An Infrared Spectrometer for Process Monitoring II, Chemometry and Automatization

Peter M. Hintenaus
C. Doppler Laboratory
Embedded Software Systems
Universität Salzburg
Jakob Haringer Straße 2
5020 Salzburg, Austria
Email: peter.hintenaus@cs.uni-salzburg.at

Wolfgang Märzinger
RECENDT GmbH
Hafenstraße 47-51
4020 Linz, Austria
Email: wolfgang.maerzinger@recendt.at

Helmut Pöll
EEVG Entsorgungs- und
Energieverwertungs GmbH
Fabriksplatz 1
4662 Steyrermühl, Austria

***Abstract*—We describe a hard and software architecture for process control applications in the chemical, pharamceutical and food industries based on spectroscopic measurements. We argue for the tight integration of the spectrometer itself, the data analysis software and the measurement automatization to achieve situation awareness and predictible real-time behavior and to be able to handle complicated sampling situations, while keeping the programming effort for an individual installation low.**

I. Introduction

Infrared spectroscopic methods find widespread use in analytical chemistry. When monitoring processes a chemist knows the substances contained in the samples, but the concentrations of the individual substances are unknown. For performing an analysis infrared light is passed through a sampling station where the light interacts with the sample being analyzed. This interaction results in the absorption of some specific wavelengths of the infrared light depending on the chemical composition of the analyte. The spectrum of the light that has passed the analyte is evaluated by a so-called chemometric model which produces the concentrations. Analytic chemists derive chemometric models from the spectra of controlled experiments using statistical methods such as partial least square regression, see e.g. [1].

In [2] we describe a Fourier transform infrared spectrometer, see e.g. [3], for process monitoring applications. This spectrometer can easily be adopted to different measurement tasks, by configuring the tradeoff between resolution on the one hand, and the time required for taking a single spectrum on the other. The number of spectra averaged to produce a single measurement is also configurable. The signal to noise ratio of this instrument is 20000 for a resolution of $3\mathrm{cm}^{-1}$ and a measurement time of 1 second.

In this paper we first describe the requirements for automatized measurement systems for process control. Next we discuss the architecture of our system and its main components, the chemometry engine and the measurment automatization. A case study rounds out this work.

II. Requirements for Automatized Measurment Systems

Devices used for automated measurements in process applications have to operate unattended for prolonged periodes of time. When compared to a lab environment, where a human operator is available for handling of the samples, data entry and similar tasks, operating measurent equipement in a process setting for routine measurements poses additional requirements:

A. Situation Awareness

Both the measurement system itself and the data analysis software have to be aware of the state their environment is in. Consider an installation with several sampling stations, which are scanned by an optical multiplexer. For each sampling station the spectrometer will have to switch its signal conditioning amplifiers to a different gain, in order to accomodate for different lighting situations. The data analysis software on the other hand will have to use different procedures for computing the analysis.

B. Complex Sampling Procedures

Contamination of the sampling station is a problem in many applications. Provisions for the automatic initiation of periodic cleaning operations have to be taken. When measuring powdery substances samples might have to be extracted by some extraction mechanism from the process. Both the spectrometer and the data analysis software have to be synchronized with the sample extraction.

C. High Speed with Predictable Timing

Quality control applications in the pharamceutical and food industries for example demand for the evaluation of up to 10 individual objects per second. An evaluation typically consists of detecting the presence of an object, taking one or more spectra of this object, running a chemometric procedure and accepting or rejecting the object depending on the outcome of the chemometric analysis. The time allowed for one evaluation is fixed by the design of the mechanical apparatus performing the actual acceptance or rejection. Although missing some objects might be inevitable, this deadline can only be missed very rarely, in order to avoid having to reject a large number of objects that have not been evaluated.

III. Architecture

The hardware of our instrument includes two signal processing clusters consisting of two ADSP-21161 digital signal processors [4] each and an ARM-9 [5]. The ARM-9 is responsible for passing data to and from the two DSP clusters, for communicating with the outside world using LAN and for the interface to the production process. It runs the LINUX operating system. One of the DSP clusters is dedicated to computing spectra, the other is used for executing chemometric models.

The software consists of three main parts, the computation of interferograms and spectra, a measurement automatization component and the chemometry engine. The automatization component is responsible for controlling the operation of the device. It receives the spectra produced by the spectrometer, it passes these along to the chemometry engine to receive the analysis results in turn and it interacts with the production process. The chemometry engine executes user-defined chemometric procedures under the control of the automatization component. A chemometric program can contain several prcedures.

A. Computing Interferograms and Spectra

A Fourier transform spectrometer consists of a Michelson interferometer (Figure 1) with one moving mirror, a detector and the electronics for driving the mirror and for processing the signal gathered by the detector. A beam of light enters the interferometer from below, to be split at the beamsplitter into two halfs, one directed at the fixed mirror, the other at the moving mirror. These two halfs are reflected at their respective mirrors and recombine at the beamsplitter. One part of the recombined beam is reflected towards the the detector, the other back into the source. The intensity of the beam reflected towards the detector depends on the displacement $d = |a-b|$ of the moving mirror and the spectrum S of the incomming beam. For $d = 0$ all light is reflected towards the detector and none back to the source. The interferogram I that is reflected into the detector when the moving mirror travels at constant speed is basically the Fourier integral of S. By applying the inverse Fourier transform to a sampled version of I the Spectrum is computed.

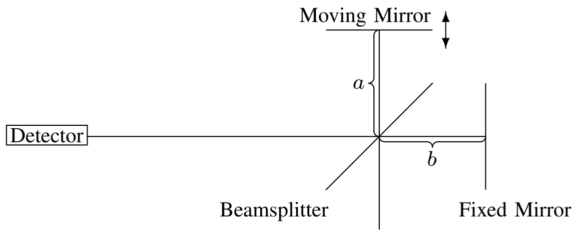

Fig. 1. Michelson Interferometer

In our device the mirror is driven sinusodually by a voice coil, therefore the signal the detector produces is modulated by the speed of the mirror. We demodulate this signal by using the interferogram produced by a laserbeam, which is coupled into the interferometer in parallel with the light we want to analyze.

B. The Chemometry Engine

The chemometry engine is a stack machine, see e.g. [6], combined with a memory for data that shall be preserved between procedure invocations. A single entry on the stack or in the memory can hold either a scalar or a vector. When a chemometric procedure is invoked by the automatization component, the parameters are passed on the stack. Upon completion, a procedure passes back zero or more results, and the stack is flushed. The engine is programmed in an assembly-like language, which is interpreted at run-time. To keep the overhead due to the interpretation of the individual instructions low, the instruction set has been designed to be as high-level as possible. As conditional execution has been moved into individual instructions themselves, the chemometric procedures represent straight-line programs only. Error-checking is performed at run-time. For preprocessing spectral data the engine offers instructions such as computing absorbance and transmission, filtering including differentiation, correction of the offset and the slope, computing various versions of normed spectra, and computing a multiplicative scatter correction. The arithmetic instructions operate on any combination of scalars and vectors as long this combination is mathematically justified. Additional instructions for manipulating vectors are provided.

Currently the chemometry engine is optimized for the evaluation of chemometric models derived by multivariate statistical methods. By adding a few instructions it can be extended to also cover models based on e.g. neural networks.

Invoking a chemometric procedure with a spectrum consisting of 2000 points takes about 3ms. The execution times of digital signal processors we use is very predictable, so well-defined real-time behavior is provided.

C. Measurement Automatization

Most sensors used in factory automatization exhibit negligible delay between the time a measurement was taken, and the time the result is available to the rest of the system. This is not true when working with spectral information. In its fastest mode of operation (80 spectra per second) our device exhibits a delay of about 60mS between the time a measurement was taken and the availability of the spectrum based on this measurement. For high-speed applications this timing behavior precludes a purely time-triggered approach for handling the spectral data, as offered by commercially available programmable logic controllers. Moreover, the volume of data contained in a single spectrum is huge, when compared to what factory automatization systems have to handle. To cope with these problems we designed a programming language called STAMPED [7] that supports timestamps explicitly for describing temporal relations. To provide a compact representation of the automatization program that can be handled easily with existing tools (e.g. version management, difference listers, editors) STAMPED is purely textual.

The main unit of execution in STAMPED is a task. A task can either be invoked periodically or, in contrast to many other automatization languages, as a reaction to an event. Tasks communicate using shared variables. The access to shared variables is guaranteed to be atomic regardless of type. No other synchronization for tasks is provided.

The description of subsystems that can interface with STAMPED are called components. A component variable can produce events for triggering task invocations, it may contain an object dictionary for sharing data with the STAMPED program, and procedure definitions for the procedures that can be called from STAMPED. Components in our STAMPED system are a spectrometer, a datastore for archiving measurement data and a CANopen field bus [8] for communicating with devices like digital and analog in- and outputs, motordrives and the like. Components like the spectrometer that require special hardware allow only one instance of the component which must have a specific name. This constraint is checked during start-up of a STAMPED program. Other components like the datastore can be instantiated several times.

An event may have variables associated with it, which are updated with new data everytime the event fires. These variables are inherited by the task that reacts to the event in the form of local variables. Besides integer and floating-point, STAMPED supports binary large objects (BLOBs) for moving data of variable length between component variables. The memory for these binary large objects is reclaimed automatically once the memory cannot be referenced anymore by using a reference counting scheme. Arrays and records of arbitrary complexity are also possible. When compared with IEC 61131 structured text [9], STAMPED allows for several tasks in one program, moreover it adds event-triggered tasks, components and binary large objects.

```
COMPONENT Spectrometer
   EVENT SpectrumAvailable
      TIMESTAMP mStart, mEnd;
      FLOAT loWN, hiWN;
      BLOB Spectrum
   END;
   PROCEDURE setIrGain(INTEGER gain);
   PROCEDURE StopMeasure();
   PROCEDURE StartMeasure();
END.
```

Fig. 2. A Component Definition

To provide a flavor of STAMPED we present two short samples. In figure 2 the component definition of `Spectrometer` is shown. The event `SpectrumAvailabe` provides the time the measurement was started and ended, the lowest and the highest wavenumber in the spectral data and the data itself to the task that serves it. Three procedures allow setting the gain of the signal conditioning amplifier, stopping the measurement process and starting it again.

```
PROGRAM

IMPORT Chemometry;
IMPORT Spectrometer

VAR
   Chemometry ash;
   Spectrometer FtIr;
   FLOAT Concentration

TASK ChemicalAnalyst
   ON FtIr.SpectrumAvailable
BEGIN
   Concentration := ash.analyze(Spectrum)
END
...
```

Fig. 3. A STAMPED Program

In figure 3 the task `ChemicalAnalyst` is invoked everytime a new spectrum is available. It sends the spectrum to the chemometric procedure `analyze` in the program `ash`. The result is posted to the global variable `Concentration` where other tasks can read it.

For describing a task's operations, STAMPED supports procedures, optionally with several return values, if-statements and for-loops. While-loops have been omitted on purpose, as their real-time behavior is hard to predict.

IV. Advantages of Tight Integration

By integrating an automatization component into our spectrometer we achieve situation awareness. The application program executed by the automatization component either controls the state of the surounding of the measurement system, or at least senses it. Therefore it can control the spectrometer and invoke the correct chemometric procedures as required by the situation at hand. For high-speed applications tight integration provides for high analysis rates and short and predictable delay times. When measuring the humidity of substrates made from sucrose we were able to achive 80 measurements per second, with a precission of less than 0.1% and a consistent delay between the time the measurement was taken and the report of the result via CANopen of less than 100mS, fast enough for automated quality control of these substrates. Complex sampling situations can be handled by the automatization component, using fieldbus based remote peripheral devices.

V. Case Study

Flyash resulting from incineration of residuals of paper production in powerplants is used as additive by the cement industry. For the additive to be acceptable the cloride content has to be kept low. In [10] the effect of other additives on the quality of concrete has been investigated. Initial laboratory measurements, applied to samples of flyash, using a Bruker Vertex 70 laboratory spectrometer suggested that the cloride content can indeed be determined by spectroscopic methods. First trials at the process showed that any sampling station that is exposed to the ash-stream permanently will get clogged in a short period of time. Therfore a sampling mechanism operated by four pneumatic cylinders was constructed that periodically extracts a sample of ash.

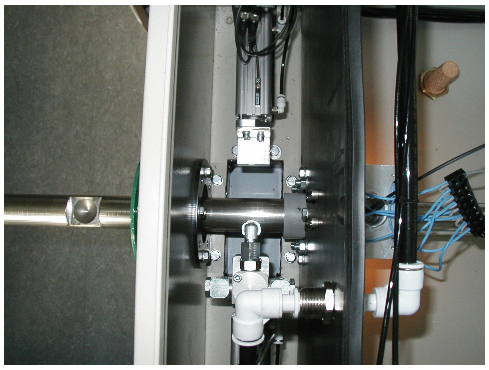

Fig. 4. Sampler: Spoon (left), Cleaning (middle) and Sampling (right)

Figure 4 shows part of the sampling station. Two cylinders move a spoon first into the ash-stream, for filling it with ash, then to the intermediate position, where the ash in the spoon is leveled by the third cylinder, then to the measurment position, where spectra are taken, and then back to the intermediate position, where the forth cylinder empties the spoon using a brush and a jet of air. The complete sampling procedure is controlled by a STAMPED program consisting of about 400 lines of code.

For modelling the data fourteen samples with cloride concentrations from 0.18% to 0.45%, covering the range than is expected during production, were prepared in the laboratory. From the concentrations of the samples together with their spectra, loading vectors were computed using the partial least squares algorithm. The loadings were applied to the spectra collected over a period of one month to predict the cloride concentrations offline.

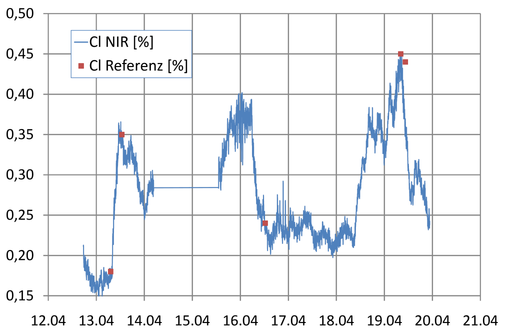

Fig. 5. Predictions and lab references for a 9-day run

The graph in figure 5 shows the predictions together with the results of some laboratory measurements taken during an 9 day period starting on April 12th of 2010. About 30000 spectra were taken during this period. On April 14th the sampling mechanism got stuck and no samples where taken for a one-day period. In table I we compare the chlorine concentration of 18 samples determined by laboratory measurements, with the predictions our system computed at the time the sample was taken. The average error is 0.0048% (0.0011% without sample 13), the variance is 0.00038 (0.00015). Although our preliminary model has to be improved by incoporating more laboratory measurements, the cloride content is prediced with acceptable accuracy.

Sample	Cl [%]	Prediction [%]	Error [%]
1	0.22	0.232	-0.012
2	0.24	0.243	-0.003
3	0.31	0.300	0.010
4	0.31	0.305	0.005
5	0.32	0.321	-0.001
6	0.33	0.335	-0.005
7	0.20	0.212	-0.012
8	0.20	0.207	-0.007
9	0.18	0.173	0.007
10	0.35	0.337	0.013
11	0.24	0.244	-0.004
12	0.45	0.437	0.013
13	0.44	0.373	0.067
14	0.37	0.361	0.009
15	0.35	0.350	0.000
16	0.35	0.345	0.005
17	0.33	0.304	0.026
18	0.28	0.306	-0.026

TABLE I
CHLORINE CONCENTRATIONS ACCORDING TO LABORATORY ANALYSIS COMPARED WITH PREDICTED VALUES

VI. Conclusion

We have identified requirements for the application of soft-computing methods for automated process control. We have described our approach towards satisfying these requirements. We have demonstrated a successful application of our system. One big problem remains: when running data analysis procedures without human supervision some measure of confidence in the results shall be computed by the procedures themselves, so that alarms can be raised automatically once the results cannot be trusted anymore.

References

[1] R. G. Brereton, *Chemometrics: Data Analysis for the Laboratory and Chemical Plant*. John Wiley and Sons, Ltd., 2003.
[2] P. M. Hintenaus, G. Kvas, and W. Märzinger, "An infrared spectrometer for process monitoring I, spectroscopy," in *Proceedings of the 33rd Annual Conference of the IEEE Industrial Electronics Society (IECON)*. IEEE, 2007.
[3] J. Kauppinen and J. Partanen, *Fourier Transforms in Spectroscopy*. Wiley-VCH, 2001.
[4] *ADSP-21161 SHARC Processor Hardware Reference*, 4th ed., Analog Devices, February 2005.
[5] *NS9750B-A1 Hardware Reference*, Digi International Inc., March 2008.
[6] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*, 3rd ed. Morgan Kaufmann publishers, 2003.
[7] P. M. Hintenaus, "Stamped language definition," C. Doppler Laboratory Embedded Software Systems, University of Salzburg, Tech. Rep., 2010, to appear.

[8] *CANopen application layer and communication profile*, CAN in Automatization, 2006, version 4.1.

[9] *Programmable Controllers - Part 3: Programming Languages*, 2nd ed., IEC.

[10] D. Traber, F. Jacobs, U. Mäder, and U. Eggenberger, “Einsatz von Sekundärstoffen im Beton: technische und ökologische Anforderungen,” *Betonwerk und Fertigteil-Technik*, November 2000.