of a page from the Washington letters, used in our evaluation. The intention of the writer was that other people should be able to read the text without misinterpretations, as opposed to taking notes for personal use. This gives a very regular and carefully written text.

lines [6, 7]. Several one-dimensional features are generated for each pixel column, after segmentation, and concatenated to form the feature vectors. Each element in the feature vector capture some, to handwriting recognition, important characteristic of the letter at that column. In [8, 7], a set of four heuristic features were proven to work on our evaluation data set. We have chosen to use these features for our implementation though many others, notably [9], would be interesting to explore.

Projection profile The vertical projection of the pixel column i.e. the sum of foreground pixels in one column.

Upper contour The position of the foreground pixel, in the pixel column, with the highest position i.e. lowest y coordinate value.

Lower contour The position of the foreground pixel, in the pixel column, with the lowest position i.e. highest y coordinate value.

Foreground/background transitions The number of transitions between the foreground and the background while following the pixels in a column from lowest to highest y coordinate value

After extraction, all feature values were normalized to the range $[0, 1]$. This also restricts the vector length that can not be longer than $\sqrt{4}$. The above features required some preprocessing of the word images before feature extraction called normalization [6]. This process increased the robustness of the feature extraction (e.g. by removing slant from the handwriting). In our evaluation database, this step had already been performed on the segmented word images.

2.2 Word-spotting

Our evaluation data was segmented into words that we have compared using dynamic time warping (DTW). We aimed to replicate the important paper [8], where they show the viability of DTW in a real world application, but with added filtering.

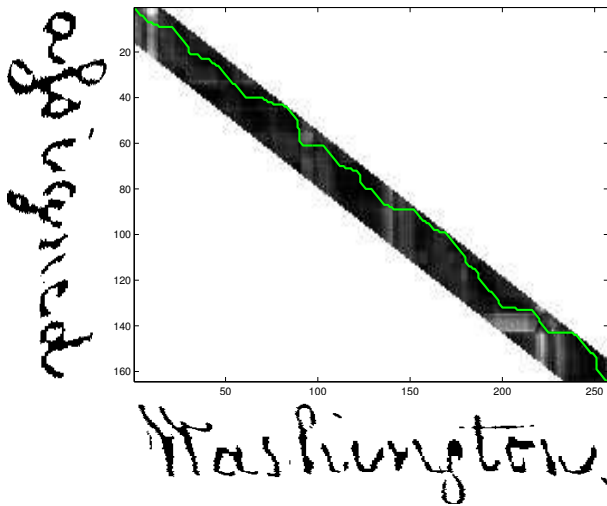


Figure 2: Illustration of the Sakoe-Chiba constraint on the cost matrix of DTW. The lowest cost path through the weight matrix is shown in green. The words along the left and bottom illustrates how the sequences are aligned. At every set point of the cost matrix, one feature vector from each of the words have been compared.

Dynamic time warping finds the optimal stretching, or warping, to fit two sequences of vectors together. By using dynamic programming, an optimal match can be found with respect to the dissimilarity measure between single points of the sequences. Let A and B be the feature vector sequences generated from any two words. A match cost matrix comparing every element in sequence A with every element in sequence B is created. Often, the square of the euclidean distance between the feature vectors is used as a cost function. The cost of the lowest cost path through the matrix, from the upper left corner to the lower right corner, is the difference cost for the word pair. The allowed steps through the matrix are down, right and down right. The diagonal step signifies matching elements of the sequences while down or right steps skips an element in one of the sequences. Each step that is not a perfect match will add a cost to the lowest cost path.

The warping path was normalized by dividing its cost with its length to not make cost proportional to word length. Dynamic programming is significantly less computationally demanding than brute forcing a warping but still takes some time. An important way to increase accuracy and lower computational cost is the Sakoe-Chiba band constraint[10]. By only allowing a narrow strip along the diagonal of the cost matrix for the lowest cost path, pathological warpings and a lot of calculations can be avoided. We have used a restriction of 15 elements around the diagonal (the same as in [8]). In figure 2, an example is shown of the cost matrix with the Sakoe-Chiba band constraint and the lowest cost path.

In [8], pruning was an important step in the word spotting pipeline. Unlikely matches were ruled out before DTW to lower the computational cost. These pruning rules used as-

pect ratio, bounding box area and a fast way to detect the number of descenders to exclude word pairs. Since these rules also created many false negatives, setting a limit on the best possible recognition, we have not used them in our implementation.

2.3 Gaussian filter

Using a Gaussian filter is a common approach to removing noise. The assumptions are that neighbouring signal/image elements are often very similar (on some chosen scale) and that noise in the data is uncorrelated. Hence, averaging data elements with their neighbours can then strengthen the signal and eliminate noise. By using a Gaussian function for a weighted average, neighbouring pixels are given a higher weight than neighbours of neighbours further away. A discrete Gaussian window w was given by equation 1.

$$w(n) = \frac{1}{K} e^{-\frac{n^2}{2\sigma^2}}. \quad (1)$$

Here, n was an interval of integers (e.g. [-20,20]) defining the filtering window with an odd number N giving the width. The discrete case normalization constant K were given by equation 2.

$$K = \sum_{n=-\lfloor N/2 \rfloor}^{\lfloor N/2 \rfloor} w(n). \quad (2)$$

Performing a weighted average on the feature vectors is equivalent to averaging each dimension separately. Hence, filtering was performed by separately convolving each dimension of the feature data with the filtering window. A drawback with using this method on images is that edges are smeared out. The same effect was observed here.

2.4 Median filter

The median filter is a common approach to removing spiking noise (salt and pepper noise) in images. It was not obvious how to apply this approach in the sliding window feature sequences.

The naive approach would be to filter each feature type separately (i.e. contour separately from FG/BG transition etc). This would remove extreme values in each feature dimension separately, without taking the full feature vector into account, which might not be desirable. We have however implemented and evaluated this approach.

Median filters for multi channel signals were introduced in [1]. A noise reducing filter in a multi channel digital receiver handles sequences of signal vectors, the same type of data as given by a sliding window feature extraction. For each vector in the sequence, the neighbouring set X of vectors in a window of a given width w was collected. In this set, the median vector $x_m \in X$ was chosen as the vector in the local set X minimizing the sum in equation 3.

$$\sum_{x \in X} \|x - x_m\| \quad (3)$$

The median feature vector from each local set X , replaced the original feature vector at each position in the feature vector sequence. The norm in equation 3 is defined in [1] as being either the l_1 or l_2 norm, depending on application. We have implemented both and compared them with the above described naive median filter.

2.5 Bilateral filter

In [11], the authors improved the performance of smoothing filters (like the Gaussian filter) by taking local image similarity into account. A common drawback of a smoothing filter is that when it encounters an edge, it does not preserve the strong gradients.

To mitigate the problem of the smoothing of edges, the authors of [11] proposed a local modification of the smoothing kernel. At each point in the area covered by the local kernel, the distance in intensity space from the middle point (i.e. the point being processed) was calculated. After some normalization, each point in the local kernel was multiplied by the normalized intensity distance at that point. This creates a smoothing that does not assume any neighbourhood to be similar, but takes differences into account according to some normalization method (with some parameter settings).

Adapting the bilateral filter to sliding window features vectors was done by defining closeness along the vector sequence and using vector distances as intensity similarity. In equation 4, the similarity function for the feature space is shown. It is the euclidean distance between two feature vectors normalized using a Gaussian function with the parameter σ_v (σ -value) as standard deviation. It was not necessary to use a Gaussian function for normalization in the original formulation of the filter. We have chosen to build upon the Gaussian case bilateral filter.

$$s(v_i, v_j) = e^{-\frac{\|v_i - v_j\|^2}{2\sigma_v^2}} \quad (4)$$

The closeness in our case is defined as a euclidean distance between pixels inside a Gaussian function, given by equation 5, with the standard deviation σ_s (σ -spatial).

$$c(i, j) = e^{-\frac{\|i - j\|^2}{2\sigma_s^2}} \quad (5)$$

Given the above equations, the filtering is performed as in equation 6 at each position i given the set of feature vectors v . To ensure that the weights given by equations 4 and 5 sum to 1, the normalization constant K is defined as in equation 7.

$$BL(i) = \frac{1}{K} \sum_j c(i, j) s(v_i, v_j) v_j \quad (6)$$

$$K = \sum_j c(i, j) s(v_i, v_j) \quad (7)$$

Note that if the parameter σ_v is set too high, the Gaussian case bilateral filter degenerates to a plain Gaussian smoothing.

2.6 Non-local means filter

In [3], a global weighted means filter was presented called the non-local means filter. Filtering was performed on local patches without taking geometric vicinity into account. Similar patches, with respect to some similarity measure, were joined by a weighted average to form the filtered data.

When filtering a 2D or 3D image using the non-local means filter, patch vectors are created from a local neighbourhood, of some predefined size, around each data element. Geometric vicinity in a patch can, if needed, be included into the patch vector by a Gaussian function suppressing elements further from the centre. A weight function for the weighted average is defined, comparing all patch vectors to the one being processed, with the sum of all weights normalized to 1. Higher weights are given to patch vectors similar to the patch vector belonging to the point being processed

Adapting the non-local means filter to sliding window feature vectors was straight forward. We defined vicinity along the sequence instead of in two or higher dimensions, the equivalent of local patches being concatenated neighbouring feature vectors. Feature vectors were concatenated from a local set of N vectors around each vector in the original sequence. Also, a weight function was defined, equation 8, that was only dependent on distances in the feature space. All feature vectors multiplied by the weight function were then summed to form the new filtered vector as in equation 10.

$$w(i, j) = \frac{1}{Z(i)} e^{-\frac{\|v_i - v_j\|^2}{2h^2}} \quad (8)$$

The vectors v_i and v_j are the concatenated vectors from the neighbourhood around the positions i and j in the sequence. Here the variable i is the position currently being processed and j traverses the sequence. The normalization constant $Z(i)$ is given by equation 9 and assures that $\sum_j w(i, j) = 1$ for any i . In our discrete implementation, $Z(i)$ is the sum of the values given by equation 8.

$$Z(i) = \sum_j e^{-\frac{\|v_i - v_j\|^2}{2h^2}} \quad (9)$$

Euclidean distance is used as a dissimilarity measure between vectors in the feature space. The exponential function provides a normalization giving greater weight to small distances in a non-linear way. The parameter h tunes the normalization to be more or less tolerant of large distances.

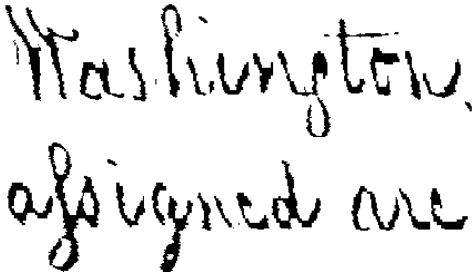


Figure 3: Segmented, binarized and normalized words from the Washington database, used in our evaluation. Note the difference in slant of the letters compared to the words in figure 1 showing an unprocessed part of a page. The normalization is also responsible for the slightly unusual backward slant of some ascenders.

A small h lets the most similar feature vectors influence the element being processed to a higher degree than with a larger h . Since h affects the shape of a Gaussian weight function, it is important to remember that values beyond three standard deviations ($\sigma = h$ in this case) are very small.

$$NLM(i) = \sum_{j \in I} w(i, j) v_j \quad (10)$$

In equation 10, the non-local weighted average is shown. This function is repeated for every position in the feature vector sequence. Running a non-local means filter is computationally heavy since all elements have to be compared to every other. If memory was not limited all distance calculation could be done only once, though this is not the case today.

3. EXPERIMENTS

The different filters were evaluated using a public database for several selected parameters. The parameter intervals were chosen to find an optimal parameter combination, assuming the test data.

3.1 The evaluation database

George Washington Papers is a collection of letters written by George Washington found at the Library of Congress¹. From the pages of letter book 1, series 2, pages 270-279 & 300-309 (written between Aug. 11, 1754 and Dec. 25, 1755) a database has been made public in [4]. Examples of words found in the database can be seen in figure 3. All word images were delivered segmented, binarized, normalized and labeled.

3.2 Performance measure

We have chosen a performance measure with its base in the Receiver Operating Characteristic (ROC) type of analysis. In a ROC plot, the y -axis shows the rate of true positive matches (TPR) (i.e. the rate of correctly identified true matches at a specific classifier threshold) and the x -axis

shows the corresponding rate of false positives (FPR). To get a single value to characterize the performance of a classifier, the area enclosed by the ROC curve is integrated to create the Area Under the Curve (AUC) performance measure[2]. A weakness in using a single scalar to characterize performance is that many relevant aspects might be lost. However, showing the ROC curves for a two-dimensional parameter search in a meaningful way is hard on paper. In the following sections AUC will be used to show the performance of each filtering technique, along with a comparison without using filtering called “baseline.”

3.3 Gaussian filtering

Results from the performance evaluation can be seen in figure 4, with selected data points shown in table 1. Evaluation was performed for several sample points of the standard deviation parameter σ in the interval $[0, 10]$. Note that σ was not the width of the filter. Filter width was variable, depending on σ , as to include as much of the Gaussian function as possible without doing unnecessary calculations. The Gaussian filtering approach is very fast since it only consists of one convolution per feature dimension. This makes it suitable for larger collections of text.

Table 1: Selected results for the Gaussian filter together with the baseline. The number in bold font shows the best performance.

Filter type	σ	AUC
Baseline	-	0.852
Gaussian	0.34	0.853
Gaussian	2	0.876
Gaussian	8	0.849

As shown in figure 4, a large span for the parameter σ gave an improvement. The smoothness of the curve suggests a stability when it comes to the choice of σ . When σ is set to 2, two standard deviations include 8 pixels. The mean letter width in our evaluation database is about 30 pixels (and as many feature vectors). Hence, a feature signal can still after filtering change several times during one character. Single pixel noise stemming from the binarization is smoothed by the filter keeping the important characteristics of each letter. When σ approaches 8, two standard deviations cover an area comparable to the mean letter width, letter characteristics are obscured. If σ is too small, the discrete Gaussian function peaks at its centre and values at neighbouring points are negligible. The convolution then returns the unfiltered sequence. In table 1, the best result using the Gaussian filter is shown along with sample point before and after the interval where performance improved.

3.4 Median filtering

The median filter (described in section 2.4) was run with the parameter *width* set to odd numbers between 3 and 23. In figure 5 and table 2, the evaluation results are shown for the three median filter types investigated. A dimension wise mean filter was implemented for comparison. The mean filtering was performed dimension wise where the mean value was found in a local neighbourhood as wide as the width parameter (in the same way as with the median filters).

As shown in figure 5, the dimension wise median filter did

¹<http://memory.loc.gov/ammem/gwhtml/gwseries2.html>

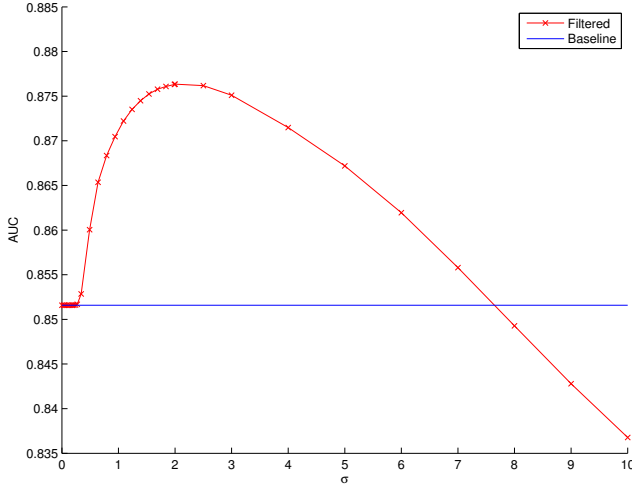


Figure 4: Evaluation results for the Gaussian filter with the parameter σ in the interval $[0, 10]$ (section 3.3).

Table 2: The dimension wise median filter performed better than the baseline. However, no parameter setting for the vector median filters gave a higher evaluation result than the baseline. Above the highest obtained result from each median filter type are shown. The numbers in bold font show the best results from the two filters performing better than the baseline.

Filter type	Width	AUC
Baseline	-	0.852
Dimension wise mean	7	0.875
Dimension wise median	5	0.859
l_1 vector median	3	0.8497
l_2 vector median	3	0.8497

improve the results. It performed under the baseline above a width of 9 and best at 5. This implies that extreme values at a very small local neighbourhood can be filtered out. With larger widths, filtering removes more important information than noise. However, the performance of the mean filter was the best in the collection implying that the median removed important local extreme values in most cases.

The vector median filter from [1] did not reach above the baseline for any parameter setting. An expectation with the vector based filters was that the extra dimensions would increase stability in the signal, and performance as a consequence. However, the opposite turned out to be the case. In contrast to the dimension wise filters, extreme vectors were removed when trying to find the local “middlesmost” vector in the vector median filters. This is an advantage when, for example filtering a signal with spiking noise, assuming a smooth sequence of vectors. Here it might have destroyed outliers, carrying important information. Another possibility is that the order of the vectors are scrambled, some removed and some duplicated. Performance were hardly affected by changing the norm from l_2 to l_1 . The noise found in our evaluation database or feature selection could not be

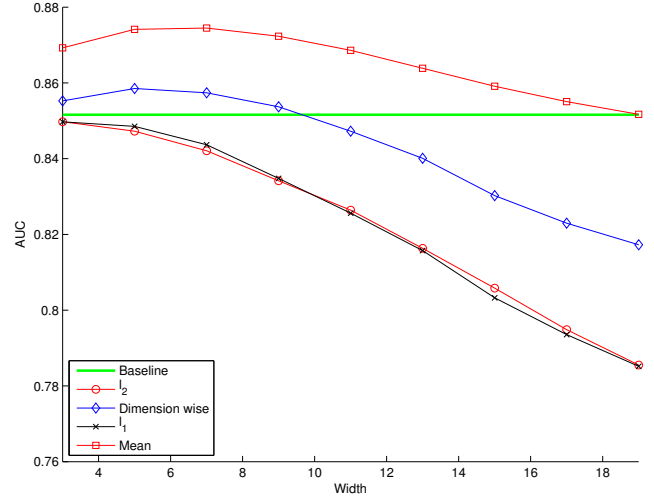


Figure 5: Evaluation results for the Median filter with the parameter $width$ set to odd numbers in the interval $[3, 19]$ (section 3.4). The baseline and the mean filter are included for comparison.

improved using the vector median approach.

3.5 Bilateral filtering

Evaluation results for the Bilateral filter (described in section 2.5) can be seen in table 3 and figure 6. Several sample points for each parameter were investigated in the interval $[0, 10]$ for σ_s and the interval $[0, 2]$ for σ_v .

Table 3: Evaluation results for some parameter settings for the Bilateral filter (section 3.5).

Filter type	σ_s	σ_v	AUC
Baseline	-	-	0.852
Bilateral	2	4	0.876
Bilateral	8	0.4	0.861
Bilateral	8	4	0.852

When $\sigma_v \approx 0.5$ and σ_s was in the interval $[5, 9]$, a ridge appeared on the parameter surface. Here the spatial Gaussian was wide but the influence of the feature vector similarity compensated by focusing on smoothing similar vectors.

As σ_v approached 2, the bilateral filter degenerated into the Gaussian filter. Because of how the features were normalized and the number of dimensions of the feature space, the maximum length of a feature vector was $\sqrt{4} = 2$. This gave that all vectors were within 1 standard deviation in the normalizing function (equation 4). The effect from the feature vector similarity function were then negligible.

Overall, the effects of the similarity function in the bilateral filter were small. Only in a small regions did it dominate over the geometrical closeness function. However, at no point did the bilateral filter perform below the Gaussian filter. The additional computational cost of running the bilateral filter instead of the Gaussian filter were small. Even when the number of pages grows, bilateral filtering is a good strategy for fast de-noising.

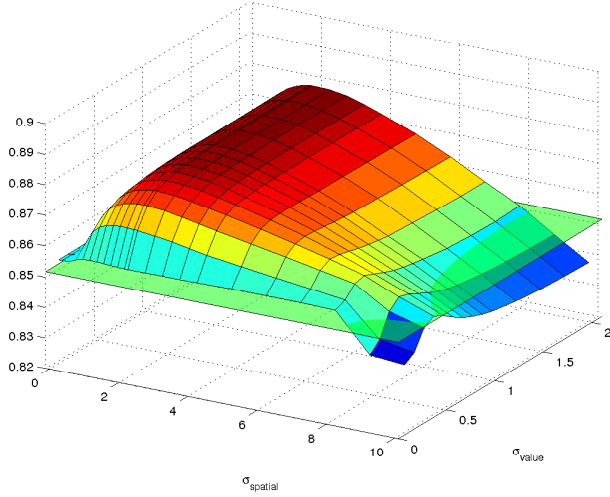


Figure 6: Evaluation results for the Bilateral filter (section 3.5). The curved surface shows the AUC at different settings for the two filtering parameters (each mesh crossing is a sample point). The semi-transparent green plane shows the AUC without using filtering as a comparison.

3.6 Non-local means filtering

Evaluation results for the Non-local means filter (described in section 2.6) can be seen in table 4 and figure 7. The parameters investigated were odd numbers from 1 to 7 for N and several points between 0.01 and 8 for h . Note that the parameter h has the same role in the non-local means filter as the parameter σ_v in the bilateral filter.

Table 4: Evaluation results for some parameter settings for the Non-local means filter (section 3.6). The numbers in bold font shows results from the plateau in figure 7, those are also the maximum performance.

Filter type	Neighbourhood	h	AUC
Baseline	-	-	0.852
Non-local means	3	4	0.913
Non-local means	1	4	0.903
Non-local means	3	0.05	0.847

The non-local means filter gave the highest performance of the investigated filters. As seen in figure 7, the performance is below the baseline for low values of h but increased steeply when raising the value of the parameter. At $h \approx 2$, the improvement from the filter reached a plateau. On this plateau a small increase in performance could be observed when letting h approach 8, but not very much. This result was unexpected since the maximum length of a feature vector is inside the span of one standard deviation in the weight function.

An important point of the non-local means filter is that a lot of feature vectors are necessary for increasing the evaluation results. When running the filter on just 500 words (i.e. 10% of the database), evaluation results was considerably lower than when using more words. A higher number of similar patches to choose from gives a higher probability of removing

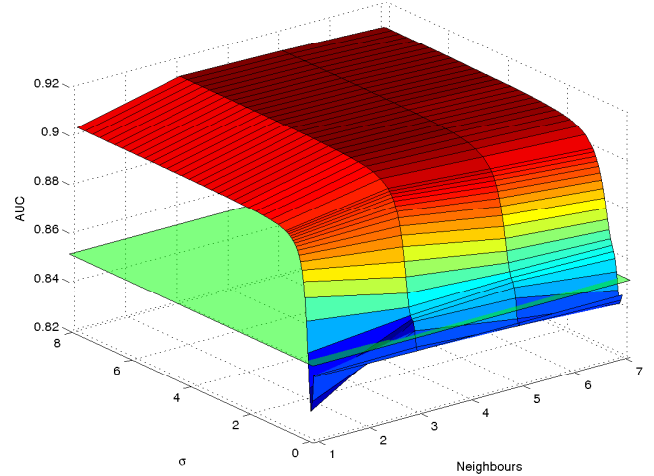


Figure 7: Evaluation results for the Non-local means filter (section 3.6). The curved surface shows the AUC at different settings for the two filtering parameters (each mesh crossing is a sample point). The semi-transparent green plane shows the AUC without using filtering as a comparison.

noise.

3.7 A Visual Inspection

To gain insight into the effects of de-noising, we have visualized the feature signals in figure 8 and the corresponding spectra in the Fourier domain in figure 9. The non-local means filter appears to “invent” information in the individual feature signals. This can be explained as borrowing information from the other feature dimensions to “repair” missing information in individual features. Also, the most successful filters (non-local means, Gaussian) both remove higher frequencies.

4. CONCLUSIONS

We have introduced de-noising for sliding window features and been able to show that it improves performance in word spotting. This new kind of filtering that we propose for word spotting is significantly different from filtering the image itself, like in image restoration.

The selection of filters that we have evaluated is far from complete. In our experiments, the non-local means filter was the most efficient de-noising method with a 6% increase of AUC score. Variants of the median filter performed better than no filtering, but not much. An interpretation of that might be that averaging, combining of information from several feature points (image columns) is important. Weighted averaging, using Gauss- and box functions, gave a 2% increase of AUC. Finally, the optimal bilateral filter was in fact a filter with a large σ_{value} , in essence a Gauss weighted average filter.

The averaging effect in weighted averaging using Gauss- and box functions may also be interpreted as a de-localization of the image features along the text line, i.e. it is easier to compare two slightly shifted or warped feature signals if they are

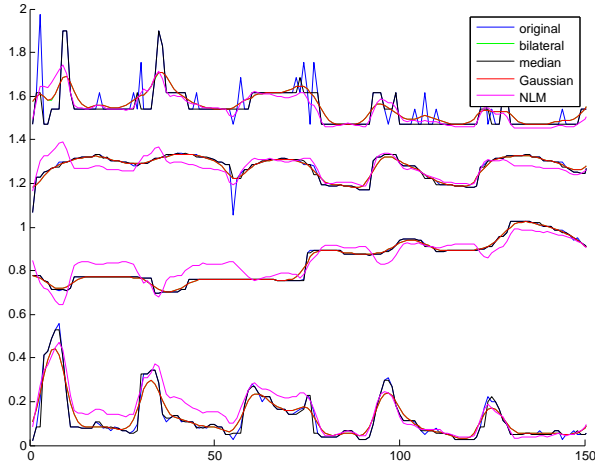


Figure 8: A comparison of the filters on feature data from the evaluation database. Note in particular that non-local means appears to sharpen individual features by using information from all feature dimensions.

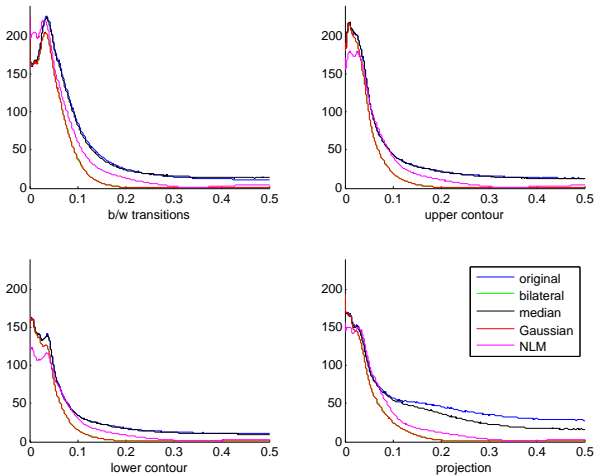


Figure 9: A comparison of the Fourier spectrum of feature data with different filters. Non-local means appears to remove higher frequencies, while at the same time enhancing the signal (in particular peaks) in the lower frequency bands.

first filtered with a low-pass filter. One explanation is thus that local averaging makes the elastic matching in dynamic time warping easier, a hypothesis that we have not been able to test yet. The efficiency of non-local means is well known in the imaging community, for 2-D and 3-D signals, but in this context we have not been able to fully explain why this method is the best. The large size of the optimal kernels in NLM, for instance $h = 2$, was also unexpected. This resulted in a compression of the feature space, where each feature channel approached its average value. In images of the feature signals, where the signals have been scaled to be easier to compare, we also see that non-local means appears

to both smooth and sharpen the signals. The suppression of high frequencies, combined with some kind of sharpening in the lower half of the spectrum, is also confirmed from Fourier analysis of the feature signals seen in figure 9.

We conclude that feature space de-noising of sliding window features increases performance in word spotting. Possibly, this is also true for other kinds of handwritten text recognition, where feature space de-noising could be included as an intermediate step. These results also point in the direction to search for more efficient and stable features, which are less prone to intrinsic noise and better captures the relevant shape variation of handwritten characters. Indeed, we have only evaluated the effects of feature space filtering on a single popular feature set in this limited study.

5. REFERENCES

- [1] J. Astola, P. Haavisto, and Y. Neuvo. Vector median filters. *Proceedings of the IEEE*, 78(4):678–689, 1990.
- [2] A. P. Bradley. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30:1145–1159, 1997.
- [3] A. Buades, B. Coll, and J.-M. Morel. A non-local algorithm for image denoising. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 2 - Volume 02*, CVPR '05, pages 60–65, Washington, DC, USA, 2005. IEEE Computer Society.
- [4] A. Fischer, A. Keller, V. Frinken, and H. Bunke. Lexicon-free handwritten word spotting using character hmms. *Pattern Recogn. Lett.*, 33(7):934–942, May 2012.
- [5] S. Kane, A. Lehman, and E. Partridge. Indexing george washington’s handwritten manuscripts. *Center for Intelligent Information Retrieval, Computer Science Department, University of Massachusetts, Amherst, MA*, 1003, 2001.
- [6] U.-V. Marti and H. Bunke. Using a statistical language model to improve the performance of an hmm-based cursive handwriting recognition system. *International Journal of Pattern Recognition and Artificial Intelligence*, 15(01):65–90, 2001.
- [7] T. M. Rath and R. Manmatha. Features for Word Spotting in Historical Manuscripts. In *International Conference on Document Analysis and Recognition*, pages 218–222, 2003.
- [8] T. M. Rath and R. Manmatha. Word image matching using dynamic time warping. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 2, pages II–521–II–527 vol.2, 2003.
- [9] J. A. Rodriguez and F. Perronnin. Local gradient histogram features for word spotting in unconstrained handwritten documents. *ICFHR 2008 (Int. Conf. on Frontiers in Handwriting Recognition)*, 2008.
- [10] H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 26(1):43–49, 1978.
- [11] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *ICCV*, pages 839–846, 1998.