USING SIMD INSTRUCTIONS FOR FAST LIKELIHOOD CALCULATION IN LVCSR

Stephan Kanthak, Kai Schütz, Hermann Ney

Lehrstuhl für Informatik VI, RWTH Aachen
University of Technology, 52056 Aachen, Germany
{kanthak,ney}@informatik.rwth-aachen.de, ks@axys.de

ABSTRACT

Most modern processor architectures provide SIMD (single instruction multiple data) instructions to speed up algorithms based on vector or matrix operations. This paper describes the use of SIMD instructions to calculate Gaussian or Laplacian densities in a large vocabulary speech recognition system. We present a simple, robust method based on scalar quantization of the mean and observation vector components without any loss in recognition performance while speeding up the whole system's runtime by a factor of 3. Combining the approach with vector space partitioning techniques accelerated the overall system by a factor of over 7. The experiments show that the approach can be also applied to Viterbi training without any loss of accuracy. All experiments were conducted on a German, 10,000-word, spontaneous speech task using two architectures, namely Intel Pentium III and SUN UltraSPARC.

1. INTRODUCTION

The number of log-likelihood calculations for the emission probabilities of HMMs increases with the complexity of the acoustic models used in LVCSR systems. With standard approaches as reported in [1, 5] the computational need for these calculations can be significantly reduced by evaluating only a small subset of densities. All of these methods have in common that their parameters depend on the whole recognition system and must be adjusted whenever the system settings change. Poorly adjusted, they can cause a significant deterioration in recognition performance.

In this paper we present a different approach. The instruction sets of most modern processor architectures have recently been extended by SIMD instructions. We will show that using SIMD instructions for the log-likelihood calculations speeds up the whole system by 67% without affecting the word error rate. While accelerating the vector distance calculation itself, the method can be combined with approximative vector space partitioning approaches. Additionally, we show that applying it to standard Viterbi training results in a speed gain of up to a factor of 2.2 without increasing the word error rate.

All fast log-likelihood calculation techniques investigated in this paper are embedded into the time-synchronous beam search algorithm. The recognition is based on a tree-organized pronunciation lexicon [3] in connection with a class-trigram language model.

The remainder of this paper is organized as follows: In Section 2, we briefly describe the principles of log-likelihood calculations for continuous mixture densities. In Section 3, a simple method for appropriately quantizing the mean and observation vector components to small integer values is introduced. In Section 4, we show the application of SIMD instructions to the quantized vectors. In Section 5, experimental results for the German spontaneous speech task VerbMobil [2] with a 10,000-word vocabulary are given.

2. MIXTURE DENSITIES

The emission probability $p(x|s)$ of an HMM state s is expressed as the weighted sum of probability density functions $p(x|l,s)$ (pdfs) for an acoustic vector $x \in \mathbb{R}^D$:

$$p(x|s) = \sum_{l=1}^{L(s)} p(l|s) \cdot p(x|l,s),$$

with $p(l|s)$ denoting the weight of the l-th mixture density component $p(x|l,s)$. Using the maximum approximation the negative log-likelihood is

$$-\log p_G(x|s) = \min_l \left\{ -\log p(l|s) + \frac{1}{2}\sum_{d=1}^{D} \left(\frac{x_d - \mu_{sld}}{\sigma_{sld}}\right)^2 + \frac{1}{2}\sum_{d=1}^{D} \log(2\pi\sigma_{sld}^2) \right\},$$

for Gaussian pdfs and

$$-\log p_L(x|s) = \min_l \left\{ -\log p(l|s) + \sum_{d=1}^{D} \frac{|x_d - \mu_{sld}|}{\sigma_{sld}} + \sum_{d=1}^{D} \log(2\sigma_{sld}) \right\},$$

for Laplacian pdfs. In both cases μ_{sld} and σ_{sld} denote the d-th component of the model-specific mean and scale vectors of the component density l of state s. Assuming a globally pooled scaling vector, we normalize each component of the observation and the mean vectors by the corresponding component of the scaling vector beforehand. We denote the resulting vectors with x' and μ'_{sl}, respectively. Using these normalized vectors the negative log-likelihoods can be rewritten as a minimization over the squared l_2 norm and l_1 norm:

$$-\log p_G(x|s) = \min_l \left\{ \frac{1}{2} ||x'_d - \mu'_{sld}||_2^2 + c_{Gsl} \right\} \qquad (1)$$

for Gaussian pdfs and

$$-\log p_L(x|s) = \min_l \left\{ ||x'_d - \mu'_{sld}||_1 + c_{Lsl} \right\} \qquad (2)$$

for Laplacian pdfs, where c_{Gsl} and c_{Lsl} denote constants that incorporate the model-specific mixture weight and normalization term.

For simplicity, in this paper we will use Gaussian pdfs to describe the proposed method. However, it should be noted that some recent processor models provide single instructions to calculate the l_1 norm. These instructions have been intended to accelerate motion-estimation and can also be used to calculate log-likelihoods of Laplacian pdfs very efficiently.

3. SCALAR QUANTIZATION

SIMD instructions of modern processor architectures support signed or unsigned integer values with data widths of 8, 16 or 32 bits. Thus, parallelizing the computation of Eq. (1) and (2) with SIMD instructions exposes the problem of scaling down the floating point values of the observation and mean vector components to small integer values appropriately. In the following we will focus on scaling the vectors to unsigned 8-bit integer values.

To avoid an additional dependency on the component index d in the log-likelihood computation, we propose to use a single scaling factor α and a bias β for all vector components. Once estimated they must be applied to all observation vectors x'_t and mean vectors μ'_{sld} before the log-likelihood calculation. To obtain comparable scores the results of the log-likelihood calculations need to be rescaled afterwards. Thus, in case of Gaussian pdfs we can rewrite Eq. (1) as:

$$-\log p_G(x|s) = \frac{1}{\alpha^2} \cdot \min_l \left\{ \frac{1}{2} ||x''_d - \mu''_{sld}||_2^2 + \alpha^2 \cdot c_{Gsl} \right\},$$

with

$$x''_d = \alpha \cdot x'_d + \beta$$

and

$$\mu''_{sld} = \alpha \cdot \mu'_{sld} + \beta.$$

α and β must be chosen appropriately to minimize the number of overflows. Note, that the bias β is only required to adjust the range of the quantized vector components, but does not change the log-likelihood itself. Nevertheless, overflows with respect to the target range cannot be completely avoided with this simple scaling method. They should be at least detected and clipped to fit the range.

Table 1: Minimum and maximum values for different sets of normalized vectors and the resulting average error and overflows measured on the normalized training vectors when using scalar quantization from single precision float to unsigned 8 bits and clipping ($\beta = 128$, 50 hours of speech).

set of vectors	$x_{\min}$	$x_{\max}$	r	α	quant. error	over-flows
mean	-22.2	16.1	255	5.74	0.09	8
			200	4.50	0.11	2
			100	2.25	0.22	0
training	-39.1	26.9	255	3.26	0.15	0
			200	2.56	0.19	0
			100	1.28	0.39	0

Obviously, the chance of overflows can be minimized by estimating α using the normalized training vectors or, even simpler, the normalized mean vectors. Assuming that the mean over the whole set of vectors is zero and the range of the target data width is given by r, estimating α can be easily performed by determining the minimum and maximum values, $x_{\min}$ and $x_{\max}$, over all components of the examined set of vectors:

$$\alpha = \frac{r}{2 \cdot \max(|x_{\max}|, |x_{\min}|)}.$$

Experiments showed that r needs to be chosen in a conservative way because the actual range of new observation vectors is unknown. In all experiments, we estimated the scaling factor α on the set of mean vectors which proved to be sufficient. Table 1 shows that for proper values of r simple scalar quantization causes a remarkably small quantization error and almost no overflows. The quantization error is measured as average absolute difference between original value and quantized value.

In addition to the computational gain from SIMD instructions using these quantized vector components they even require less memory. In comparison with `float` values a quantization to 8 bits reduces the storage requirements for mean vectors by a factor of 4.

4. SIMD INSTRUCTIONS

SIMD instructions help to accelerate algorithms that apply the same operation to many data paths containing different data. Most vector or matrix manipulations, and, as we will show, calculating the l_2 or l_1 norm can benefit from SIMD instructions. Examples of SIMD instruction sets of recent processor architectures are:

- MMX [7] (Intel Pentium MMX, AMD K6)
- VIS [8] (SUN UltraSPARC V8)
- MVI (Compaq Alpha EV6)
- AltiVec (IBM/Motorola PowerPC G4)

In Figure 1 the overall structure of the MMX implementation of the squared l_2 norm is shown. The example code processes two 8-dimensional vectors with each component

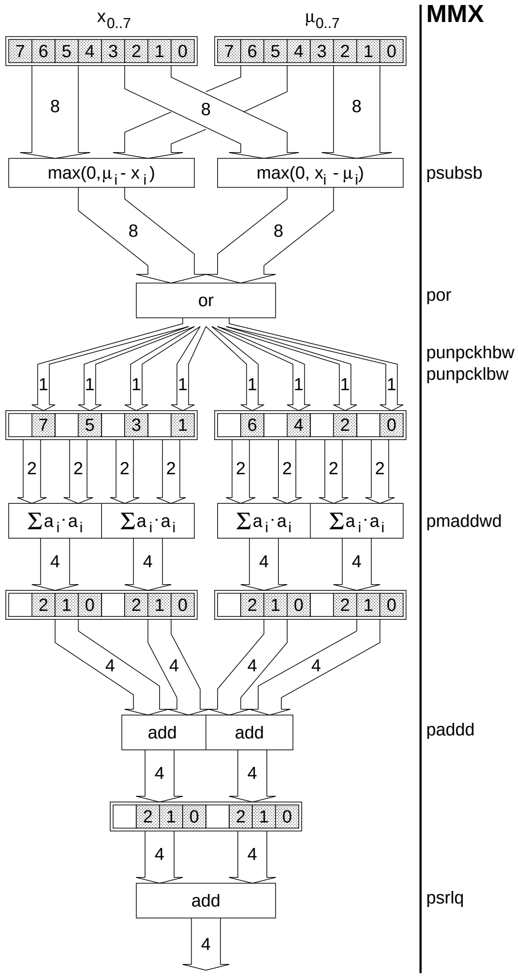

Figure 1: Data flow for the Intel Pentium III of calculating the squared l_2 norm of the difference of two 8-dimensional vectors using SIMD instructions.

being an unsigned 8-bit integer. In the first step, we calculate the difference of the two vectors. To avoid the 9-bit results, we calculate the absolute difference instead. With MMX this can be accomplished using subtraction with saturation in both directions and combining the two partial results. After squaring the components and summing them up in two steps as shown in Figure 1, we obtain the total sum as a positive 19-bit integer. For the l_1 norm the operation labeled `pmaddwd` must be replaced by a simple summation. Although we took care of latencies and pipelining we will not discuss these issues here due to space limitations.

For the implementation on SUN UltraSPARC we utilized the linear algebra part of SUN's mediaLib [8] in a slightly modified form. The functions of that library are highly optimized to the UltraSPARC pipeline architecture and make intensive use of the VIS SIMD instructions.

We emphasize the importance of aligning the data vectors in memory appropriately when used by SIMD instruction. Without proper memory alignment the computation time would increase dramatically.

5. EXPERIMENTAL RESULTS

5.1. Test Condition

We performed all tests using the RWTH LVCSR system on the German spontaneous speech task VerbMobil [6]:

- spontaneous speech, speaker independent,
- vocabulary of 10,157 words, articulatory and non-articulatory noises,
- 50 hours of acoustic training data,
- 30 minutes of acoustic testing data: 8 speakers, 336 sentences, 4346 spoken words,
- 33-dimensional acoustic vector, obtained from a MFCC signal analysis followed by linear discriminant analysis (LDA),
- within-word phoneme models with 2501 tied mixtures containing a total of 245,000 Gaussian densities,
- class-trigram language model with a test-set perplexity of 74.6.

The parameters of the scalar quantization were set to $r = 200, \beta = 128$ throughout all experiments.

5.2. Recognition Experiments

Recognition experiments were conducted using within-word acoustic models. In the baseline system we already used look-ahead techniques to increase the efficiency of the search as described in [4]. All pruning parameters were optimized with respect to a minimum search space without increasing the word error rate. The experiments were carried out on a Pentium III 450-MHz PC and a SUN UltraSPARC clocked at 167MHz.

A comparison of the results of various fast log-likelihood calculation methods is given in Table 2. The table shows the real-time factors (RTF) of the log-likelihood calculations and the whole recognition system together with the search space and recognition errors. On both processors the log-likelihood calculations for the baseline results needed more than 80% of the overall recognition effort. It can be seen that using SIMD instructions reduces the RTF of likelihood calculations from 8.4 to 2.0 for the Intel Pentium III and from 19.5 to 9.2 for the SUN UltraSparc without any loss in recognition performance.

The VQ (vector quantization) and PSA (projection search algorithm) techniques described in [1] and [5] reduce the RTF of likelihood calculations from 8.4 to 1.4 causing only a slight increase of the word error rate. By combining them with the SIMD approach, the RTF decreases further to 0.5. The overall system RTF can be reduced from 9.9 to 1.4 for the Intel Pentium III and from 22.6 to 4.2 for the SUN UltraSPARC using a combination of all three techniques.

5.3. Training Experiments

As we have seen in Section 3, the quantization error was remarkably small. Therefore, the proposed method of calculating the log-likelihood of continuous mixture densities

Table 2: Fast log-likelihood calculation methods for a 10,000-word spontaneous speech task using 245,000 Gaussian densities and a class-trigram language model with a test-set perplexity of 74.6 (Pentium III 450 MHz and UltraSPARC 1 Model 170).

method	architecture	search space			word errors [%]			RTF	
		states	arcs	trees	del	ins	WER	log-likelihood calculations	total recognition
baseline	Pentium	1729	784	19	4.3	5.5	25.2	8.4	9.9
baseline	UltraSPARC	1729	784	19	4.3	5.5	25.2	19.5	22.6
SIMD	Pentium	1731	785	20	4.2	5.6	25.2	2.0	3.3
SIMD	UltraSPARC	1731	785	20	4.2	5.6	25.2	9.2	12.9
PSA	Pentium	2130	953	22	4.1	5.7	25.4	4.2	5.8
VQ & PSA	Pentium	1664	777	20	4.2	5.7	25.5	1.4	2.4
VQ & PSA & SIMD	Pentium	1669	778	20	4.2	5.8	25.5	0.5	1.4
VQ & PSA & SIMD	UltraSPARC	1670	778	20	4.2	5.8	25.5	1.5	4.2

Table 3: Real-time factor (RTF) of training and comparison of recognition performance when using the SIMD approach in training. RTFs are given for one iteration of time alignment with an average beam of 14 states (Pentium III 450 MHz).

method	training RTF		recognition word errors [%]		
	log-lik.	total	del	ins	WER
baseline	0.072	0.073	4.3	5.5	25.2
SIMD	0.031	0.033	4.2	5.1	24.2

can also be applied to standard Viterbi training. Table 3 shows that the time spent on log-likelihood calculations is reduced by a total factor of 2.2 while the resulting word error rate decreases. The speedup of the Viterbi alignment is not as high as for the recognition process, because 20% of likelihood calculations were not accelerated. Nevertheless, other training experiments showed that the word error rate at least never increased.

As an implementation detail we add that we reestimated the scaling factor α at the end of each training iteration to keep track of the changing minimum and maximum values measured over all mean vector components.

6. SUMMARY

In this paper, we describe the use of SIMD instructions to accelerate LVCSR systems. We propose a simple, but efficient method for quantizing the mean and observation vector components in order to use them in combination with SIMD instructions. We show that the relatively small quantization error does not affect the word error rate.

Experiments carried out on the German, 10,000-word, spontaneous speech task VerbMobil showed that by the proposed method the time spent on log-likelihood calculations can be reduced by factor of 4.2. This resulted in an overall system speedup by a factor of 3 without any loss in recognition performance. We showed that our approach can be successfully combined with previously published space partitioning methods which finally lead to a total runtime reduction down to 14%.

Additionally we applied the approach to standard Viterbi training and obtained a total speedup a factor of 2.2.

Acknowledgement. This research was partly funded by grant 01 IV 701 T4 from the German Ministry of Science and Technology (BMBF) as a part of the VERBMOBIL project. The views and conclusions contained in this document are those of the authors.

7. REFERENCES

[1] E. Bocchieri: Vector Quantization for the Efficient Computation of Continuous Density Likelihoods, Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing, pp. 692-695, Minneapolis, MN, Apr. 1993.

[2] T. Bub, J. Schwinn: VERBMOBIL: The Evolution of a Complex Large Speech-to-Speech Translation System, Proc. 4th Int. Conf. on Spoken Language Processing, pp. 2371-2374, Philadelphia, PA, Oct. 1996.

[3] H. Ney, L. Welling, S. Ortmanns, K. Beulen, F. Wessel: The RWTH Large Vocabulary Continuous Speech Recognition System, Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing, pp. 853-856, Seattle, WA, May 1998.

[4] S. Ortmanns, A. Eiden, H. Ney, N. Coenen: Look-Ahead Techniques for Fast Beam Search, Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing, pp. 1783-1786, Munich, Apr. 1997.

[5] S. Ortmanns, T. Firzlaff, H. Ney: Fast Likelihood Computation Methods for Continuous Mixture Densities in Large Vocabulary Speech Recognition, Proc. 5th Europ. Conf. on Speech Communication and Technology, pp. 139-142, Rhodos, Greece, Sep. 1997.

[6] A. Sixtus, S. Molau, S. Kanthak, R. Schlüter, H. Ney: Recent Improvements of the RWTH Large Vocabulary Speech Recognition System on Spontaneous Speech, submitted to Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing, Istanbul, Turkey, June 2000.

[7] AMD: AMD-K6 Processor Multimedia Technology, `http://www.amd.com/K6/k6docs/`, Nov. 1999.

[8] SUN: VIS Software Developer's Kit, `http://www.sun.com/microelectronics/vis/`, Nov. 1999.