ELSEVIER

Computer Methods and Programs in Biomedicine 69 (2002) 225–236

Computer Methods and Programs in Biomedicine

www.elsevier.com/locate/cmpb

Software design for analysis of multichannel intracardial and body surface electrocardiograms

Mark Potse [a,*], André C. Linnenbank [b], Cornelis A. Grimbergen [a,c]

[a] *Medical Physics Department, Academic Medical Center (AMC), University of Amsterdam, P.O. Box 22700, 1100 DE Amsterdam, The Netherlands*

[b] *Experimental Cardiology Department, Academic Medical Center (AMC), University of Amsterdam, Amsterdam, The Netherlands*

[c] *Control Department, Faculty of Design, Construction, and Production, Delft University of Technology, Delft, The Netherlands*

Received 27 February 2001; received in revised form 26 November 2001; accepted 24 January 2002

Abstract

Analysis of multichannel ECG recordings (body surface maps (BSMs) and intracardial maps) requires special software. We created a software package and a user interface on top of a commercial data analysis package (MATLAB) by a combination of high-level and low-level programming. Our software was created to satisfy the needs of a diverse group of researchers. It can handle a large variety of recording configurations. It allows for interactive usage through a fast and robust user interface, and batch processing for the analysis of large amounts of data. The package is user-extensible, includes routines for both common and experimental data processing tasks, and works on several computer platforms. The source code is made intelligible using software for structured documentation and is available to the users. The package is currently used by more than ten research groups analysing ECG data worldwide. © 2002 Elsevier Science Ireland Ltd. All rights reserved.

Keywords: Electrocardiography; Body surface mapping; Cardiac mapping; Program documentation; Literate programming

1. Introduction

Electrocardiographic body surface maps (BSM) and high-resolution intracardial maps typically consist of many signals sampled simultaneously. Recordings with over 500 leads have been reported [1]. Our group records mapping data with 64 up to 247 leads which are sampled with frequencies ranging from 0.5 to 4 kHz and 8, 14, and 16-bit resolution at bit steps of 0.73 up to 40 μV [2]. Datafile sizes range from 48 kB to circa 300 MB. A variety of recording situations is employed, resulting in intracardial electrograms, surface electrograms, electrograms recorded in cell cultures, and combined multichannel endocardial and BSM recordings [3,4].

Processing methods for multichannel ECG recordings differ strongly from those for single-

* Corresponding author. Tel.: +31-20-566-5363; fax: +31-20-691-7233

E-mail addresses: m.potse@amc.uva.nl (M. Potse), a.c.linnenbank@amc.uva.nl (A.C. Linnenbank).

PII: S0169-2607(02)00014-7

lead or standard 12-lead ECGs. The large number of channels makes visual inspection of all waveforms almost impossible, but does enable spatial representation of parameters such as potential or activation time, using, for example, pseudocolour maps. In addition, computer algorithms of varying complexity are employed to convert the large amounts of data into concise diagnostics. Standard ECG-analysis tools are therefore inadequate for multichannel ECG analysis.

2. Background

Analysis of multichannel ECG data requires special software. Due to the variety in research purposes and available hardware, many custom software packages were previously created in laboratories involved in electrocardiographic research. However, it is advantageous to attempt to make such software as general as possible, making it applicable at many different laboratories. This could prevent repetition of effort, and make implementation of difficult algorithms and a sophisticated user interface more worthwhile. We shall analyse some requirements for such software, and present a software package that attempts to fulfil these requirements and to be general enough to be useful for several research groups.

3. Design considerations

Interactive processing is required in various cases. For example, if analysis of BSM recordings is used on-line to guide catheter ablation of an arrhythmia, a computer program can perform all computations, but the physician has to select the QRS complex of interest [5,6]. An interactive system is also desired if data is processed off-line but with input from a human expert. Such a system should, for example, be able to apply baseline corrections, integrate over time, and produce maps of potentials and integral values in several different styles (Fig. 1). It has to detect activations in intracardial recordings and QRS complexes in BSMs, and allow the user to correct detection results interactively. For research purposes, it is necessary that new algorithms can be incorporated in a user interface package with little effort, so that they can be tested interactively.

There are also circumstances where batch processing of data is more appropriate. For example, to create and test a new algorithm it is often necessary to go through the cycle of changing and testing a program several times, while testing a large number of recordings. The creation of programs for such jobs is easier if separate routines are available for basic tasks such as loading of recordings from file, baseline correction, and acti-

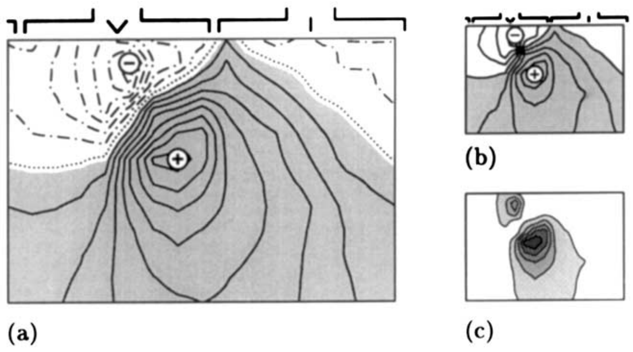


Fig. 1. Examples of BSMs. (a) Large format, used to show details clearly. Potential values are indicated with isopotential lines; solid lines are used for the positive area, dash–dotted lines for the negative area, and a dotted line for the zero level. The area with positive potentials is shaded. Maximum and minimum are indicated. Above the map, the positions of shoulders, sternum, and spine are indicated schematically. (b) Economy-size map. All contour lines are drawn in a solid style because their curvature is too high for display in a dashed or dotted style. (c) Special format for unsigned data, such as squared potentials. Both contour lines and grey values are used to display the values. In this map, the schematic torso anatomy was omitted as is often done when many maps are shown in a single display.

vation detection. Note that these are tasks that are also performed in an interactive analysis program.

Interactive and noninteractive software thus have many requirements in common. Creating them in the same programming environment has the advantage that basic routines can be shared, and that data interchange between interactive and noninteractive software is more easily achieved than when both are completely separate programs. It is expedient if basic routines, for tasks like sorting, differentiation, principal components analysis, and notably graphics presentation are already present in the programming environment. The same applies to user interface routines; for the creation of interactive software, the need of an easily programmable user interface is obvious. Batch programs can also benefit from user interface routines, particularly in the testing stage. For example, when inspecting outliers in a scatter plot, it is practical to just click on a data point with a pointer device to view the corresponding ECG data. A convenient way to achieve this is by making calls to an existing user interface package in the same programming environment. The common environment is a clear advantage in such cases.

Software documentation and source control are indispensable, particularly for experimental software. Good documentation facilitates maintenance of software. Source control, by which we mean the automated logging of changes to software, can provide crucial information to the programmer who wants to change an existing program. Such a system can also provide file locking services, which prohibit multiple programmers overwriting each other's changes by accident.

In conclusion, research in multichannel electrocardiography benefits from a software package that has both a user interface and a programmer interface, provides basic as well as sophisticated ECG analysis routines, has access to existing graphical, mathematical, and user interface programming libraries, and is well documented and organised. We describe how we created and worked with a package that fulfils all these requirements. We shall not discuss the algorithms that are used, since the purpose of our software is to provide a framework for the implementation of algorithms, and not to make a selection from them. In fact, several alternative methods were implemented for particular tasks, so that users can make their own choices.

We chose to create our software under an interactive programming environment (IPE). By IPE we mean an interactive program that can execute typed commands and programs or 'script files'. An IPE can include libraries of mathematical, graphical, user interface, and other functions. It can be platform-independent in the sense that the same commands have equivalent results on different platforms. It is an advantage if an IPE can communicate with external programs or call functions written in a system programming language to allow efficient handling of bottleneck operations. It is also advantageous if the IPE programs are plain text files, because this facilitates documentation and source control. 'Visual' programming languages and IPEs with a graphic programming interface are therefore less suitable for our purposes.

An IPE can also be an extremely versatile debugging program, because it allows access to internal data at any time, can use its graphical capabilities for display of debugging output, and may permit substitution of program parts without restarting the whole program.

4. System description

4.1. Choice of platform

The IPE we used was MATLAB (The MathWorks Inc., Natick, MA, USA), a program that can apply mathematical operations to matrices, has functions for graphics and user interfaces built-in, and includes libraries of mathematical and graphical routines. Apart from platform-specific bugs, MATLAB programs run without modification on several operating systems and computer architectures. This allowed us to develop the software and do our own experimental work on a UNIX system, while also providing code for MS-windows and Apple Macintosh systems.

Interpreted programs run slower than system programs. Modern computers are capable of running an interpreter fast enough for a user interface or small computations, but for bottleneck operations such as baseline correction of a multilead ECG it could be necessary to write a function in a compiled language and use this function in the IPE. The IPE that we use supports dynamic linking of functions programmed in C. We used this feature selectively for often-used routines that caused significant delay when programmed in the IPE language.

The C functions are a limitation to platform-independence because they must be compiled specifically for each platform. We made our C sources compilable for all platforms used by programming strictly conforming to the ANSI standard, without using compiler-specific or platform-specific libraries, whenever possible. We had to resort to platform-specific code at two occasions: for directory reading and for TCP/IP communication.

4.2. Documentation and source control

For a program to be understood by humans, and not only by compilers, program documentation is indispensable. We chose the 'literate programming' technique [7,8] to document our software. A 'literate' program is written in the style of a monograph and consists of small pieces of code, which are written in the programming language(s) of choice—in our case MATLAB and C. These code fragments are called 'refinements' and can be partially or completely defined in terms of each other. Each refinement comes with a documentation part written in a document formatting language—in our case LATEX. This means that the program documentation can include mathematical formulae, graphics, references, indexes, etc. The refinements can be presented in any order the author wishes; by application of refinements in other refinements the code is ordered for the compiler's purposes. Because the documentation parts can be large compared with the code parts, without obscuring the program structure, it is possible to give a thorough description of the code, explaining not only what it does, but also why it does so. The latter is often neglected in traditional software documentation but highly important for scientific and experimental software. Literate programming systems allow the definition of multiple program files in a single document. This facilitates comprehensive documentation of interrelated program files. A small program, illustrating some of these aspects, is shown in Fig. 2.

A literate programming system consists of two filter programs. One filter extracts the program parts and puts them in the specified order for the compiler (Fig. 3). A second filter program translates the source into document formatting instructions, taking care of prettyprinting (syntax highlighting and standard indentation), and copies the documentation parts, which are already written in the document formatting language (Fig. 2). A well-known example of literate programming is the published source code of TEX [10].

Our C-language functions were documented with the CWEB system [11]. For the MATLAB programs we created a literate programming system, called MWEB [12]. We also made provisions to include CWEB and MWEB programs in LATEX documents [14], and to enforce a uniform layout of the circa 90 program documents. The documentation could be printed and was also made available on our internal web server as hypertext in PDF format.

Additional tools were the revision control system (RCS) [15], which provided for automated revision archiving, the standard UNIX make program, and a few small programs written in a scripting language (PERL [16]) to control the compilation process.

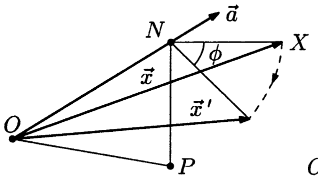
4.3. Structure

To accommodate both interactive use and batch processing we created two packages: a low level toolbox, called MAPLIB, and a user interface, called MAPLAB, which was created using MAPLIB routines. The toolbox contains all routines that interactive and noninteractive programs can share; it can also be used independently for batch jobs and for user-created additions. Communication of data between MAPLAB and batch jobs is

1. Introduction. A transform matrix for rotating three-dimensional vectors may be specified by a rotation axis and an angle. This function takes a 3-element vector $\vec{a}$ and an angle ϕ, and returns the transformation matrix R.

```
format phi TeX        % makes phi show up as φ
⟨rot3d.m 1⟩ ≡
  % ROT3D(vec,ang)  3d rotation  R(vec,ang)
  % $RCSfile: rot.web,v $  $Revision: 5.3 $
  function R = rot3d(vec, φ)
    ⟨check arguments 4⟩
    ⟨compute R(a, φ) 3⟩
```

2. Algorithm. Suppose we are rotating $\vec{x}$ around a unit vector $\vec{a}$ by an angle ϕ. Let point N be the projection of $\vec{x}$ on $\vec{a}$ and define point P by $\vec{NP} = \vec{a} \times \vec{x}$:



$$\vec{NP} = \vec{a} \times \vec{x}$$
$$\vec{ON} = \vec{a}(\vec{a} \cdot \vec{x}) = \vec{a}\vec{a}^T\vec{x}$$
$$\vec{NX} = \vec{x} - \vec{ON} = (\underline{1} - \vec{a}\vec{a}^T)\vec{x}$$

Then

$$\begin{aligned} \vec{x}' &= \vec{ON} + \vec{NX}\cos\phi + \vec{NP}\sin\phi \qquad (1) \\ &= \left[\vec{a}\vec{a}^T + (\underline{1} - \vec{a}\vec{a}^T)\cos\phi\right]\vec{x} + (\vec{a} \times \vec{x})\sin\phi \end{aligned}$$

expresses the new vector $\vec{x}'$ in terms of $\vec{x}$ and ϕ.

3. Using, Algorithm due in part to Faux and Pratt [9],

$$A = a_i\varepsilon_{ijk} = \begin{pmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{pmatrix} \qquad (2)$$

the cross product $\vec{a} \times \vec{x}$ can be turned into a matrix multiplication so that we can express (1) in the form $\vec{x}' = R\vec{x}$ with

$$R = \vec{a}\vec{a}^T + (\underline{1} - \vec{a}\vec{a}^T)\cos\phi + A\sin\phi \qquad (3)$$

```
⟨compute R(a, φ) 3⟩ ≡
  A = [0, −a(3), a(2);  a(3), 0, −a(1);  −a(2), a(1), 0];
  aat = a*a';        % 3 × 3 matrix
  R = aat + (eye(3, 3) − aat)*cos(φ) + A*sin(φ);
This code is used in section 1.
```

4. We expect a 3-vector and a scalar as input arguments. The axis vector is normalized to make sure that R is orthonormal, and reshaped into a column vector, so that $a*a'$ is a matrix.

```
⟨check arguments 4⟩ ≡
  if nargin ≠ 2
    error('two␣arguments␣needed');
  end
  if length(vec(:)) ≠ 3
    error('first␣argument␣must␣be␣3d␣vector');
  end
  if length(φ) ≠ 1
    error('second␣argument␣must␣be␣a␣scalar');
  end
  a = reshape(vec/norm(vec), 3, 1);        % column
This code is used in section 1.
```

Fig. 2. A tiny literate program written in the MATLAB language, which shows some of the possibilities of a literate programming system. It consists of three sections. Section 1 defines the file rot3d.m. This file begins with two comment lines, the second of which contains a source file identification that is automatically updated by the revision control system. Sections 2 and 3 describe the algorithm and section 3 defines a refinement named '⟨compute $R(\vec{a}, \phi)$ 3⟩', containing the actual computation. Section 4 defines a refinement named '⟨check arguments 4⟩'. Sections 3 and 4 are used in section 1. The literate programming system typesets keywords like **function** in bold type and identifiers in italic type, and ensures correct indentation. The table of contents, identifier index, and list of refinements were omitted.

done with IPE variables, and with files if the data have to be passed between invocations of the IPE.

The user interface package consists of several tens of functions, whose syntax is documented in the source files. This allows experienced programmers to make calls to the user interface from experimental programs, for example to inspect internal data of these programs with existing tools. It is also possible to let user-created programs be called from MAPLAB. This is done by a general hook mechanism. Programs that run in the IPE can add a code fragment to one of MAPLAB's hooks. This code fragment is then executed whenever the hook is activated, for example, every time a new file is loaded, an activation marker is moved, or baseline adjustment is performed. Some of MAPLAB's programs are activated by this hook mechanism as well.

Two new file formats are used by our software: a datafile format for storage of ECG data and a metafile format for descriptions of data (we consider it undesirable to store such descriptions in the datafile itself because the integrity of the datafiles, which may contain clinical data, must be guaranteed, and because datafiles are often stored on a read-only medium such as CD-ROM).

A metafile contains the name of a datafile and a set of time instants, such as begin and end of a QRS complex and activation times, which refer to the datafile. The aim of the metafile format was to store descriptions of datafiles on disk. From the user's perspective, metafiles can be loaded in the MAPLAB package in the same way as datafiles. The software will then load the corresponding datafile and place the time markers (Fig. 5) at the positions indicated in the metafile. This makes it convenient to load, for example, previously determined time instants for baseline correction together with the datafile. Our metafiles are often used to postprocess hand-made analyses in a batch job. In addition to the datafile reference and time instants, a metafile can contain a checksum to allow verification of the identity of the datafile, a timestamp, and an identification of the researcher and the analysis software. The software finds a datafile by the name and location recorded in the metafile. If the file is not found at the specified location, for example because metafile and datafile were transported to a computer with a different directory structure, the software will search for it by means of a configurable search path.

```
% ROT3D(vec,ang)  3d rotation  R(vec,ang)
% $RCSfile: rot.web,v $  $Revision: 5.3 $
function R  =  rot3d(vec, phi)

   if nargin~=2
     error('two arguments needed');
   end
   if length(vec(:))~=3
     error('first argument must be 3d vector');
   end
   if length(phi)~=1
     error('second argument must be a scalar');
   end
   a = reshape(vec/norm(vec), 3,1); % \rlap{column}

   A = [0,-a(3),a(2);a(3),0,-a(1);-a(2),a(1),0];
   aat = a * a';     % $3\times 3$ matrix
   R = aat + (eye(3,3)-aat) * cos(phi) + A * sin(phi);
```

Fig. 3. Contents of the file rot3d.m: a MATLAB program created automatically by a filter program from the MWEB source. The typeset documentation, created from the same source by another filter program, was shown in Fig. 2. One can observe that the program parts have been tangled together. Comments, which start with a percent sign, are copied literally from the web file; the LATEX code for the comment in the pre-last line is visible.

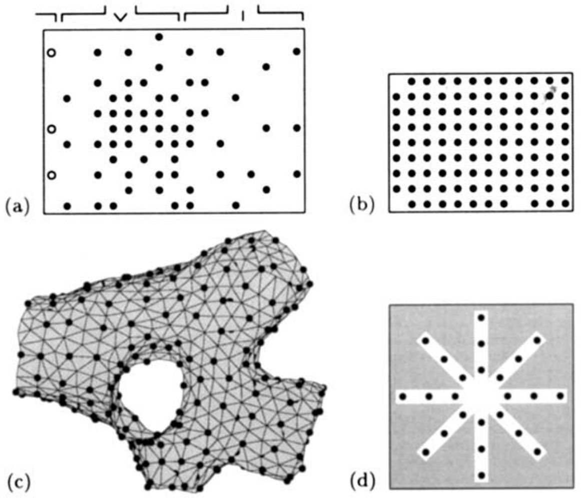


Fig. 4. Examples of electrode configurations. Dots indicate electrode positions. (a) Sixty-four-channel BSM configuration. This configuration was used for the maps in Fig. 1. This is a cyclic grid; the rightmost column of electrodes is repeated on the left, indicated by open circles. The vertical interelectrode distance s is approximately 3 cm; the electrode diameter d is 1 cm. (b) A 105-channel configuration for intracardial maps, $s = 0.8$ mm, $d = 0.1$ mm. (c) A 3-D multielectrode with 197 terminals that fits in the left atrium of a dog, $s \approx 1$ mm, $d = 0.1$ mm [19]. (d) A star-shaped grid used for cell cultures; $s = 300$ µm, $d = 24$ µm [28]. Cells grow only in the white area.

The two file formats are designed according to the IFF metaformat, which also underlies, for example, Amiga's ILBM format. This design allows extensions (such as new compression types [17], or additional parameters) with backward and forward compatibility.

4.4. Generality

The MAPLAB package was designed to cope with every possible electrode configuration, including BSM configurations [18] and various grids for intracardial recordings (Fig. 4). Regular and irregular one-, two-, and three-dimensional (3-D) shapes [19] can be handled. For each configuration, an *electrode definition file* was created, which specified the positions of the electrodes, the corresponding lead numbers, and the type of lead (surface, endocardial, epicardial, etc.). Users can create such files when they employ new grids or need to make ad hoc changes to compensate for wiring errors. The software uses this configuration information to display maps, and takes into account the differences between types of leads. For example, it does not automatically perform activation detection in surface leads, which would be meaningless.

Potential maps, integral maps, activation time maps, and maps of other data can be displayed in several different formats. It is possible to indicate data values for electrodes with an interpolated or noninterpolated pseudocolour map. In addition, data values can be printed at the electrode positions, and several contour algorithms are available to draw, for example, activation isochrones (Fig. 5). Data for pseudocolour maps, texts, and contours can be chosen independently; this facilitates comparison of e.g. potential maps to an activation (isochrone) map.

4.5. Features

The MAPLAB package is intended as a basis for development and implementation of algorithms. It includes basic, established, and experimental algorithms. Among the implemented routines are various filters, zeroth- and first-order baseline correction, QRS detection [20], arrhythmia localisation by body surface mapping using either databases [5,21] or a continuous algorithm [6], a configurable activation-detection algorithm, computation of nondipolar content [22,23], signal alignment and signal averaging, and various display methods discussed above. As mentioned in Section 3, we shall not go into details about the algorithms.

Important features are the possibility to edit results of automatic analyses, such as activation detection, by hand quickly and easily, and the provisions for storing such edited results for later processing by batch jobs.

The software package creates several windows on the screen; each window displays the data or a part of the data in a different way, and contains user interface controls for performing tasks relevant to the kind of display. For example, the 'Channel' window (Fig. 5) shows one or more ECG tracings and contains a user interface for selection of time instants, baseline correction, filtering, etc. In contrast to this temporal display, the 'Map' windows show data in the spatial domain, such as potential maps and activation maps. Examples are shown in Fig. 5. Time instants for maps are selected in the 'Channel' window, and working channels can be selected conversely from Map windows by clicking at electrode positions.

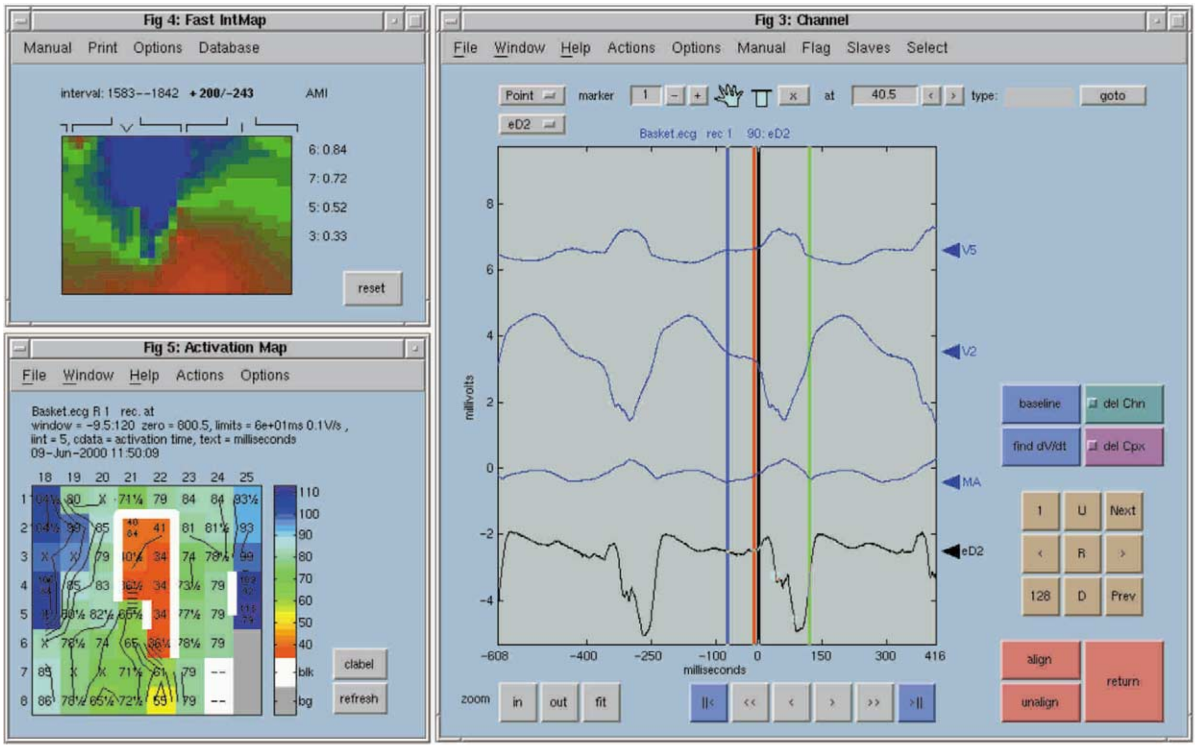
We chose to have multiple instances of some windows, while other windows could be one-of-a-kind. For example, multiple 'Map' windows can show potential maps and activation maps simultaneously. In contrast, only one 'Channel' window was needed because it can show as many simultaneous ECG traces as the user wishes, and because it is usually preferred to display all ECG traces in a single frame.

Another window that can have multiple instances is the 'Localizer' window, which shows results of arrhythmia localisation obtained with BSM. Multiple instances of this window were desirable because it is useful to see results for different cavities or by different algorithms at the same time (Fig. 6). Localisation is performed either by comparing QRS integral maps to database maps [21] or by a continuous algorithm [6]. Databases exist for the human ventricles and atria [21,24,25]. The Localizer windows can present localisation results in several different diagrams, for both ventricles and atria.

For each window there are one or more programs to create and maintain that particular window. We tried to make programs corresponding to different windows as independent as possible, but some interrelations were inevitable, for example to keep track of a 'current channel,' rejected channels, and marker values in different windows. Most relations, however, are hierarchical; for example, a button in one window can call a function that creates another window.

5. Status report

Examples of MAPLAB's graphical output are shown in Fig. 1. The software has been used for graphics creation for several years [5,24,26,27]. It has become the primary tool for several research

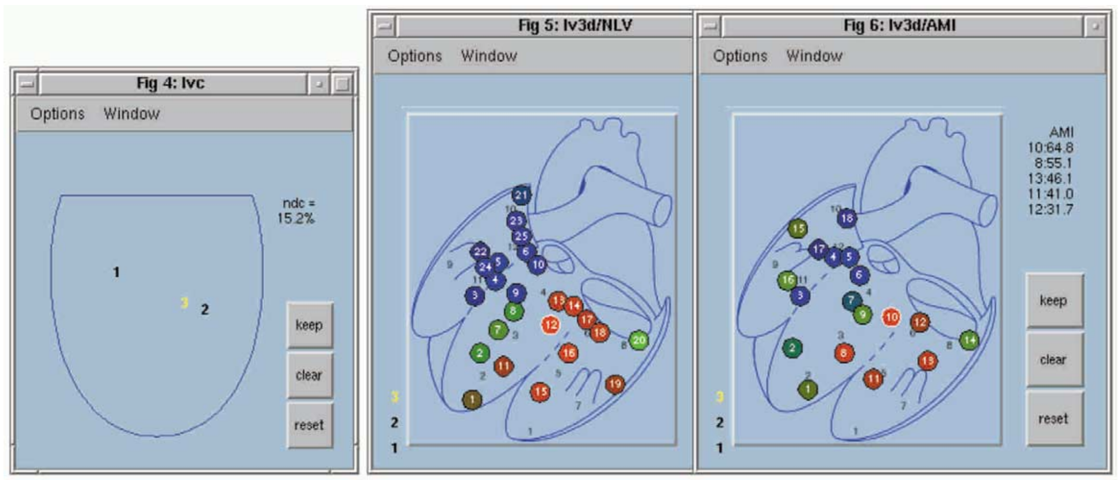


interval: 1583--1842 +200/-243 AMI

6: 0.84
7: 0.72
5: 0.52
3: 0.33

Basket.ecg R 1 rec. at
window = −9.5:120 zero = 800.5, limits = 6e+01ms 0.1V/s ,
lint = 5, cdata = activation time, text = milliseconds
09–Jun–2000 11:50:09

Fig. 5



ndc = 15.2%

AMI
10:64.8
8:55.1
13:46.1
11:41.0
12:31.7

Fig. 6

Figs. 5. and 6.

groups, and has played a part in many clinical and experimental studies [3,4,28–35]. The MAPLIB toolbox has been used in all ECG-related studies that were performed by its authors, where automated processing was appropriate [6,23,36,37].

6. Lessons learned

Software for multichannel ECG analysis should be fast, flexible, extensible, easy to program, and easy to use. These requirements are partly at cross purposes. The software we created is flexible, but not as fast as would be possible if it were implemented entirely in a system programming language such as C. This is a trade-off between programming efficiency and usage efficiency. User responses are fast enough if a modern computer is used. For example, on a Pentium-II PC at 233 MHz, drawing an annotated isochrone and pseudocolour map (as in Fig. 5) with 121 electrodes takes approximately 0.3 s, activation detection on 121 channels and 130 samples takes about 1 s, and dragging of markers happens with no noticeable flicker or delay.

One requirement for the package was that it would be extensible. Several extensions have been written and subsequently incorporated in the package. One such extension constitutes the interactive display of a 3-D multielectrode for the left atrium of a dog heart [19] (panel c in Fig. 4). The left atrium is a complex 3-D structure that cannot be mapped on a flat surface without losing important geometrical information. To overcome this problem, the 3-D shape of the 197-terminal multielectrode was obtained from computed tomography (CT) scans and stored as a triangulated surface, which was visualised on the computer. Measured potentials and activation times were displayed on this surface, and electrode positions were indicated. The object could be rotated with the mouse to allow inspection of all sides. This software was written in the same IPE as MAPLAB, so that it could easily interface with the latter. For example, analysis of ECG leads was performed with MAPLAB and shown directly on the 3-D surface. Electrode markers on the surface could be clicked with the mouse to show the corresponding ECG tracing in MAPLAB's Channel window (Fig. 5). The special window thus operated in the same way as MAPLAB's own Activation Map window.

On several occasions, we implemented new methods first as small MATLAB scripts, then improved them, and finally re-implemented them in C for better performance.

A minor drawback of an IPE is that—in principle—it allows the user to corrupt internal data. Therefore, we used variable names that are unlikely to be used by accident, and we experienced no problems in this respect.

As it stands, our software is less suitable for real-time clinical applications, such as BSM-guided catheter ablation [5]. Although it can serve

Fig. 5. Channel window (right), BSM window (top left), and Activation Map window (bottom left). The activation map is shown with isochrones, pseudocolours, and texts. The white lines in the activation map represent lines of block, as judged by the activation time difference between adjacent electrodes. White areas represent electrodes where no activation was detected within the current complex. The BSM window shows the body surface QRS integral map corresponding to this activation map. The format of this map is quite different from that shown in Fig. 1; the pseudocolour map used here can be generated faster and is useful for display on a computer monitor. A contour plot as in Fig. 1 is more suitable for hardcopy. The vertical lines between the ECG traces in the Channel window, and the dots in the lowest trace, are called 'markers', and are used to indicate time instants. Dot-shaped markers indicate instants that are applicable to a single lead, such as activation times, while line-shaped markers are used for instants that apply to all leads, such as the onset and offset of the QRS complex. The markers can be placed automatically by the software, e.g. by the activation detection routine, as well as by the user, by dragging them with a pointer device.

Fig. 6. Examples of Localiser windows. Several of these windows can be active in a single MAPLAB session, while each of them displays localisation results in a different format. The middle and right windows show database localisation results [21], indicated in a left-ventricular pseudo-3-D plot, using different databases. The title of the right window indicates that this is a (psuedo) 3-D left ventricular diagram featuring the anterior myocardial infarction (AMI) database. At the right side of this window, the five best-correlating segment numbers are indicated with the correlation values in percents. The left window shows continuous localisation results [6], given in a schematic diagram of the left ventricle. Numbers (in this case 1–3) are used to indicate localised positions; these numbers can be clicked with the mouse to reload the corresponding ECG data.

as a prototype in the hands of an experienced operator during clinical evaluation of a new procedure, it is too complicated and perhaps not sufficiently robust for routine clinical use. For such purposes, dedicated software and hardware, such as previously created for BSM procedures [38], remains a better choice.

The software is used by several medical and biomedical researchers, in both clinical and fundamental research groups [3–5,24,26–35,37]. The authors use the software extensively for their own experimental work. This experience, and the frequent contact with other users, have accelerated the development of the system.

Platform-independence turned out to be important. We programmed most comfortably on a LINUX system. LINUX, a UNIX implementation for the IBM-compatible PC, is a robust operating system for which all programmer's tools, such as a powerful text editor, text processing tools, C compiler, TEX system, make, and RCS are readily available, stable, free, and work seamlessly together. The system's robustness is even more important for programmers than for other users: segmentation faults are common during development, and a UNIX system handles them gracefully without consequences for other processes or the OS itself, even after hundreds of segmentation faults have occurred. On the other hand, two or three segmentation faults of our software or problems in other applications sufficed to crash any variant of the MS-windows system. However, most users prefer systems such as MS-windows or the Apple Macintosh. Therefore, we wanted to generate code for these platforms, while developing on a UNIX system.

We found that literate programming makes our programs more readable for ourselves and for others. It allows the authors of MAPLAB to understand, comment, and change each other's programs. Our experience confirms the observation that literate programming encourages the programmer to scrutinise the code [8]; many a misconception was brought to light by trying to explain the code.

Source control using the RCS system [15] and software building using the make program (which is standard software on UNIX systems) allowed different programmers to work on the same set of source files without the risk of overwriting each other's changes and to create consistent distributions of the package at any time.

The MAPLAB software is available free of charge for research purposes. Our work has several other aspects that may be of interest to other researchers. Our approach to interaction between a user interface and batch-job programming, as well as the combination of documented and revision-controlled software with a commercial data-processing package, may be of general interest.

This work may be helpful to researchers who have to choose new data analysis software, either in ECG analysis or other areas. Options range from home-grown programs in a low-level programming language, via higher level or visual programming languages, IPEs, to off-the-shelf data analysis packages. The latter are not suitable when experimentation with new algorithms is needed. Visual programming languages have the disadvantage that they cannot be combined with general tools for literate programming and source control, which require plain text program files. We chose an IPE because it allows a combination of high-level and low-level programming and interactive development, and—in our case—is largely platform-independent. Because our IPE uses plain-text program files, we could apply literate programming and source control. We thus obtained a compromise between program speed, programming speed, ease of use, flexibility, extensibility, and readability that serves us well.

7. Future plans

The software described here facilitates the implementation of data processing methods for multichannel ECGs and allows biomedical researchers to use them interactively. Researchers with some programming skills have written their own extensions and can also use the software for batch processing. Availability of a graphical user interface during batch-job programming was found to be very useful. These properties make the software suitable for continuing development. We expect that a growing user community will create custom extensions, and that we will be involved in several projects where extensions of the software are desired that are more difficult to program. In addition, we shall continue to improve the software, particularly in those places where weaknesses exist or develop. In our experience, addition of features to software often implies the addition of bugs, as well as new manifestations of existing bugs in other parts of the software. Handling these will be a continuing effort while MAPLAB is further developed.

Acknowledgements

We thank all users of our software for their encouragement. In particular we would like to thank Francien J.G. Wilms-Schopman (University of Amsterdam), Pascal F.H.M. van Dessel (Antonius Hospital, Nieuwegein, The Netherlands), and Michael Mlynash (University of California at San Francisco) for their efforts in debugging the software, and Sara J.A. Tasseron (University of Amsterdam) for her thorough review of this manuscript. This work was supported in part by the Dutch Technology Foundation STW under grant AGN 66 4098.

References

[1] P.V. Bayly, B.H. KenKnight, J.M. Rogers, R.E. Hillsky, R.E. Ideker, W.M. Smith, Estimation of conduction velocity vector fields from epicardial mapping data, IEEE Trans. Biomed. Eng. 45 (1998) 563–571.
[2] A.C. MettingVanRijn, A.P. Kuiper, A.C. Linnenbank, C.A. Grimbergen, Patient isolation in multichannel bioelectric recordings by digital transmission through a single optical fiber, IEEE Trans. Biomed. Eng. 40 (1993) 302–308.
[3] P.F. van Dessel, N.M. van Hemel, J.M. de Bakker, M. Potse, A.C. Linnenbank, E.R. Jessurun, J.A. Defauw, Comparison of simultaneous body surface mapping and 64-electrode endocardial mapping of postinfarction ventricular tachycardia exit, Circulation 100 (Suppl. I) (1999) 655–656 abstract.
[4] P.F.H.M. van Dessel, N.M. van Hemel, J.M.T. de Bakker, A.C. Linnenbank, M. Potse, E.R. Jessurun, A. Sippens-Groenewegen, E.F.D. Wever, Relation between body surface mapping and endocardial spread of ventricular activation in postinfarction heart, J. Cardiovasc. Electrophysiol. 12 (2001) 1232–1241.
[5] H.A.P. Peeters, A. SippensGroenewegen, E.F.D. Wever, H. Ramanna, A.C. Linnenbank, M. Potse, C.A. Grimbergen, N.M. van Hemel, R.N.W. Hauer, E.O. Robles de Medina, Clinical application of an integrated 3-phase mapping technique for localization of the site of origin of idiopathic ventricular tachycardia, Circulation 99 (1999) 1300–1311.
[6] M. Potse, A.C. Linnenbank, H.A.P. Peeters, A. Sippens-Groenewegen, C.A. Grimbergen, Continuous localization of cardiac activation sites using a database of multichannel EGG recordings, IEEE Trans. Biomed. Eng. 47 (2000) 682–689.
[7] D.E. Knuth, Literate programming, Comput. J. 27 (1984) 97–111.
[8] D.E. Knuth, Literate Programming, CSLI Lecture Notes No. 27, CSLI, Stanford University, 1992.
[9] I.D. Faux, M.J. Pratt, Computational Geometry for Design and Manufacture, Wiley, New York, 1979.
[10] D.E. Knuth, TEX: The Program, Series Computers and Typesetting, vol. 2, Addison-Wesley, 1986.
[11] D.E. Knuth, S. Levy, The CWEB System of Structured Documentation, Addison-Wesley, 1993.
[12] M. Potse, MWEB (MATLAB web) version 2.10, Free software. Available from the CTAN archives [13] in the directory web/matlabweb, 2000.
[13] The Comprehensive TEX Archive Network (CTAN). FTP sites: ftp://ctan.tug.org/tex-archive, ftp://ftp.dante.de/tex-archive, ftp://ftp.tex.ac.uk/tex-archive.
[14] M. Potse, The Webfiles package, version 0.5.39, Free software. Available from the CTAN archives [13] in the directory web/webfiles, 1999.
[15] D. Bolinger, T. Bronson, Applying RCS and SCCS, O' Reilly, 1995.
[16] L. Wall, T. Christiansen, R.L. Schwarz, Programming PERL, O' Reilly, 1996.
[17] A.C. Linnenbank, J.G.C. Kemmelings, S.L.C. Muilwijk, A. Peper, C.A. Grimbergen, An effective compression scheme for multilead ECG signals containing ectopic ventricular activation, in: Proceedings of the 14th Annual International Conference IEEE EMBS, 1992, pp. 527–528.
[18] R. Hoekema, G.J.H. Uijen, D. Stilli, A. van Oosterom, Lead system transformation of body surface map data, J. Electrocardiol. 31 (1998) 71–82.
[19] A.C. Linnenbank, J.G. Snel, M. Hocini, M. Potse, J.M.T. de Bakker, C.A. Grimbergen, Display of activation fronts on a 3D reconstruction of the left atrium of a dog, in: Proceedings of the 20th Annual International Conference IEEE EMBS, 1998, pp. I-90–I-93.
[20] J.G.C. Kemmelings, A.C. Linnenbank, S.L.C. Muilwijk, A. SippensGroenewegen, A. Peper, C.A. Grimbergen, Automatic QRS onset and offset detection for body surface QRS integral mapping of ventricular tachycardia, IEEE Trans. Biomed. Eng. 41 (1994) 830–836.
[21] A. SippensGroenewegen, H. Spekhorst, N.M. van Hemel, J.H. Kingma, R.N.W. Hauer, M.J. Janse, A.J. Dunning, Body surface mapping of ectopic left and right ventricular activation: QRS spectrum in patients without structural heart disease, Circulation 82 (1990) 879–896.

[22] J.A. Abildskov, L.S. Green, R.L. Lux, Detection of disparate ventricular repolarization by means of the body surface electrocardiogram, in: Cardiac Electrophysiology and Arrhythmias, Grune & Stratton, 1985, pp. 495–499.

[23] M. Potse, A.C. Linnenbank, H.A.P. Peeters, C.A. Grimbergen, Nondipolar content of body surface QRS and QRST integral maps, in: World Congress on Medical Physics and Biomedical Engineering, 1997.

[24] A. SippensGroenewegen, H.A.P. Peeters, E.R. Jessurun, A.C. Linnenbank, E.O. Robles de Medina, M.D. Lesh, N.M. van Hemel, Body surface mapping during pacing at multiple sites in the human atrium: P-wave morphology of ectopic right atrial activation, Circulation 97 (1998) 369–380.

[25] A. SippensGroenewegen, M.D. Mlynash, P.G. Guerra, F.X. Roithinger, B. Tilg, M.D. Lesh, Atlas of paced body surface P wave integral maps for localization of focal left atrial arrhythmias, PACE 23 (Part II) (2000) 556 abstract.

[26] J.-H.E. Dambrink, A. SippensGroenewegen, W.H. van Gilst, K.H. Peels, C.A. Grimbergen, J.H. Kingma, Association of left ventricular remodeling and nonuniform electrical recovery expressed by nondipolar QRST integral map patterns in survivors of a first anterior myocardial infarction, Circulation 92 (1995) 300–310.

[27] H.A.P. Peeters, A. SippensGroenewegen, E.F.D. Wever, M. Potse, M.C.G. Daniëls, C.A. Grimbergen, R.N.W. Hauer, E.O. Robles de Medina, Electrocardiographic identification of abnormal ventricular depolarization and repolarization in patients with idiopathic ventricular fibrillation, J. Am. Coll. Cardiol. 31 (1998) 1406–1413.

[28] J.M.T. de Bakker, R. Derksen, T. Kawara, S. Tasseron, A.C. Linnenbank, M. Potse, M.J. Janse, Load mismatch as a cause of decremental conduction in diseased myocardium, Circulation 100 (Suppl. I) (1999) 836 abstract.

[29] M. Hocini, J.M.T. de Bakker, A.C. Linnenbank, Y. Ho, M.J. Janse, M. Haissaguerre, High resolution mapping of the left atrium and the pulmonary veins during pacing and atrial fibrillation in the isolated canine heart, Circulation 99 (Suppl. I) (1998) 208–209 abstract.

[30] P. Loh, J.M.T. de Bakker, M. Borggrefe, M.J. Janse, High resolution mapping of re-entrant activation in the AV node during ventricular echoes, Circulation 100 (Suppl. I) (1999) 836 abstract.

[31] E.R. Jessurun, J.M.T. de Bakker, N.M. van Hemel, A.C. Linnenbank, J.J.A.M. Defauw, A. Brutel de la Riviere, Surgical modification of the right atrium for the abolishment of atrial fibrillation with maze III does not affect fibrillation intervals, Circulation 100 (Suppl. I) (1999) 67 abstract.

[32] H.V.M. van Rijen, T.A.B. van Veen, M.J.A. van Kempen, F.J.G. Wilms-Schopman, M. Potse, O. Krueger, K. Willecke, H.J. Jongsma, J.M.T. de Bakker, Impaired conduction in the bundle branches of mouse hearts lacking the gap junction protein connexin40, Circulation 103 (2001) 1591–1598.

[33] J.R. de Groot, R. Coronel, F.J.G. Wilms-Schopman, C.A. Remme, M.J. Janse, Critical cellular coupling required for 1B ventricular fibrillation in the isolated regionally ischemic porcine heart, Circulation 100 (Suppl. I) (1999) 839 abstract.

[34] J.R. de Groot, F.J.G. Wilms-Schopman, T. Opthof, C.A. Remme, R. Coronel, Late ventricular arrhythmias during acute regional ischemia in the isolated blood perfused pig heart. Role of electrical cellular coupling, Cardiovasc. Res. 50 (2001) 362–372.

[35] T. Kawara, R. Derksen, J.R. de Groot, R. Coronel, S. Tasseron, A.C. Linnenbank, R.N.W. Hauer, H. Kirkels, M.J. Janse, J.M.T. de Bakker, Activation delay after premature stimulation in chronically diseased human myocardium relates to the architecture of interstitial fibrosis. Circulation 104 (2001) 3069–3075.

[36] M. Potse, A.C. Linnenbank, P.F.H.M. van Dessel, N.M. van Hemel, C.A. Grimbergen, The use of simultaneous multichannel endocardial and surface electrocardiograms for verification of exit site localization using body surface mapping, in: Proceedings of the 20th Annual International Conference IEEE EMBS, 1998, pp. I-75–I-78.

[37] J.R. Willemsen, A.C. Linnenbank, P.F.H.M. van Dessel, M. Potse, C.A. Grimbergen, N.M. van Hemel, A new method for detecting late potentials using multichannel ECGs, in: Proceedings of the First Joint BMES/EMBS Conference, 1999.

[38] A.C. Linnenbank, On-site recording, analysis, and presentation of multichannel ECG data, Ph.D. thesis, University of Amsterdam, The Netherlands, August 1996.