Available online at www.sciencedirect.com
SCIENCE DIRECT®
Chemometrics and Intelligent Laboratory Systems 71 (2004) 107–120
Chemometrics and intelligent laboratory systems
www.elsevier.com/locate/chemolab

Review

Information technologies for comprehensive two-dimensional gas chromatography

Stephen E. Reichenbach*, Mingtian Ni, Visweswara Kottapalli, Arvind Visvanathan

Computer Science and Engineering Department, University of Nebraska-Lincoln, Lincoln, NE 68588-0115, USA

Received 5 November 2003; accepted 5 December 2003

Available online 8 March 2004

Abstract

Comprehensive two-dimensional gas chromatography (GC × GC) presents information technology challenges in data handling, visualization, processing, analysis and reporting. With its significantly greater separation capacity, GC × GC generates data sets that are two to three orders-of-magnitude larger than for conventional GC and that exhibit an order-of-magnitude or more distinct peaks in complex multidimensional patterns. GC × GC is a much richer source of data for chemical analyses, but extracting relevant information from the large, complex data sets requires advanced information technologies. This paper reviews a range of state-of-the-art information technologies for GC × GC, including graphical user-interfaces (GUIs), computer graphics for data visualization, data file formats for storage and interchange, digital image processing, image analysis and pattern recognition, data and information formats, and software architecture and engineering.

PACS: 82.80.Bg; 07.05.−t; 07.05.Bx; 07.05.Kf; 07.05.Pj; 07.05.Rm; 07.05.Wr
Keywords: Comprehensive two-dimensional gas chromatography; GC×GC; Information technology; Image processing; Visualization; Computer interfaces

1. Introduction

Comprehensive two-dimensional gas chromatography (GC × GC) is an emerging technology for chemical separation that provides an order-of-magnitude increase in separation capacity over traditional gas chromatography and is capable of resolving several thousands of chemical compounds [1]. Fig. 1 illustrates a portion of GC × GC separation of chemical warfare agents in a hydrocarbon matrix, displayed as a two-dimensional image.

The quantity and complexity of GC × GC data makes human analyses of GC × GC images difficult and time-consuming and motivates the need for computer assisted and automated processing. GC × GC transforms chemical samples into raw data; information technologies are required to transform GC × GC data into usable information.

The lack of software for GC × GC data and information processing is a significant impediment to the adoption of GC × GC for routine applications. Presently, GC × GC is limited to research laboratories in large part by the difficulty of data interpretation. A recent article on GC × GC data interpretation notes that most GC × GC research groups use "their own custom-written software" (Ref. [3], p. 881). GC × GC will not be used more widely until there are generally available information technologies that meet requirements for routine applications.

This paper provides an overview of the challenges in extracting information from GC × GC data, including:

- formatting data for storage, access and interchange;
- visualizing multidimensional data;
- processing data to remove acquisition artifacts;
- analyzing data to quantify and identify chemical compounds;
- managing data and information repositories; and
- reporting.

Advanced information technologies offer powerful solutions for these problems, including:

- graphical user-interfaces (GUIs),
- computer graphics,
- digital image processing,

* Corresponding author. Tel.: +1-402-472-5007; fax: +1-402-472-7767.
E-mail address: reich@cse.unl.edu (S.E. Reichenbach).
URL: http://cse.unl.edu/~reich.

0169-7439/$ - see front matter © 2004 Published by Elsevier B.V.
doi:10.1016/j.chemolab.2003.12.009

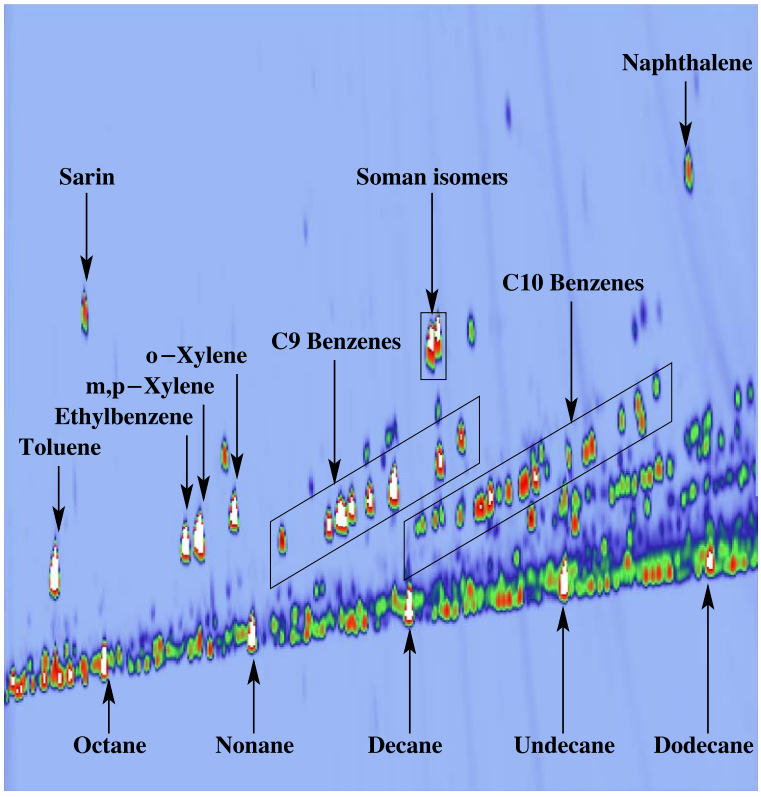


Fig. 1. A GC × GC separation of chemical warfare agents in a hydrocarbon matrix displayed as a two-dimensional image with colorization and annotations [2].

- pattern recognition,
- data formatting and database technologies, and
- software architectures and engineering.

Fig. 2 presents a conceptual overview of a GC × GC software system. The software interacts with the user via GUIs that present visualizations to the user and take commands from the user to form an execution script. Scripts also can be generated without a GUI (e.g., with a text editor) and used to specify a sequence of actions that do not require interaction. A script can be recorded as a journal of actions performed—an important element of good laboratory practice (GLP) [4–6].

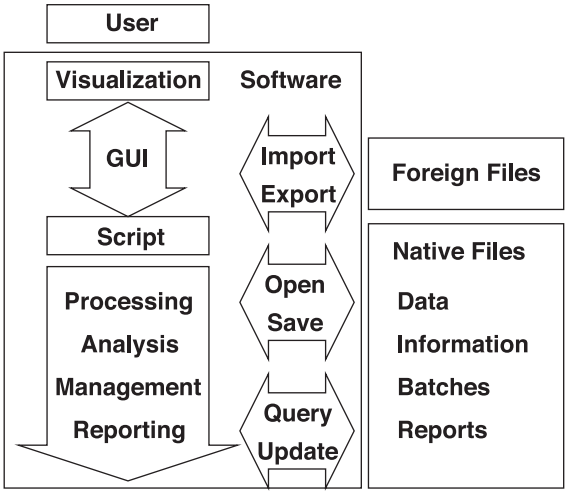


Fig. 2. Software system overview.

The typical data flow is a sequence of: processing data to correct artifacts, analyzing data to produce higher-level information, managing data and higher level information (e.g., the collection of multiple GC × GC runs into batches and projects) and generating reports.

The software imports and exports data and other information from and to foreign file formats, reads and writes data and other information from and to files in its native formats, and queries and updates its repository of data and other information.

The following sections examine various information technologies for GC × GC. Sections 2 and 3 present a quick overview of GC × GC data processing and introduce GC Image, a software system developed at the University of Nebraska-Lincoln for GC × GC visualization, processing, analysis and reporting. GC Image is used in subsequent sections to illustrate the application of information technologies to GC × GC. Sections 4, 5 and 6 consider information technologies for GUIs, visualization and scripts. Sections 7 and 8 examine GC × GC processing and analysis. Sections 9, 10 and 11 consider formatting, managing and reporting on GC × GC data and information. Section 12 draws conclusions about the state-of-the-art and the future of information technologies for GC × GC.

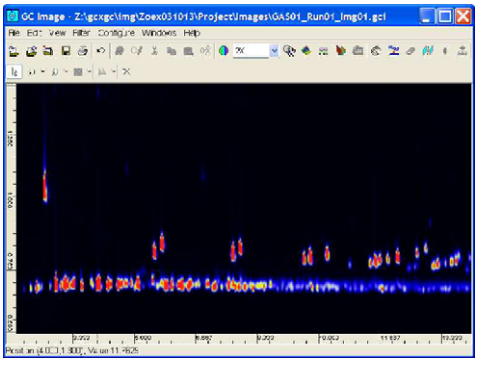
2. GC×GC data processing

This section presents a quick overview of processing GC × GC data sets to produce a summary analytical report. Depending on the data itself and the type of report, various operations may be required, but the following sequence is typical:

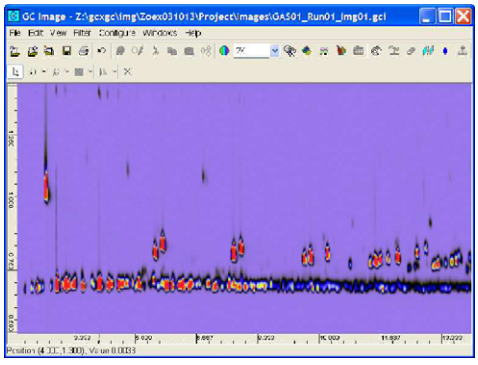
(1) background removal
(2) blob detection
(3) template matching
(4) batch creation
(5) report generation

Fig. 3A illustrates a raw GC × GC chromatogram colorized and presented as an image with GC Image (a software system described in Section 3). The abscissa is the retention time in the first column and the ordinate is the retention time in the second column. The status bar at the bottom of the Image Viewer indicates the location of the cursor and the pixel (or data) value at that location.

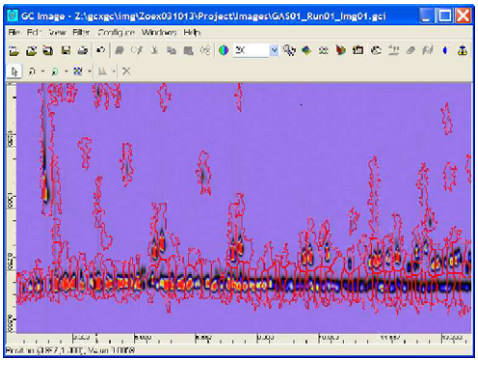
The background level varies slowly with time, for example, in this chromatogram, from slightly less than 12 pA at the beginning, down to just over 11.5 pA in the middle and just over 12 pA at the end. Typically, the first step in processing GC × GC data is to remove this slowly varying background [7]. Then, as in Fig. 3B, the peaks in



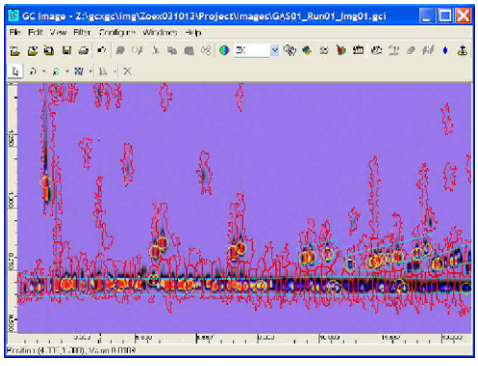
A. Raw GCxGC chromatogram.



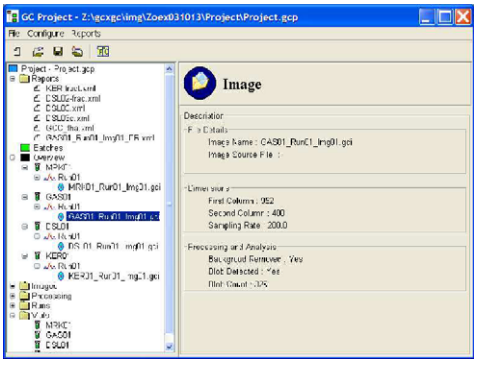
B. Background removal.



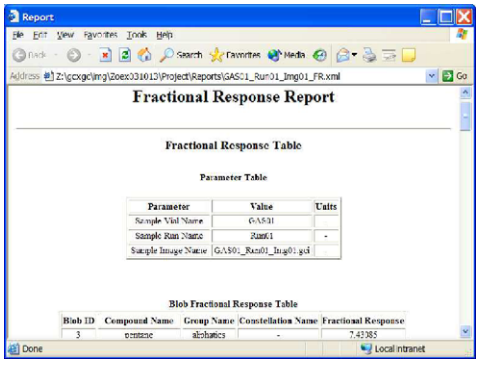
C. Blobs detection.



D. Template matching.



E. Batch creation.



F. Report display.

Fig. 3. Typical GC × GC data processing sequence.

the chromatogram rise above a near-zero baseline and can be quantified more accurately.

The next step is blob detection—the process of aggregating clusters of pixels that form distinct peaks, as illustrated in Fig. 3C. Then, the chemical identities of the peaks (including metadata such as compound names, compound classes and internal standard associations) are assigned, for example using a previously generated template for pattern matching as illustrated in Fig. 3D.

Analyses may involve multiple chromatograms (e.g., one or more calibration runs and one or more sample runs, organized as illustrated in Fig. 3E). Finally, Fig. 3F illustrates a summary report—in this case, a fractional response report.

This quick introduction only brushes across the surface of the problems of GC × GC data processing and the information technology challenges these problems present. Subsequent sections examine these issues in more depth.

3. GC Image

GC Image is a software system developed at the University of Nebraska-Lincoln, for GC × GC visualization, processing, analysis and reporting. Advanced information technologies provide high-performance, speed development of powerful interactive tools, and provide a solid foundation for maintenance, inter-operability and continued development.

GC Image is built on an object-oriented (OO) software architecture, is written in the Java programming language [8,9], and makes extensive use of resource files. The OO software architecture and implementation provide a high degree of modularity and abstraction for enhanced maintainability and extensibility. Java has many built-in objects that can be reused or extended. GC Image uses both built-in objects and additional objects specifically designed for GC × GC. Independent resource files further enhance maintainability and extensibility, e.g., internationalization with multiple languages.

GC Image formats data files, reports, scripts and journals using the eXtensible Markup Language (XML) [10]. XML structures data to provide enhanced extensibility, interchange and access via query, search and data mining. Programming tools, such as the Simple Application Programming Interface (API) for XML (SAX) [11], speed development of XML applications. The XML Style Language (XSL) and XSL Transformations (XSLT) [12] provide distributed access to data and reports via ubiquitous web browsers. GC Image has specially designed XML Schema [13] for GC × GC data, reports, scripts and journals and XSLT [12] specifications for presentation. GC Image also supports import from and export to foreign formats such as comma-separated values (CSV) text, binary and digital image formats (e.g., JPEG [14]).

GC Image has powerful GUIs and a suite of visualization tools customized for GC × GC. GUIs simplify complex user interactions with GC × GC data and methods. Computer graphics provide interactive visualization of complex GC × GC data with a variety of views. Programmer toolkits, such as Java2D [15], Java Swing [16], Java3D [17] and OpenGL [18], speed custom GUI and computer graphics development. GC Image uses a novel approach for GUI architecture and control, using hierarchical state machines (HSM) with state-based accessibility control (SAC) to simplify interfaces and interface programming [19].

GC Image employs advanced algorithms specially developed for processing and analyzing GC × GC data. These algorithms are implemented using an OO approach that makes use of the Java Advanced Imaging (JAI) [20] toolkit. Calibration and correction techniques can remove acquisition artifacts for improved analysis. Image segmentation algorithms demarcate peaks and background. Pattern recognition algorithms and library search tools identify and label individual chemicals and groups in complex matrices separated by GC × GC.

Information management and report generation tools support manipulation and output of higher-level information. Project management tools organize data and analyses of individual runs into a hierarchical grouping of related runs. Reporting tools deliver both standard reports such as ASTM D5580 [21] as well as more generic types of reports such as internal standard and fractional response reports.

4. Graphical user-interfaces

GUIs provide sophisticated output to the computer screen and allow richer input specifications from the user. GUIs are central to the usability of interactive applications. GUIs should be easy to learn, easy to use and highly configurable.

The GC Image GUIs include a main frame for general operations and numerous popup dialog windows for specialized operations. The resizable main frame contains menus, toolbars, a status bar and a work area. All major functions of the software are accessible through the menus. Frequently used functions are accessible through buttons on the toolbars. Other tasks such as editing and navigation are accessible through mouse clicks, key strokes or their combinations. Fig. 4 shows the GC Image main interface and one of the popups, the Image Navigator.

The Image Navigator addresses the problem that many GC × GC images are too large to be displayed at native resolution on typical computer screens. The thumbnail of the entire image is shown in the Image Navigator (a resizable popup dialog window) with a small rectangular region indicating the part of image that is displayed in the Image Viewer in the main frame. The region-of-interest (ROI) box in the Image Navigator can be relocated or resized via click-and-drag with the mouse. The Image

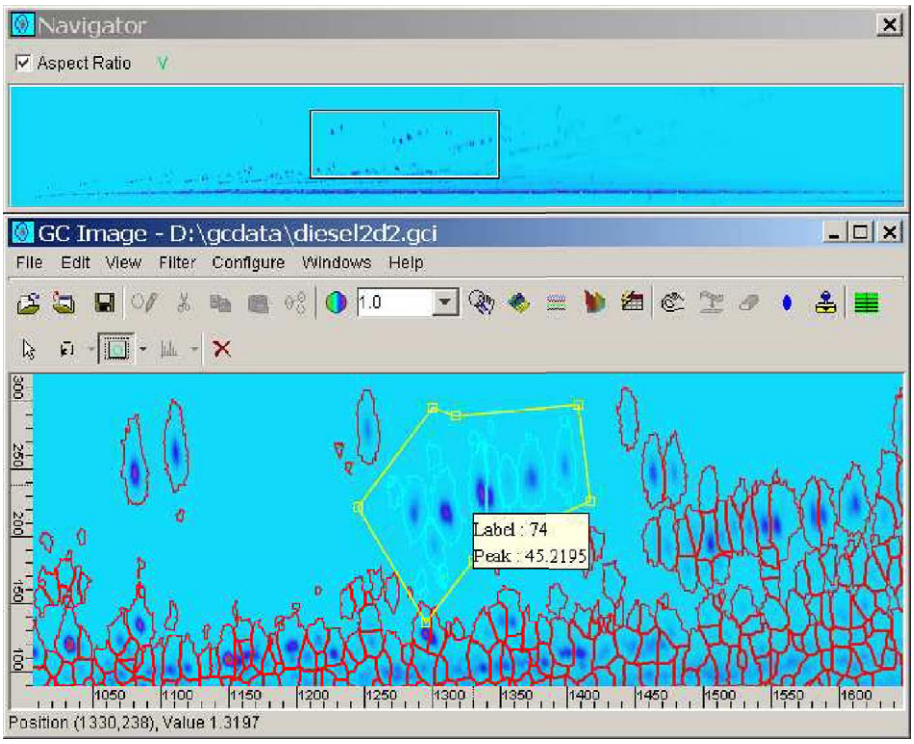


Fig. 4. The GC Image main interface with Image Viewer (bottom) and the Image Navigator (top).

Viewer also supports click-and-drag for pan and scroll and has pull-down menus and text boxes for resizing and rescaling. These schemes illustrate how GUIs can provide sophisticated, yet intuitive, interfaces that are easy to learn and use.

Effective GUIs can make software easy to learn and use, but GUIs are challenging to design, implement, modularize, debug and modify [22]. GC Image uses several techniques to develop modular, easily maintained GUIs.

- GC Image uses Java technologies as the basis for the implementation. Java provides: (1) packages of high-level GUI components such as the Abstract Window Toolkit (AWT) and Swing [16]; (2) a well-designed event handling mechanism—the delegation-based event model; (3) packages of advanced graphics and image processing algorithms; and (4) good support for multithreaded programming.
- Each GUI component is associated with resources (such as button labels and mouse-over tooltips) in a resource file. The resource file maintains most resources needed to define a component. This mechanism not only simplifies the GUI programming, but also makes modification and maintenance easier. Resources are important for modular internationalization of GUIs.
- GC Image integrates a HSM [23] and a novel technique, SAC, to model and control the entire interface [19]. This innovative approach helps modularize the GUI, simplify the GUI programming and reduce users' operational errors.

SAC makes accessible only the components that are semantically applicable in the current state. A typical GUI consists of many components such as windows, buttons and menus [24]. Each component is associated with a certain function(s). A GUI component can be either accessible or inaccessible. A component is accessible if it is visible and enabled. An inaccessible component is either hidden or disabled. A GUI component (and its associated function) may not be applicable all of the time. Its applicability depends on the current state of the GUI. Given the fact that users can be easily overwhelmed by the numerous options of a modern GUI, it is desirable to have state-sensitive control over GUI components' accessibilities. With SAC, because non-applicable GUI components are not accessible to users, users' operational error rates are reduced. For GUI programmers, SAC guarantees the applicability of the accessed component. Complex logic for validating the execution context of the corresponding function becomes unnecessary. Consequently, GUI implementation is simplified. The experience with GC Image shows that integrating a HSM with SAC provides a powerful framework for modeling and controlling GUIs.

5. Visualization

Visualization is frequently critical to GC × GC processing and analysis. Visualization technologies convert GC × GC data into visual form, often allowing intuitive interpretation of what otherwise cannot be discerned. By providing prompt visual feedback, software can efficiently guide users to solutions.

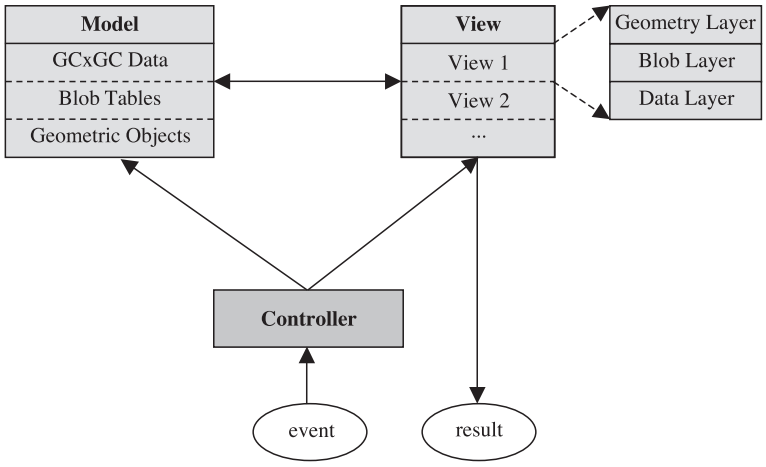


Fig. 5. The multiview multilayer architecture.

The visualization module of GC Image is based on a multiview multilayer (MVML) architecture, with a variety of supporting toolkits and rendering techniques. This approach supports rich visualizations in multiple views under user control.

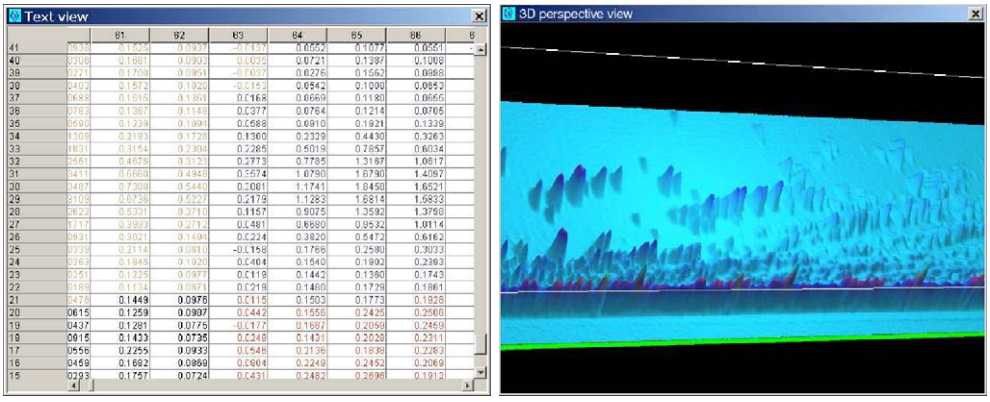
5.1. The multiview multilayer architecture

The kernel of the MVML architecture is the model-view-controller (MVC) architecture [16,25], which is a software design pattern for the creation of interactive applications. In the MVML architecture, the GC × GC visualization module is divided into three components as depicted in Fig. 5. The model component is the center of the program. It maintains all the data of the system, including GC × GC data, the blob peak tables generated from the data and the geometric objects that users create. The controller component handles the interaction between users and the program. The controller accepts events from users and dispatches them to appropriate receivers. The view component shows results to users.

Based on the MVML architecture, the GC × GC visualization module is able to visualize the same piece of data in several different ways, including two-dimensional image view, three-dimensional perspective view, one-dimensional slice/projection view, text view and statistics view. In addition, different types of data in the model are displayed in different layers in each view, and each layer can be controlled and accessed separately. In GC Image, three layers currently are distinguished. From bottom to top, they are data layer (the data or pixel values), blob layer (two-dimensional clusters of pixels constituting separate peaks) and graphics layer (user-defined geometric objects). Not all views may support all three layers. For example, two-dimensional image view supports all three layers, but the text view currently does not support the graphics layer. Two example visualizations of regions of a GC × GC image are shown in Fig. 6.

5.2. Hierarchy of supporting toolkits

Layering is an important design method for complex applications such as GC × GC visualization. The visualization module is built on top of other lower-level toolkits. A conceptual view of the hierarchy is shown in Fig. 7.



A) Text view. B) Three-dimensional perspective view.

Fig. 6. Visualizations of GC × GC data in the MVML architecture, (a) text view and (b) three-dimensional perspective view.

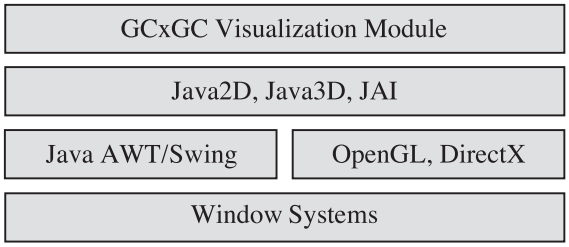


Fig. 7. Layered view of supporting toolkits.

Window systems are part of the underlying operating system. They act as a general interface between higher-level applications and input/output devices. Java AWT/Swing wraps around the APIs of the window systems in OO fashion, making them easier to use. Low-level graphics libraries such as OpenGL and DirectX are more hardware-oriented and focus more on performance. Acting as black-box systems, low-level graphics layers hide their architecture and provide users with a set of well-defined APIs. Java2D [15], Java3D [17] and JAI [20] are high-level graphics and image processing libraries. They are designed as collections of classes in the OO paradigm and pay more attention to usability and portability. While graphics and image processing libraries provide generic rendering and processing facilities, the visualization modules specifically focus on solving GC × GC visualization problems. They provide sophisticated data structures and a wide variety of visualization methods appropriate for GC × GC.

5.3. Rendering techniques

Different views use different rendering techniques. GC Image renders the different data layers with the following views.

5.3.1. Two-dimensional image view

The data layer can be visualized as two-dimensional images through color mapping. Color mapping is controlled by three piecewise linear or spline functions (for Red/Green/Blue or Hue/Saturation/Brightness) and many other parameters. Interpolation is used for display at non-native resolution. Supported interactions include panning and zooming. Blobs can be rendered as highlighted outlines or with other graphical highlights under user control). Different types of blob peaks (e.g., internal standards) can be encoded with configurable colors. The graphics layer consists of user-defined geometric objects such as polygons and polylines.

5.3.2. Three-dimensional perspective view

The data layer can be visualized as a three-dimensional wireframe or surface based on Gouraud shading [26] and the Java3D viewing model [17]. Multiresolution techniques accelerate rendering. Level-of-detail control [17], panning, zooming and rotating are supported in the three-coordinate system. The blob-peak layer can be rendered on top of the three-dimensional surface through texture mapping [17,26].

5.3.3. One-dimensional slice/projection view

Selected rows and columns of the data layer can be sliced or projected for one-dimensional display. Different rows/columns can be rendered in different colors and overlaid. Supported interactions are panning and zooming. Blob peaks are visualized by color coding the one-dimensional projections.

5.3.4. Text view

The values of the data layer can be shown directly as a table. Each table cell displays a value. Horizontal and vertical scrolling are supported. Blob peaks are visualized by color coding the tables cells so that adjacent blobs have different colors. Color selection is based on a vertex graph coloring algorithm [27] with an adjacency graph.

5.3.5. Statistics view

Many different statistics can be computed on selected regions of the input data. Then, a combination of one or two selected statistics can be plotted. For example, pixel value versus first-column retention time or blob volume versus blob area.

6. Scripts and journals

Users interact with GUIs to specify actions to be performed by the software, such as processing and analyzing operations. The sequence of actions can be regarded as a script. As an information technology, a script is a sequence of actions, much like a computer program, but without some formalisms such as data types. Scripts "typically interact either with other programs (often as glue) or with a set of functions provided by the interpreter" [28].

The script specified by the user through the GUIs is executed by functions in the software. As it is specified by the user and executed, the script can be recorded and saved to a file. The file subsequently can be used to drive execution of the software even without GUI interaction. This is a great convenience if the same sequence of actions is required on multiple images. As it is executed, the script can be augmented with additional information to create a journal, i.e., a record of actions and events. Additional information can include details of errors and other exceptions encountered during processing and details required to undo operations. The journal is a record that can be used as an element of GLP [4–6].

Journals can be kept either by session (i.e., actions performed on an open GC × GC image during the current session) or complete (i.e., all actions performed since image creation). Session journals are more efficient, discarding information when the open image is closed. Complete journals store information in the image file, with associated

storage overhead. A journal can be switched from complete to session (i.e., discarding journal entries), but not vice versa.

Journals can be kept as reversible or permanent. A reversible journal records artifacts that support undo operations. A permanent journal is script without artifacts for undoing, meaning operations once performed cannot be undone. A prohibition on undo operations may be desired as a part of laboratory practices.

In GC Image, scripts and journals are formatted using the XML. (See Section 9 for a discussion of XML for GC × GC.) Parsing of scripts takes full advantage of the extensibility of XML, with the interpreter passing all parameters to the execution module in a generic manner. This approach allows easy plug-and-play for new or updated operations without modification of the interpreter. GC Image does not currently support management or querying of scripts or journals, but the XML format provides the basis for future support.

7. Processing

As with any sensing instrument, GC × GC introduces artifacts during data acquisition. Digital signal processing, or for two-dimensional GC × GC data, digital image processing, can characterize artifacts and correct for them, thereby improving data quality. Corrective processing can be considered for both value, i.e., the response of the system to chemical inputs, and for retention time, i.e., the time of responses in the data stream. Signal processing of GC × GC data presents special problems and opportunities. Two examples of GC × GC signal processing, one for value and one for time, are outlined here.

Signal peaks, corresponding to chemical constituents in the sample, rise above a background level in the output. Even under controlled conditions, the background level consists primarily of the sum of two slowly varying components: a steady-state standing-current offset (characteristic of many GC detectors) and temperature-induced column-bleed. Accurate quantification of the chemical-related peaks requires characterization, i.e., determination of the time-variant background level, and correction, i.e., subtraction of the background level from the signal.

GC Image uses signal processing to characterize and remove the background level from GC × GC data [7]. The approach takes advantage of the deadband present in each GC × GC secondary chromatogram. Values from each deadband are used to statistically characterize system noise. Then, values within the noise range of the deadband values are used to statistically characterize the background level. Finally, the background level is subtracted from the image, producing a chromatogram in which the peaks rise above a near-zero mean background.

It is desirable to normalize GC × GC images such that the time zero of the second column separations is synchronized with the row zero of the image. However, depending on the temporal relationship between the start of sampling and the thermal modulation period, post-acquisition phase-shifting of the data may be required. Even if sampling and thermal modulation are synchronized, other variations can make shifting phase desirable.

As an example of the need for phase shifting, suppose that in one data set, sampling is initiated at the time of the start of the thermal modulation cycle (as in Fig. 8A below) and that in another data set, sampling is initiated at the time just before the start of the thermal modulation cycle (as in Fig. 8B below). Then, the peaks in the second image will be offset relative to the first image.

GC Image supports phase shifting of GC × GC data by up to plus or minus one-half of the thermal modulation period. This is sufficient for any phase misalignment between the thermal modulation cycle and the start of sampling. Then, the first and last columns, which may be padded in this process, are excluded from analysis.

Other corrections are possible depending on particular system characteristics.

8. Image analysis and pattern recognition

In a GC × GC image, each resolved chemical substance produces a small cluster of pixels with values that are larger than the background values. Well-formed blobs have a single peak. Ideally, there is little or no overlap (due to co-elution) of blobs related to different chemicals. (However, this ideal is

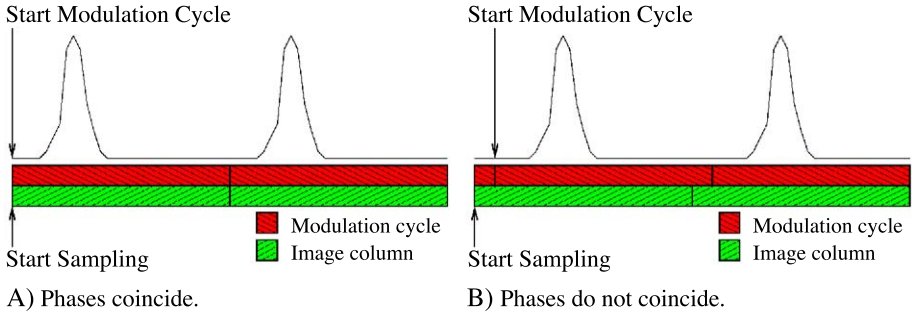


Fig. 8. Relative phases of modulation cycle and data sampling.

commonly not realized.) The objective of image analysis is to:

(1) distinguish the separate blobs in the GC × GC image—an operation termed segmentation;
(2) compute important statistical properties of each blob, e.g., the total of all pixel values in the blob; and
(3) identify important metadata about each blob, especially the name of the chemical producing the blob.

8.1. Blob detection and statistics

The *drain algorithm* for detecting blob peaks is an inversion of the *watershed algorithm* [29]. The algorithm can be understood conceptually by picturing the image as a relief map with larger values having higher elevation. The surface is placed under enough "water" to cover the point of highest elevation. Then, the water is "drained". As the draining proceeds, peaks appear as "islands" and are distinguished with a unique identification number. When the draining eliminates the "water" between "land masses", then the border between blob peaks is set. One of the problems associated with the watershed algorithm is over-segmentation-detection of multiple regions that should be segmented as a single region. This problem is caused by noise artifacts. Various approaches, such as smoothing, can be used to reduce the problem.

Once segmentation is complete, each separately identified blob consists of a group of pixels. Statistical features describe and characterize the blob. For example, the number of pixels is the area of the blob and the sum of pixels is the volume of the blob. The statistical moments of the blob can be used to measure characteristics such as symmetry, orientation, eccentricity, etc. GC Image has built-in support for more than 70 statistical blob features [30].

8.2. Chemical identification by template matching

GC × GC images contain potentially thousands of peaks in complex patterns, making chemical identification a challenging problem. Manually identifying chemicals is tedious and time-consuming. An alternative is to use template matching. Template matching involves two peak sets: a peak template and a target peak set. A peak template is a set of peaks with metadata that identify and characterize the peak (e.g., chemical name, chemical group, internal standard, etc.). A target peak set is a set of peaks whose metadata is to be determined. A target peak has only computed statistical features, such as peak location, area, volume, shape, etc. Determining the proper metadata values for the target peak set is the objective of the chemical identification process.

Given a peak template and a target peak set, peak template matching tries to establish as many correspondences as possible from peaks in the template to peaks in the target peak set. One fundamental difficulty of the matching process comes from image-to-image distortions, which cause the same chemicals appear at slightly different locations in different images. The distortions may be caused by differences in proportional integral derivative (PID) loop control of the temperature program, column deterioration over time and instrument-to-instrument variations in physical parameters such as carrier gas pressure and flow rate. Matching techniques seek to remove these distortions and find the correct correspondences.

Peak template matching for chemical identification consists of three steps:

- Construct a peak template. A peak template begins with the manual selection of peaks of interest and specification of metadata (e.g., chemical name). Templates also can be created by editing other templates. Ancillary information such as time-of-flight mass spectra also can be used to provide metadata.
- Match template and target peaks. Matching establishes peak correspondences from the template to a target peak set.
- Apply the template. After peak correspondences are established, the metadata carried by the peaks in the template is copied into the corresponding peaks in the target peak set. Consequently, all the matched chemicals in the target peak set are identified.

Templates in GC Image also may include graphical objects, such as polygons and polylines, that are used to ascribe metadata to the target image. Such graphics objects can be used to include target peaks for reporting (e.g., the C9+ aromatics for ASTM D5580) or to ignore peaks for reporting (e.g., blobs associated with solvent or with column bleed).

8.3. Point pattern matching

Identified peaks have both computed features and metadata. Typically, only computed features are used for matching. Among computed features, retention time (i.e., peak location or coordinates of the pixel with the largest value within the peak) encodes the most important information for a specific peak. Therefor, peak location is used as the primary feature for matching. Other computed features (e.g., volume) may be used for pruning the search space in matching algorithms. Using peak location, a peak set can be represented as a point set and the peak template-matching problem becomes a point pattern-matching problem.

Let $P=\{p_i(x_i,\ y_i)\}_{i=1}^{n}$ represent the peak template and $Q=\{q_i(u_i,\ v_i)\}_{i=1}^{n}$ represent the target peak set. The peak template-matching problem can be formalized as follows [31]:

Given template point set P, target point set Q, distance measure d, transformation space T and distance threshold ϵ, compute

$$\max_{t\in T, A\subset P, B\subset Q}\{|A|\ |\ d(t(A),B)\leq\epsilon\}.$$

Generally, P may not be congruent to Q or any subset of Q. The above definition is merely intended to maximize the number of points in P which can be matched to points in Q subject to the distance measure and the transformation space.

An alternative definition is to fix the number of points to be matched and minimize the distance:

Given template point set P and target point set Q, distance measure d, transformation space T and desired number of points to be matched l, compute

$$\min_{t\in T, A\subset P, B\subset Q}\{d(t(A),B)\,|\,|A|\geq l\}.$$

The solution to the problem is a transformation. From the transformation, the point correspondences from P to Q are then computed.

A wide variety of techniques have been proposed for solving point pattern matching problems, including Recognition by Alignment [32], Generalized Hough Transforms [33], Geometric Hashing (also called Pose Clustering) [34], Searching Correspondence Space [35], Minimizing Hausdorff Distance [36,37], Recognition by Adaptive Subdivision of Transformation Space (RAST) [38,39], etc. All these techniques work well only for relatively simple transformations (e.g., rigid transformations) and small point sets. For more complex transformations and large point sets, the high dimensionality of the transformation space and the large size of the point sets usually render those techniques ineffective.

GC × GC images may contain several thousand peaks. Experiments show that rigid transformations (combinations of translations, scales and rotations) are not flexible enough to remove image-to-image distortions. These two facts suggest that existing methods may not be adequate to solve the peak template-matching problem for GC × GC and that more efficient and effective techniques are required. This is an area of active research [40–42].

9. Formats for storage, access and interchange

Information technologies require formatted data and information for various uses, including storage, access and interchange. Storage in a long-term archive may be mandated [4]. Direct access to databases and information repositories may be required to support operations such as query, search and data mining. Increasingly, distributed access, e.g., via the world-wide web or intranets, requires interchange between different software systems.

Several formats are used for chromatographic and other analytical data (e.g., spectroscopic, electrophoretic, etc.), but no format has gained universal acceptance, at least in part due to proprietary controls and other problems with existing formats. Proprietary formats are designed for use with specific software, which limits their general use, especially for interchange. Other formats have been released into the public-domain so that they may be used more widely and as the basis for interchange, but in reviewing existing public-domain formats, James Duckworth [43] noted a variety of limitations, including, complexity, incompleteness, limited numerical precision, limited readability, difficult validation and lack of extensibility.

The added dimensionality of GC × GC data and the information extracted from it, require, at a minimum, changes and extensions to existing GC file formats. Changes in basic data formats are not welcome, but are certainly inevitable in long-term management of scientific data. The necessity of changes provides an opportunity to consider superior approaches to those of the past.

The XML [10] is an emerging information technology that is rapidly being adopted in standardization efforts across many industries and applications. XML structures data with a flexible and extensible language of tags. Document structures for particular applications can be specified using XML Schema [13] and documents can be formatted for web browsers with XSLT [12].

Based on XML, Duckworth proposed the Generalized Analytical Markup Language (GAML) for analytical data [43], including an XML Schema which formalizes the language. GAML uses the hierarchical nature of XML to efficiently represent relationships between arrays of numbers acquired from analytical instruments such as GC, GC-MS, liquid chromatography (LC) and LC-MS. XML is text-based, so GAML uses MIME base64-encoding of data values. Development of GAML is supported by Thermo, as a license-free, open standard. Although GAML has not yet gained wide acceptance, it points a promising direction and is a reference for badly needed efforts at standardization.

GC Image uses XML to format files describing analytical data. Analytical data is more efficiently stored in binary format. The XML files describe the binary data for operations accessing the data. GC Image uses XML for files containing several kinds of metadata (data about data) and information, including:

- metadata files including blob and graphics tables,
- templates with peaks and graphic objects,
- scripts and journals,
- projects with batches and
- reports.

Metadata files contain information about the data (e.g., dimensions), acquisition (e.g., date), visualization (e.g., color map) and processing (e.g., baseline removed from data). Blob tables contain information about detected peaks, including chemical name, chemical group, retention times, etc. Graphics tables contain information describing graphical objects and their relationship to the data. Templates describe patterns of peaks and graphics objects for pattern matching. Scripts and journals (described in Section 6) contain a sequence of operations to be executed or which have been executed. Projects (described in Section

10) organize batches of data in relationship with one another. Reports contain elements such as tables and graphs. XML Schema describe each of type of document and XSLT allows formatted viewing of documents with web browsers.

GC Image also supports importing and exporting of data and information from other formats, including CSV text, binary, digital image formats (e.g., JPEG [14]) and HyperText Markup Language (HTML) [44].

10. Project and batch management

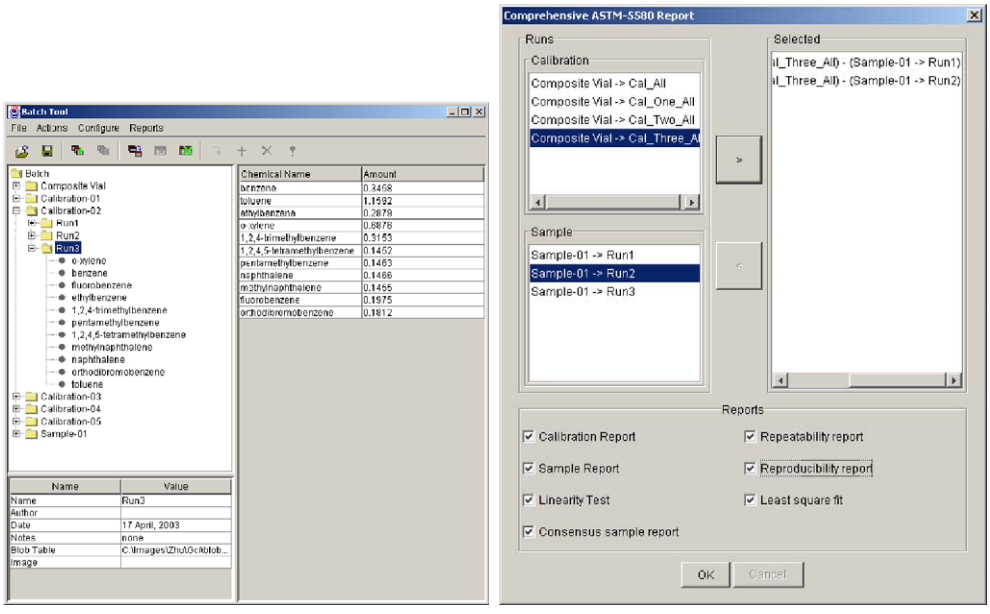
The U.S. Environmental Protection Agency (EPA) defines a batch as: "A group of samples which behave similarly with respect to the sampling or the testing procedures being employed and which are processed as a unit" (Ref. [45], p. 23). For example, batches consisting of multiple runs of several related standards mixtures are required for calibration in such standard methods as EPA 8000B [46] and ASTM D5580 [21]. The same batch relationships may exist for a variety of different reports and so the management of batches is distinct from but related to reporting.

Several operations are required for managing multiple data sets and analytical results in a project.

- An essential step is to specify the multiple chromatographic runs that are to be included in a project. This step requires the operations of creating a new project, adding runs with raw chromatograms to a project and saving a project to a file.
- It is useful to aggregate subsets of multiple runs in a project for the same mixture (and so are presumed to have the same chemical composition in the same concentrations), for example, in order to compute mean retention times, mean responses and precision of measurements. Therefore, the process of adding data sets to a project should support groupings of runs from the same mixture.
- Reporting operations may require data about the mixtures themselves. For example, the creation of a calibration function based on an internal standard requires not only the response ratio of the chemical to the internal standard (available from the chromatogram), but also the amount ratio (which is external to the chromatograms). The amounts are descriptive of a mixture (rather than the individual runs), so the concept emerges of a vial object to describe mixtures. In the project, the *vial* object is comprised of descriptive data and references to the runs taken from it.
- It may be desirable to make multiple analyses of the same chromatogram, for example to explore the effect of different algorithmic parameters. So, the *run* object is comprised of descriptive data, a raw chromatogram and possibly multiple alternative analytical results.

The conceptual structure that emerges from these considerations is a four-level hierarchy:

- A *project* consisting of descriptive project information and a collection of one or more vials.
- *Vials* consisting of descriptive mixture information and a collection of one or more chromatographic runs for each mixture.



A) Project management interface. B) Report configuration interface.

Fig. 9. GC Image interfaces for higher-level information management and reporting.

- *Runs* consisting of descriptive run information and possibly multiple analytical results for each run.
- *Analytical results* such as chemical peaks, each described by retention times, response, chemical name, associated internal standard, etc.

This hierarchical structure is represented naturally in tree, as shown in the GUI interface for project management pictured in Fig. 9A. The GUI supports data management operations to create, save and open projects and to add, view and edit vials, runs and analytical results (with limits on editing related to GLP). The hierarchical structure is represented naturally in files using XML.

11. Methods and reporting

Many types of reports can be generated from the information contained in the hierarchical structure created using the project management tool outlined in the previous section. A first step in reporting may be to create a batch—a collection of runs related for the purpose of reporting. Then, various types of reports are used, including retention-time reports, calibration reports, response-slice reports, fractional-response reports, fractional-amount reports, external-standard reports and internal-standard reports.

The richness of GC × GC data allows extensions of traditional reports. For example, GC × GC separations on boiling point and polarity effectively separate hydrocarbon classes, so in ASTM D2887 [47], the boiling point distributions of aliphatics and aromatics can be reported separately.

Different types of reports require different configuration parameters. GUIs allow users to conveniently configure reports to meet particular needs. For each type of report, the interface may support both a generic configuration and selectable restricted configurations that adhere to the specific requirements of published standard methods.

GC Image generates reports in several formats, including:

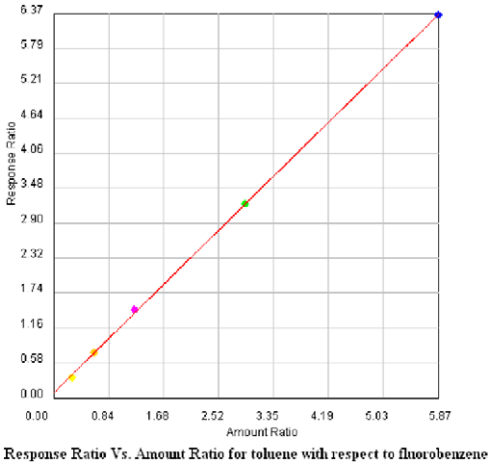
- CSV format that can be imported easily into other programs such as Microsoft Excel;
- HTML format that can be displayed easily in any standard web browser and edited with various editors, including Netscape Communicator or word processors; and
- XML format that can be interchanged easily with an accompanying XML Schema, displayed by an XML-capable browser such as Microsoft Internet Explorer using an accompanying XML Style Language Transformation (XSLT), or edited with XML-capable editors.

Two elements of a D5580 report, one graph and one table, rendered from reports formatted in XML are illustrated in Fig. 10.

12. Conclusion

Advanced information technologies offer powerful solutions for many of the challenges associated with GC × GC. Software systems such a GC Image, developed at the University of Nebraska-Lincoln, are developing and applying technologies such GUIs, computer graphics, digital image processing, pattern recognition, structured markup, and software architectures and engineering to the problems of handling, visualizing, processing, managing and reporting on GC × GC data.

In conclusion, two problems are noteworthy. First, automated recognition of chemicals in GC × GC data is technically difficult because of the large data sets, com-



A) Calibration function.

ASTM 5580 Sample Report

Chemical Name	Mass	Mass Percent
1,2,4,5-tetramethylbenzene	0.00223	0.04511
1,2,4-trimethylbenzene	0.07917	1.60144
benzene	0.04594	0.92936
ethylbenzene	0.12065	2.44040
methylnaphthalene	0.00603	0.12191
naphthalene	0.00751	0.15200
o-xylene	0.09606	1.94304
toluene	0.31273	6.32590
c9 aromatics	0.29993	6.06686
c10 aromatics	0.11134	2.25225
c11 aromatics	0.02243	0.45362
c9 plus aromatics	0.43370	8.77273

B) Sample report.

Fig. 10. GC Image report elements.

plex patterns of chemicals and high-dimensional transformation space. Solutions to this problem require additional research. Second, using data structuring technologies to provide enhanced access and interchange is both a challenging design problem and a challenge to the GC × GC community. The community has an opportunity to use a powerful new technology—the eXtensible Markup Language—to define new standards for representing GC × GC data and information. Standard structures would facilitate data access, portability and interchange. The solutions to this problem require creativity, but more importantly cooperation and leadership in the GC × GC community.

Acknowledgements

This work was supported by the National Science Foundation, Award DMI-0231746. Zoex (Lincoln, NE), www.zoex.com, provided GC × GC data.

References

[1] W. Bertsch, Two-dimensional gas chromatography. Concepts, instrumentation, and applications: Part 2. Comprehensive two-dimensional gas chromatography, Journal of High Resolution Chromatography 23 (3) (2000) 167–181.
[2] S.E. Reichenbach, M. Ni, V. Kottapalli, A. Visvanathan, E.B. Ledford, J. Oostdijk, H. Trap, Chemical warfare agent (CWA) detection with comprehensive two-dimensional gas chromatography (GC × GC), Chemical and Biological Sensing IV, Proc. SPIE, vol. 5085, (2003) i28–i36.
[3] J. Harynuk, T. Gorecki, C. Campbell, On the interpretation of GC × GC data, LCGC North America 20 (9) (2002) 876–892.
[4] U.S. Food and Drug Administration, Electronic records; electronic signatures; final rule, Tech. rep., HHS (1997).
[5] Environment Directorate, Chemicals Group and Management Committee, OECD principles on good laboratory practice (as revised in 1997), Tech. rep., Organisation for Economic Co-operation and Development (1998).
[6] Working Group on Information Technology, Good laboratory practice (GLP): guidelines for the archiving of electronic raw data in a GLP environment, Tech. rep., AGIT (2003).
[7] S.E. Reichenbach, M. Ni, D. Zhang, E.B. Ledford Jr., Image background removal in comprehensive two-dimensional gas chromatography, Journal of Chromatography, A 985 (1) (2003) 47–56.
[8] J. Gosling, Java: An Overview, Sun Microsystems, 1995.
[9] J. Gosling, B. Joy, G. Steele, G. Bracha, The Java Language Specification, Sun Microsystems, 2000.
[10] T. Bray, J. Paoli, C. Sperberg-McQueen, E. Maler, Extensible, Markup Language (XML) 1.0, World Wide Web Consortium, 2000.
[11] D. Brownell, SAX2, O'Reilly, Sebastopol CA, 2002.
[12] J. Clark, XSL Transformations (XSLT) Version 1.0, World Wide Web Consortium, 1999.
[13] D. Fallside, XML Schema Part 0: Primer, World Wide Web Consortium, 2001.
[14] Joint Photographic Engineering Group, Information technology—digital compression and coding of continuous-tone still images: requirements and guidelines, ISO/IEC-10918-1 (1994).
[15] Sun Microsystems, Programmer's Guide to the Java 2D API, SunMicrosystems, 2001.
[16] D. Geary, Graphic Java 2, Swing, vol. II, Sun Microsystems, 1999.
[17] H. Sowizral, K. Rushforth, M. Deering, The Java 3D API Specification, Addison-Wesley, 2000.
[18] M. Segal, K. Akeley, The OpenGL Graphics System: A Specification (Version 1.4), Silicon Graphics, 2002.
[19] M. Ni, S.E. Reichenbach, GUI State-Based Accessibility Control in Hierarchical State Machines, Submitted for publication.
[20] Sun Microsystems, Programming in Java Advanced Imaging, Sun Microsystems, 1999.
[21] ASTM, Standard test method for boiling range distribution of petroleum fractions by gas chromatography, Tech. Rep. D5580-02, ASTM, West Conshohocken PA (2002).
[22] B. Myers, M. Rosson, Survey on user interface programming, ACM Proceedings SIGCHI'92, 1992, pp. 195–202.
[23] Object Management Group, OMG Unified Modeling Language specification, version 1.4, http://www.omg.org/cgi-bin/doc?formal/01-09-67.pdf (2001).
[24] B.J. Jansen, The graphical user interface: an introduction, SIGCHI Bulletin 30 (2) (1998) 22–26.
[25] J. Cooper, The Design Pattern: Java Companion, Addison-Wesley, 1998.
[26] J. Foley, A. Dam, S. Feiner, J. Hughes, Computer Graphics: Principles and Practice in C, Addison-Wesley, 1995.
[27] J. Yellen, J. Gross, Graph Theory and Its Applications, CRC Press, 1998.
[28] J. Imperial College Department of Computer Science, FOLDOC: Free On-Line Dictionary of Computing, Available at: http://foldoc.doc.ic.ac.uk/, 2001.
[29] S. Beucher, C. Lantuejoul, Use of watersheds in contour detection, International Workshop on Image Processing, Real-Time Edge and Motion Detection/Estimation, 1979, pp. 17–21.
[30] S.E. Reichenbach, GCxGC Blob Metadata and Statistics in GC Image, Available at: www.gcimage.com/statistics.pdf, 2003.
[31] S. Venkatasubramanian, Geometric shape matching and drug design, PhD thesis, Stanford University (1999).
[32] D.P. Huttenlocker, S. Ullman, Object recognition using alignment, Proc. International Conference on Computer Vision, 1987, pp. 102–111.
[33] D.H. Ballard, Generalized hough transform to detect arbitrary patterns, IEEE Transactions on Pattern Analysis and Machine Intelligence 13 (2) (1981) 111–122.
[34] H.J. Wolfson, I. Rigoutsos, Geometric hashing: an overview, IEEE Computational Science and Engineering 4 (4) (1997) 10–21.
[35] W.E.L. Grimson, Object Recognition by Computer: The Role of Geometric Constraints, MIT Press, Cambridge, MA, 1990.
[36] D. Huttenlocher, G. Klanderman, W. Rucklidge, Comparing images using the Hausdorff distance, IEEE Transactions on Pattern Analysis and Machine Intelligence 15 (9) (1993) 850–863.
[37] W.J. Rucklidge, Efficient visual recognition using the Hausdorff distance, Lecture Notes in Computer Science 1173.
[38] T.M. Breuel, Fast recognition using adaptive subdivisions of transformation space, Proceedings-IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1992, pp. 445–451.
[39] T.M. Breuel, A practical, globally optimal algorithm for geometric matching under uncertainty, Electronic Notes in Theoretical Computer Science, vol. 46.
[40] M. Ni, S.E. Reichenbach, A statistics-guided progressive RAST algorithm for peak template matching in GCxGC, IEEE Workshop on Statistical Signal Processing, 2003, pp. 369–372.
[41] M. Ni, Q. Tao, S.E. Reichenbach, MCMC-based peak template matching for GC × GC, IEEE Workshop on Statistical Signal Processing, 2003, pp. 497–500.
[42] M. Ni, S.E. Reichenbach, Hierarchical Point Pattern Matching, Submitted for publication.
[43] J. Duckworth, An XML-based file format for archival storage of analytical data, 2001.
[44] D. Pemberton, et al, XHTML 1.0: The Extensible HyperText Markup Language, World Wide Web Consortium, 2000.

[45] U.S. Environmental Protection Agency, Test Methods for Evaluating Solid Waste, Physical/Chemical Methods, 1996, SW-846 (Revision 1, 1992; Revision 2).
[46] U.S. Environmental Protection Agency, Determinative chromatographic separations, Tech. Rep. Method 8000B, EPA, Washington, DC, 1996.
[47] ASTM, Standard test method for boiling range distribution of petroleum fractions by gas chromatography, Tech. Rep. D2887-01a, ASTM, West Conshohocken, PA, 2002.