Abstract — The problem of resolving superpositions in electromyographic (EMG) signals is considered. The shapes of the motor unit action potentials that make up each superposition are assumed to be known a-priori (known constituent problem). Two different and novel belief propagation algorithms have been developed to solve this problem. These algorithms and simulation results are presented in this paper.

Index Terms — EMG signal decomposition, factor graphs, belief propagation

I. INTRODUCTION

EMG signals: Muscle contraction produces electrical activity that can be measured and analyzed. An electromyographic (EMG) signal consists of contributions from several sources (motor units) which discharge repeatedly producing spikes called motor unit action potentials (MUAPs). Four MUAPs are shown at the bottom of Fig. 8. A block diagram as in Fig. 1 can be used to model and to simulate EMG signals; source \( i \) emits a discrete-time binary signal \( X_i \) with \( X_{i,k} \in \{0,1\} \). If \( X_{i,k} = 1 \), we say that source \( i \) “fires” at time \( k \). Each electrode picks up a noisy and filtered superposition of these source signals. For example, electrode \( j \) picks up the EMG signal \( Y_j \) with

\[
Y_{j,k} = \sum_{i=1}^{N} M_i \cdot X_{i,k} + W_{j,k},
\]

where \( h_{i,j,\ell} \in \mathbb{R} \) is the \( \ell \)-th filter coefficients expressing the waveform of MUAP \( i \) in channel (electrode) \( j \), \( N \) is the number of sources, \( M_i \) is the order of the FIR filter corresponding to source \( i \), and \( W_j \) is additive white Gaussian noise (AWGN) for channel \( j \).

Problem statement: Given the block diagram in Fig. 1, decomposing a multi-channel EMG signal \( Y \) means estimating the binary source signals \( X_i \). Many decomposition algorithms have been proposed. However, many practical tools are limited to a few superimposed MUAPs, e.g., only two in [1].

This paper: This paper considers EMG signals consisting of a single superposition as shown in Fig. 8. We want to resolve such superpositions given all filter coefficients \( h_{i,j,\ell} \) and assuming that each source fires exactly one time (known constituent problem). For this, we present two new EMG signal decomposition algorithms, which are extended versions of previously outlined algorithms in [2][3]. As in these papers, our new algorithms are based on graphical models (factor graphs [4]). Non-optimal message-passing (belief propagation) algorithms run on these factor graphs to resolve the superpositions by calculating messages that are sent along the edges of the factor graph. The edges correspond to variables.

II. METHODOLOGY

Factor graph: Based on the block diagram in Fig. 1, we constructed a factor graph. Fig. 2 shows one section of this factor graph. The complete factor graph has one such section for each discrete time \( k \). Note the similarities between the block diagram and the factor graph. We explain the individual nodes of the factor graph below.

Belief propagation algorithm and messages: We use the sum-product rule [4] to calculate an approximation of the marginal conditional probability distributions of the binary source signals \( X \) for the variables \( X \) and \( S \), and joint-Gaussian messages for the

---

Volker M. Koch\textsuperscript{1,2}, Kevin C. McGill\textsuperscript{2}, and Hans-Andrea Loeliger\textsuperscript{1}

\textsuperscript{1}Signal and Information Processing Laboratory, ETH Zurich, 8092 Zurich, Switzerland
\textsuperscript{2}Rehabilitation R&D Center, VA Palo Alto Health Care System, Palo Alto, CA 94304, U.S.A.

koch@isi.ee.ethz.ch, mcgill@va51.stanford.edu, loeliger@isi.ee.ethz.ch
variables below the coefficient nodes \(^2\) (runs on the factor graph in Fig. 3).

\(\textbf{1} \text{ Source nodes:}\) The boxes labeled \(\text{1}\) in Fig. 2 and Fig. 3 represent the state transition probabilities \(p(s_{i,k+1}|s_{i,k})\), which are defined by the finite state model in Fig. 4. This finite state model is basically a shift register with idle states \(\text{1}\) and active states \(\Delta\). State \(\text{5}\) is the initial start state at time “\(k = -1\)” which is only used for simulating signals; it is not used in the decomposition algorithm. The first \(M+1\) (length of the MUAPs of source \(i\) ) active states in Fig. 4 correspond to the taps of the FIR filter that models MUAP \(i\). We expect the MUAPs to lie completely within the EMG signal \(Y\). Therefore, at \(k = 0\), the system can be in one of the left idle states \(\text{1}\) or in the first (left) active state. At \(k = 1\), it can be in the second left idle states or in one of the first two active states, etc. The message update rules are derived and implemented in a straightforward fashion (in accordance with the sum-product rule) given this model. When calculating message \(\mu_T\) in Fig. 4, \(\mu_X\) is assumed to be uniform, which is equivalent to having no incoming message on the edge corresponding to \(X\).

\(\textbf{2} \text{ Coefficient nodes:}\) The boxes labeled \(\text{2}\) in Fig. 2\(^1\) represent the coefficient nodes (see also Fig. 5). They translate between state variables \(S\) and MUAP values \(Z\). For this, we interpolate the MUAP waveforms over a fine grid of \(S\) using cubic splines. For easy numeric integration, we then assume that each segment of the MUAP between the “fine-grain” values (on the previously mentioned fine grid) of \(S\) can be approximated by a straight line segment.

\[
\mu_Z(z) \propto \sum_s G(z,s)\mu_S(s) \quad \text{(2)}
\]

Note the similarity to
\[
p(z) = \sum_s p(z|s) = \sum_s p(z|s)p(s). \quad \text{(3)}
\]

The message update rule for \(p_s(s)\) in upward direction is derived similarly.

\(\textbf{3} \text{ Addition nodes:}\) The equation \(Z = X + Y\) describes the addition node in Fig. 6. Using the sum-product rule [4], \(\mu_Z\)

\[
\mu_Z(z) \propto \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} \delta(z - x - y) \cdot \mu_X(x) \cdot \mu_Y(y) \, dx \, dy \quad \text{(4)}
\]

\[
= \int_{-\infty}^{+\infty} \mu_X(z - y) \cdot \mu_Y(y) \, dy. \quad \text{(5)}
\]

\(^1\)Here we omit an explanation of the corresponding nodes in Fig. 3.
IV. DISCUSSION AND CONCLUSION

In this paper we present novel approaches to EMG signal decomposition. In particular the joint-Gaussian message passing algorithm is described here for the first time.

The decomposition algorithm described in [5] is optimal in the sense that it finds the firing times that minimize the mean squared error between the reconstructed signal and the EMG signal. Thus it is able to decompose all of the 100 test signals correctly—even for the single channel case. In contrast, the message passing algorithms described in this paper are suboptimal since the underlying factor graphs have loops. Loopy belief propagation algorithms are iterative algorithms, which might not converge to the correct solution.

For the 100 EMG signals decomposed for this paper, the algorithm in [5] is not only optimal, it is also faster than the new algorithms presented in this paper. However, the computational effort in the optimal algorithm from [5] increases exponentially with the length of the MUAPs and the number of MUAPs. In contrast, our new message passing algorithms show a roughly linearly growing computational demand with an increase in the number of sources, the number of channels, and the lengths of the MUAPs.

Given the inherent sub-optimality of our message-passing algorithms, the results are still promising. We also expect to obtain good results for the case of non-integer phase shifts, for the unknown constituent problem, and for signals with high levels of noise. The ability of our approach to decompose multiple channels (not only two, but any number of channels) simultaneously might yield good results in the case of multi-channel EMG signals recorded with surface electrode arrays.

V. APPENDIX: JAVA PACKAGE FACTORGRAPH

We developed a Java package called factorgraph to facilitate implementing message-passing algorithms that are based on factor graphs. This package is briefly described in this section.

One basic class is called Node. The package allows to define the factor graph topology by connecting nodes. Since every node has a node function, the second basic class is called Function. Its subclasses implement specific node functions for different message types, e.g., discrete messages, Gaussian messages, or joint-Gaussian messages. We have separate classes for nodes and its functions to separate the topology of the factor graph from the algorithm that runs on it. In this way we can easily change node functions without touching the topology of the factor graph. Nodes compute messages that are sent out of these nodes along the edges of the factor graph. Instances of the class Schedule define the order of

<table>
<thead>
<tr>
<th>TABLE I</th>
</tr>
</thead>
<tbody>
<tr>
<td>DECOMPOSITION RESULTS FOR 100 SIGNALS AS IN FIG. 8. THE TABLE SHOWS THE NUMBER OF CORRECTLY DECOMPOSED SIGNALS AND THE AVERAGE DECOMPOSITION TIME IN SECONDS PER SIGNAL ON A 2GHZ INTEL PENTIUM M PROCESSOR.</td>
</tr>
<tr>
<td>---------</td>
</tr>
<tr>
<td>Discrete messages, both channels</td>
</tr>
<tr>
<td>Discrete messages, only first channel</td>
</tr>
<tr>
<td>Discrete messages, only second channel</td>
</tr>
<tr>
<td>Joint-Gaussian messages, both channels</td>
</tr>
</tbody>
</table>
When designing factor graphs for signal processing applications, the class Message deals with storing and processing various kinds of messages. The Java package is very flexible in that it can easily be extended to deal with arbitrary message formats. When designing factor graphs for signal processing applications, the class Message deals with storing and processing various kinds of messages. The Java package is very flexible in that it can easily be extended to deal with arbitrary message formats.

When designing factor graphs for signal processing applications, the class Message deals with storing and processing various kinds of messages. The Java package is very flexible in that it can easily be extended to deal with arbitrary message formats.

When designing factor graphs for signal processing applications, the class Message deals with storing and processing various kinds of messages. The Java package is very flexible in that it can easily be extended to deal with arbitrary message formats.