Journal of Electromyography and Kinesiology xxx (2011) xxx–xxx

Contents lists available at ScienceDirect

Journal of Electromyography and Kinesiology

journal homepage: www.elsevier.com/locate/jelekin

Fast implementation for EMG signal linear envelope computation

Ouriel Barzilay *, Alon Wolf

Biorobotics and Biomechanics Lab (BRML), Faculty of Mechanical Engineering, Technion – Israel Institute of Technology, Haifa, Israel

A R T I C L E I N F O

Article history:
Received 14 February 2011
Received in revised form 10 April 2011
Accepted 18 April 2011
Available online xxxx

Keywords:
Electromyogram
Linear envelope
Fast implementation

A B S T R A C T

Numerous medical and biomechanical applications involve electromyogram (EMG) signal processing in real time. Amplitude analysis of the EMG often requires computation of the signal's linear envelope. For this purpose, several methods are commonly described in the literature; however, not all match the speed requirement of real-time applications. We introduce an implementation which accelerates the computation of EMG signals linear envelopes, based on the pipeline commonly found in the literature for this kind of operation. The algorithm improves the computation's time requirement, at the expense of memory requirement, by using the result of the envelope's computation at the previous instant. This algorithm saves approximately 96% of the computation time and allows computing linear envelopes of several EMG signals in real time.

1. Introduction

Computation of the linear envelope of an electromyogram (EMG) signal is a step required by numerous applications involving the amplitude analysis of muscular activation. Although some electronic hardware devices performing part of the operation, such as band-pass filtering or rectification, have been developed (e.g. (Motion Lab Systems™)), linear envelopes are still computed in many cases by software algorithms. However, an increasing number of studies require computation of the EMG envelope in a fast and efficient way.

Examples of applications involving electromyographic signal processing in real time are abundant. For instance, several studies aim to develop and implement prosthetic robotic hands for amputees. Such prostheses may be controlled by the activation of some key muscles monitored by EMGs. The capability of this kind of system to rapidly process incoming EMG signals provides the opportunity to insert additional degrees of freedom and define further commands to prostheses: for example, finger movement and control from additional muscles could be included in a robotic hand prosthesis, allowing new abilities such as grasping or handling (e.g. in Bitzer and Van Der Smagt (2006) and Castellini and Van Der Smagt (2008)).

Another kind of applications necessitating a fast EMG signal processing are systems providing a visual feedback on the activation of muscles. For instance, the EMG-driven virtual arm described in (Manal et al., 2002) is a graphical anatomic model of the human arm controlled from processed EMG signals. In real-time graphical applications, the data acquisition and processing need to be conducted in less than 40 ms in order to obtain a smooth display.

In such applications, the computation needs to be fast and to induce the smallest possible delay. Consequently, it is essential to minimize the number of operations performed for the computation of envelopes at each instant, as a prolonged signal processing may slow down the whole system.

We introduce in this paper a fast implementation for the pipeline commonly used in the literature for linear envelope computation of EMG signals (see (Hodges and Bui, 1996) or (Gagnon et al., 2001), and Fig. 1). As pointed out in (Farina and Merrletti, 2000), the estimation of amplitude features is mostly performed by two evaluators: average rectified value and root mean square. The proposed method computes the envelope based on the average rectified value. The basic idea of the method is to exploit as much as possible the envelope computed at the previous instant, and to update only, in an efficient manner, the last part of the processed signal. This optimization results in a faster computation, allowing the determination of the envelopes of several EMG signals in real time. We would like to stress that the method described in this paper accelerates the standard algorithm without altering its accuracy.

2. Methods

2.1. Standard algorithm description

The standard algorithm used for the computation of EMG signals linear envelopes involves four successive operations. First, the signal is cleaned from irrelevant frequencies with a band-pass

* Corresponding author. Tel.: +972 778871857; fax: +972 4 8295711.
E-mail addresses: barzilay@technion.ac.il (O. Barzilay), alonw@technion.ac.il (A. Wolf).

1050-6411/$ - see front matter © 2011 Published by Elsevier Ltd.
doi:10.1016/j.jelekin.2011.04.004

JJEK 1235
4 May 2011
ARTICLE IN PRESS
No. of Pages 6, Model 5G

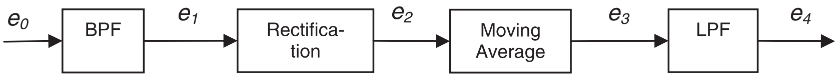

Fig. 1. Linear envelope computation pipeline.

filter (BPF), with cutoff frequencies commonly being 10 and 500 Hz. The signal is then rectified with an absolute value, before being smoothed with a moving average (MA). Finally, high frequencies are filtered out with a low-pass filter (LPF), generally with a cutoff frequency of 30 Hz (Fig. 1).

At the input of the pipeline, the raw EMG signal e_0 is given as an array whose elements are the recordings at each time sample, with the latest EMG reading at its end. This array is updated at each timestamp as a queue, popping out its first element, shifting all elements leftward, and inserting the last recording at the last position. Its size L remains constant all along the recording session. An envelope of the same size L is expected to be computed as the product of the signal processing.

The band-pass and low-pass filters can be defined as finite impulse response (FIR) filters and implemented as discrete convolutions with pre-computed kernels, dependant on the filters' properties.

The convolution operation results in an array longer than the input array, holding the greatest amount of information in its most inner part. We truncate both the first and last parts of the result vector to obtain a vector with the same size as the input vector. Alternatively, it is possible to keep the vector's last part only, consisting of the last recordings. That solution may reduce the delay induced by the processing, but only to the detriment of accuracy.

We note the result vectors by e_m, the kernels by k_m and their lengths by L_m, with m specifying the corresponding step in the pipeline (BPF: 1, rectification: 2, MA: 3, and LPF: 4). For the BPF, MA, and LPF, we assume, without loss of generality, the kernels' lengths to be odd and defined in each case as $L_m = 2n_m + 1$, where $n_m > 0$ (the MA can be considered as a particular case of convolution where the kernel's elements are all equal). Considering only odd-sized kernels makes the implementation simpler and the further development clearer. When required to make a distinction between the signals at the previous timestamp $t-1$ and the current timestamp t, the timestamp is indicated by a superscript.The standard pipeline described above can be mathematically expressed by Eqs. (1)–(4).

- BPF

$$e_1(i) = \begin{cases} \sum_{j=-n_1}^{i-1} e_0(i-j)\cdot k_1(j+n_1+1), & i \in [1, n_1] \\ \sum_{j=-n_1}^{n_1} e_0(i-j)\cdot k_1(j+n_1+1), & i \in [n_1+1, L-n_1] \\ \sum_{j=i-L}^{n_1} e_0(i-j)\cdot k_1(j+n_1+1), & i \in [L-n_1+1, L] \end{cases} \tag{1}$$

- Rectification

$$e_2(i) = |e_1(i)|, \quad i \in [1, L] \tag{2}$$

- MA

$$e_3(i) = \begin{cases} \frac{1}{n_3+i}\cdot \sum_{j=-i+1}^{n_3} e_2(i+j), & i \in [1, n_3] \\ \frac{1}{L_3}\cdot \sum_{j=-n_3}^{n_3} e_2(i+j), & i \in [n_3+1, L-n_3] \\ \frac{1}{n_3+L-i+1}\cdot \sum_{j=-n_3}^{L-i} e_2(i+j), & i \in [L-n_3+1, L] \end{cases} \tag{3}$$

- LPF

$$e_4(i) = \begin{cases} \sum_{j=-n_4}^{i-1} e_3(i-j)\cdot k_4(j+n_4+1), & i \in [1, n_4] \\ \sum_{j=-n_4}^{n_4} e_3(i-j)\cdot k_4(j+n_4+1), & i \in [n_4+1, L-n_4] \\ \sum_{j=i-L}^{n_4} e_3(i-j)\cdot k_4(j+n_4+1), & i \in [L-n_4+1, L] \end{cases} \tag{4}$$

2.2. General considerations for the computation's acceleration

The EMG raw signals of two consecutive instants are very similar: all the older array elements are shifted by one place in the new array, with the exception of the older signal's first element which disappears in the new one, and of the new EMG reading at the last position of the new array. To reduce the number of computations, we consider the mathematical relationship between the envelopes of a signal at two successive timestamps. The purpose is to minimize the number of computations by retrieving as much of the known information contained in the previous envelope's calculations as possible.

At any time, the resulting arrays from each of the pipeline's four operations are stored into the computer's memory. In the case of real-time EMG envelope computation, this is a reasonable cost, considering the small size of commonly computed envelopes compared to the average storage capacity of today's computer. We show that a considerable amount of calculations can be saved by this method.

The convolution required in the computation of element $e_m(i)$ involves $2n_m + 1$ elements from the input array, from $e_{m-1}(i-n_m)$ to $e_{m-1}(i+n_m)$. If none of these $2n_m + 1$ input array's values has been modified since the previous timestamp, the matching result vector element is unchanged from the previous timestamp result vector and can be inherited with no additional computation.

However, the first n_m elements, as well as the last n_m ones, do not have as many elements to their left and right sides and their computation is less accurate. Nevertheless, whereas the last part really lacks future information, the first part involves elements known in the past time samples and were only discarded from storage. They were used in the previous timestamps convolution operations, together with elements that have not been modified. Consider, for instance, element number $n_1 + 1$ in e_1. Its computation involves elements number 1 to $2n_1 + 1$ in e_0. At the next timestamp, element number n_1 in e_1 involves exactly the same operations on the same elements, but lacks the discarded first element of e_0 at the previous timestamp. It is thus more accurate and efficient to set that element's value as the already known $e_1(n_1 + 1)$ from the previous timestamp. In a general manner, the first part with length n_m is derived from the previous timestamp's more precise computation on the inner part of the vector, with no further calculation. In that way, only the computation of the first-timestamp envelope requires the total number of calculations defined in the pipeline whereas, at the following instants, the envelope computation can be accelerated.

The last element of e_m obtainable from the previous timestamp is the element involving only elements of e_{m-1} that have not been previously modified at the current timestamp.

2.3. Band-pass filter

As mentioned in the previous section, the initial and central parts of the BPF result vector can be obtained from the previous time-

Please cite this article in press as: Barzilay O, Wolf A. Fast implementation for EMG signal linear envelope computation. J Electromyogr Kinesiol (2011), doi:10.1016/j.jelekin.2011.04.004

stamp. The only new element in e_0 is the last one. From the convolution formula (Eq. (1)), only e_1's elements on $[L - n_1, L]$ involve that element in their computation. These elements need to be recalculated. All the elements up to index $L - n_1 - 1$ are inherited, according to the previous subsection's last consideration (Eq. (5)).

The calculations performed on the last interval at the previous timestamp can be used again. According to Eq. (1), the interval $[L - n_1 + 1, L]$ corresponds to the elements of e_1 that use less than L_1 elements of e_0 in their computation. Those elements, if given a new input element after the last one, will be able to complete the convolution sum once step further, by adding to their value the new element multiplied by the corresponding kernel element. That is precisely what happens at the following timestamp. There, all of e_0's elements are shifted to the left and one additional element is inserted. Elements of e_1 on $[L - n_1, L - 1]$ can be obtained from the elements of previous-timestamp e_1 on $[L - n_1 + 1, L]$ with the adequate corrective term (Eq. (5)). The last element of e_1 requires a full computation over $n_1 + 1$ elements (Eq. (5)).

$$e_1^t(i) = \begin{cases} e_1^{t-1}(i+1), & i \in [1, L - n_1 - 1] \\ e_1^{t-1}(i+1) + e_1^t(L) \cdot k_1(i - L + n_1 + 1), & i \in [L - n_1, L - 1] \\ \sum_{j=0}^{n_1} e^t(L - j) \cdot k_1(j + n_1 + 1), & i = L \end{cases} \tag{5}$$

2.4. Rectification

The rectification is performed by a simple absolute value on e_1's elements. Although this operation is almost costless, it was performed at the previous timestamp for the elements of e_1 that have not changed, and can thus be partially inherited (Eq. (6)).

$$e_2^t(i) = \begin{cases} e_2^{t-1}(i+1), & i \in [1, L - n_1 - 1] \\ |e_1^t(i)|, & i \in [L - n_1, L] \end{cases} \tag{6}$$

2.5. Moving average

Similar to the band-pass filter case, the inner part of the MA result vector e_3 is the most relevant, but the first part can be quickly replaced by the inner part elements from previous instants. The last element of e_2 to be modified at the current timestamp is at location $L - n_1$. Therefore, the first element of e_3 requiring a new computation is at location $L - n_1 - n_3$.

Elements on the last interval can be computed recursively, starting from the last element. That element is computed as the mean of the last $n_3 + 1$ elements of e_2 (Eq. (7)). The previous element computes the average of the same elements and one additional element at location $L - n_3 - 1$. We can thus retrieve the sum of the $n_3 + 2$ last elements of e_2 by multiplying $e_3^t(L)$ by the number of elements of which the average was computed, i.e. $n_3 + 1$. To obtain the final average, it suffices then to add element number $L - n_3 - 1$ and to divide the sum by the new total number of elements, $n_3 + 2$. Every element of e_3 on $[L - n_3 + 1, L - 1]$ is efficiently computed in the same manner (Eq. (7)).

On $[L - n_1 - n_3, L - n_3]$, the elements average L_3 elements of e_2. The difference between two consecutive elements on that interval is that the averaging window moves by one index. Instead of computing the whole average for element i of e_3, we correct the following element $i + 1$ already computed, by adding element $i - n_3$, and subtracting from it element $i + n_3 + 1$, not included in the present averaging. This corrective term must be normalized by L_3 (Eq. (7)).

$$e_3^t(i) = \begin{cases} e_3^{t-1}(i+1), & i \in [1, L - n_1 - n_3 - 1] \\ e_3^t(i+1) + \frac{1}{L_3} \cdot (e_2^t(i - n_3) - e_2^t(i + n_3 + 1)), & i \in [L - n_1 - n_3, L - n_3] \\ \frac{1}{w_i} \cdot ((w_i - 1) \cdot e_3^t(i+1) + e_2^t(i - n_3)), & i \in [L - n_3 + 1, L - 1] \\ \frac{1}{n_3+1} \cdot \sum_{j=-n_3}^{0} e_2^t(L + j), & i = L \end{cases} \tag{7}$$

where $w_i = n_3 + L - i + 1$

2.6. Low-pass filter

The first part $[1, L - n_1 - n_3 - n_4 - 1]$ of e_4, only involving unchanged elements of e_3, can be inherited from the previous timestamp. The rest of the elements is computed according to the convolution formula (Eq. (8)).

$$e_4^t(i) = \begin{cases} e_4^{t-1}(i+1), & i \in [1, L - n_1 - n_3 - n_4 - 1] \\ \sum_{j=-n_4}^{n_4} e_3^t(i - j) \cdot k_4(j + n_4 + 1), & i \in [L - n_1 - n_3 - n_4, L - n_4] \\ \sum_{j=i-L}^{n_4} e_3^t(i - j) \cdot k_4(j + n_4 + 1), & i \in [L - n_4 + 1, L] \end{cases} \tag{8}$$

3. Results

A comparison between the number of computations (additions and multiplications) required in both the full and fast implementations is provided in Table 1. The results presented there depend on each of the kernels' sizes and on the size of the input vector L. For clarity, we chose to display the case where all n_m are equal to a given n. This assumes the same size for all three kernels and is a pertinent approximation in comparison with the kernel sizes commonly used in practice for this application. For the exact number of computations or the separate numbers of additions and multiplications, the reader is invited to contact the corresponding author. We also remind the reader that the kernel sizes are typically much smaller than the input vector size ($n << L$). The gain in computation for typical input arrays and kernel sizes is presented in Fig. 2.

Most of the fast implementation's efficiency relies on the retrieval of the envelope's unmodified part from the previous timestamp. Nevertheless, if we solely consider the interval $[L - n_1 - n_3 - n_4, L]$, corresponding to the envelope's part requiring an update, the fast implementation saves yet $(33n^2 - n)/2 - 1$ operations (Table 1), or typically 56–60% of the operations (Fig. 3).

Both methods have been implemented in Matlab™. Their runtime on a PC Intel Core i5 750@2.67GHz, 4Gb RAM is presented for different values of n and L in Table 2. The results fit the theoretical expectation. Similarly, a gain of 56–60% has been obtained for comparison on the last interval only.

4. Discussion

We have presented in this paper an efficient way to implement the computation of linear envelopes for EMG signals, based on the knowledge of the envelope at the previous time sample. The algorithm exploits the properties of convolution, used by the FIRs, to accelerate the envelope computation. The scope of this paper may be extended to the use of infinite response filters (IIR), often used in this application, with the generalized convolver developed by Wiklund and Knutsson (1995). Furthermore, the kernels used in the described pipeline are generally symmetric and contain several zeros and, by taking advantage of these properties, the implementation could be accelerated further.

Please cite this article in press as: Barzilay O, Wolf A. Fast implementation for EMG signal linear envelope computation. J Electromyogr Kinesiol (2011), doi:10.1016/j.jelekin.2011.04.004

Table 1
Comparison of the number of operations required in both methods.

	Total numbers of operations	
	On the whole interval: $[1, L]$	On the last interval: $[L - n_1 - n_3 - n_4, L]$
Full computation	$10Ln + 3L - 5n^2 - n$	$\frac{55}{2}n^2 + \frac{33}{2}n + 3$
Fast implementation	$11n^2 + 17n + 4$	$11n^2 + 17n + 4$
Number of saved computations	$10Ln + 3L - 16n^2 - 18n$	$\frac{33}{2}n^2 - \frac{1}{2}n - 1$

Table 2
Computation time of an EMG envelope.

	n	15	50	100	200	400
	L					
Full implementation [msec]	2000	12.8	27.7	48.6	88.8	161.8
	4000	25.7	56.1	98.2	181.0	339.8
	8000	51.4	111.9	197.8	367.0	699.8
	10,000	64.5	139.7	246.9	460.1	878.0
Fast implementation [msec]	2000	0.26	0.95	2.95	10.6	41.1
	4000	0.39	1.06	3.04	10.8	41.1
	8000	0.66	1.32	3.31	11.1	41.6
	10,000	0.79	1.45	3.56	11.3	42.0
Gain in time [%]	2000	98.0	96.6	93.9	88.1	74.6
	4000	98.5	98.1	96.9	94.0	87.9
	8000	98.7	98.8	98.3	97.0	94.1
	10,000	98.8	99.0	98.6	97.5	95.2

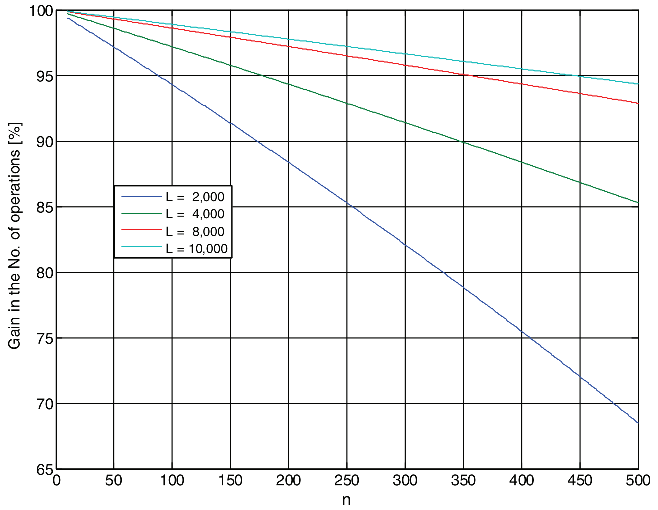

Fig. 2. Gain in the number of operations saved on the whole interval.

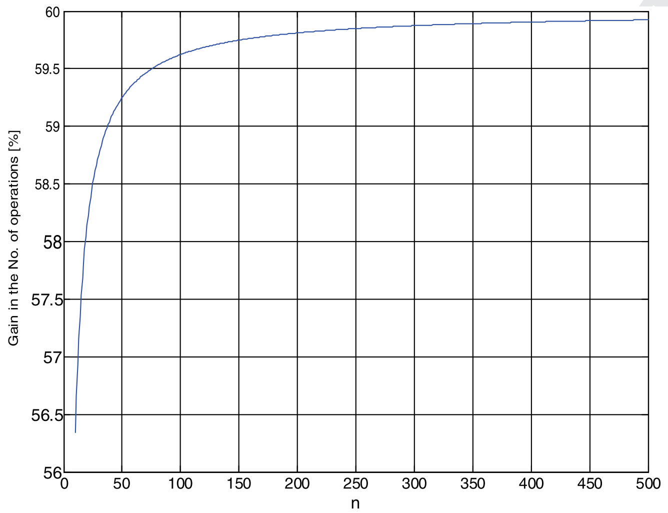

Fig. 3. Gain in the number of operations saved on the last interval on which the envelope computation requires an update.

This method may find its application in every study involving knowledge on the amplitude of muscular activity in real-time, and can save precious time for further processing, such as classification or pattern matching (e.g. (Naik et al., 2006)). Several studies on robotic prosthetics (e.g. (Bitzer and Van Der Smagt, 2006)) and exoskeletons (e.g. (Kiguchi et al., 2004)) controlled by muscles could gain from the method's computation speed in the EMG signal processing.

Some specific applications, such as the virtual arm of (Manal et al., 2002), use the amplitude of muscle activation for immediate visual feedback to the user. This is the framework in which the method has been developed. In (Barzilay and Wolf, 2009), we present a virtual reality platform for patient-specific neuromuscular rehabilitation. In that platform, the user bears reflectors tracked by (Vicon™) motion capture system and 3D goggles in which a virtual task is displayed in the form of a game. A virtual replica of the user's arm is displayed in the virtual environment. The task may be, for instance, to track a floating ball with one's virtual hand consistently displayed according to the recorded kinematics. The user continuously receives an audiovisual feedback on the distance to the target. Additionally, activations of the muscles involved in the task (biceps and triceps in the case of arm rehabilitation) are represented by sweat drops coming out of the sleeve in the virtual environment, proportionally to the amplitude of the corresponding EMG signal envelopes. The rationale for providing EMG biofeedback lays in the fact that augmented feedback on the performance has been proved to enhance motor learning (Adams et al., 1977), and thus physical therapy. The system employs then artificial neural networks to adapt the task to the subject's performance.

The three-dimensional display in the goggles requires a minimum frame rate of 60 Hz (30 Hz for each eye, alternately). In order to display the activations of several muscles in real time, the EMG signal processing needs to be performed quickly. Since the speed attained with the naïve implementation was not sufficient, the processing had to be accelerated with the proposed method.

The described implementation allows digitally computing the envelope of an EMG signal in less than 10 ms for a standard raw EMG array and kernel sizes. To the best of our knowledge, no study describing a faster computation has been published in the literature. This computation speed allows an almost immediate EMG feedback that could be used in the analysis or visual display of the activations of several muscles in real time.

Conflict of interest

All the authors were fully involved in the analysis and manuscript preparation. The manuscript has not been submitted for publication elsewhere. The authors have no conflict of interest to declare or acknowledgment to state.

References

Adams JA, Gopher D, Lintern G. Effects of visual and proprioceptive feedback on motor learning. J Mot Behav 1977;9:11–22.
Barzilay O, Wolf A. An adaptive virtual system for neuromuscular rehabilitation. In: IFMBE proceedings of world congress on medical physics and biomedical engineering, Munich, Germany. 2009; 25(4): 1291–94.
Bitzer S, Van Der Smagt P. Learning EMG control of a robotic hand: towards active prostheses. In: Proceedings of international conference on robotics and automation, Orlando. 2006. pp. 2819–23.

Please cite this article in press as: Barzilay O, Wolf A. Fast implementation for EMG signal linear envelope computation. J Electromyogr Kinesiol (2011), doi:10.1016/j.jelekin.2011.04.004

Castellini C, Van Der Smagt P. Surface EMG in advanced hand prosthetics. Biol Cybern 2008;100:35–47.
Farina D, Merrletti R. Comparison of algorithms for estimation of EMG variables during voluntary isometric contractions. J Electromyogr Kinesiol 2000;10(5):337–49.
Gagnon D, Lariviere C, Loisel P. Comparative ability of EMG, optimization, and hybrid modelling approaches to predict trunk muscle forces and lumbar spine loading during dynamic sagittal plane lifting. Clin Biomech 2001;16:359–72.
Hodges P, Bui B. A comparison of computer-based methods for the determination of onset of muscle contraction using electromyography. Electroenc Clin Neurophysiol 1996;101:511–9.
Kiguchi K, Tanaka T, Fukuda T. Neurofuzzy control of a robotic exoskeleton with EMG signals. IEEE Trans Fuzzy Systems 2004;12(4):481–90.
Manal K, Gonzalez R, Lloyd D, Buchanan TS. A real-time EMG-driven virtual arm. Comput Biol Med 2002;32:25–36.
Motion Lab Systems. Available from: <http://www.motion-labs.com/>.
Naik GR, Kumar DK, Singh VP, Palaniswami M. Hand gestures for HCI using ICA of EMG. HCSNet Workshop on the Use of Vision in HCI 2006:67–72.
VICON Motion Systems. Available from: <http://www.vicon.com>.
Wiklund J, Knutsson H. A generalized convolver. In: Proceedings of the 9th Scandinavian conference on image analysis, Uppsala, Sweden, June 1995. SCIA.

Ouriel Barzilay received his B.Sc. and M.Sc. degrees in Mechanical Engineering at the Technion I.I.T. He is currently a Ph.D. candidate in the Biorobotics and Biomechanics Laboratory (BRML) at the Technion I.I.T. His fields of interest include biorobotics, kinematics, computer vision, computational geometry, and artificial intelligence.

Alon Wolf, PhD, earned all his academic degrees from the Faculty of Mechanical Engineering at Technion-I.I.T. In 2002 he joined the Robotics Institute of Carnegie Mellon University and the Institute for Computer Assisted Orthopaedic Surgery as a member of the research faculty. He was also an adjunct Assistant Professor in the School of Medicine of the University of Pittsburgh. In 2006 Dr. Wolf joined the Faculty of Mechanical Engineering at Technion, where he founded the Biorobotics and Biomechanics Lab (BRML). The scope of work done in the BRML provides the framework for fundamental theories in kinematics, biomechanics and mechanism design, with applications in medical robotics, rehabilitation robotics, and biorobotics, such as snake robots.