MULTIVARIATE DATA PROCESSING SYSTEM: TRANSPUTER BASED DATA ACQUISITION, ANALYSES AND PRESENTATION.

R.G. Belleman, P.M.A. Sloot* and L.O. Hertzberger
High Performance Computing Group,
Department of Computersystems, University of Amsterdam
Kruislaan 403, 1098 SJ Amsterdam, The Netherlands.
e-mail: robbel@fwi.uva.nl or peterslo@fwi.uva.nl

1. INTRODUCTION

In previous years we have developed an expert system for the analysis and control of a biophysical cell separation experiment known as Centrifugal Elutriation [1,2,3,4]. Centrifugal elutriation is a technique in which differences in sedimentation velocity of human peripheral blood cells are exploited to isolate various types of cells from inhomogeneous cell populations. These cells can then be used for clinical and fundamental research applications.

We designed and built a three-parameter light scattering device that simultaneously measures Forward Scattering (FS), Side Scattering (SS) and Back Scattering (BS) interfaced to the CE system. We have shown that it is possible with these three parameters to discriminate between various (sub-)populations present in a mixture [1].

In this so called Computer Assisted Centrifugal Elutriation (CACE [5]) system we integrated a dedicated data acquisition module with monitoring primitives and off-line software to analyze the experimental multivariate data, including the detection of the various sub-populations that are present in a mixture. This allows for controlled centrifugal elutriation and facilitates optimal cell separation where previously specialized knowledge and time consuming cell identification techniques were required.

One major drawback of the system is the relative inflexibility with respect to the number of studied parameters and the inability to scale the computational power accordingly. It would favour the overall usability of the system if sufficient information could be obtained *during* a separation experiment instead of afterwards, transferring the acquired data to an off-line system and analyzing it there. Besides, the inherent complexity of this type of expert systems used in a non-expert environment requires a structured extendable design and a well considered user interface.

In this paper we show that many of the data analysis programs used in the system have an inherent parallelism that can be exploited if implemented on a parallel architecture.

Section 2 provides an overview of the data acquisition used in this system, the heterogeneous hardware and the low-level routines that access it. Section 3 shows how we parallelised the off-line programs into a pipe-line of flexible, non-interactive parallel data analysis tools that can be scaled to the number of transputers present. Section 4 provides insight in the user-interface we have used for this system and finally, section 5 offers a conclusion and some notes on further research.

* Author to whom all correspondence should be addressed.

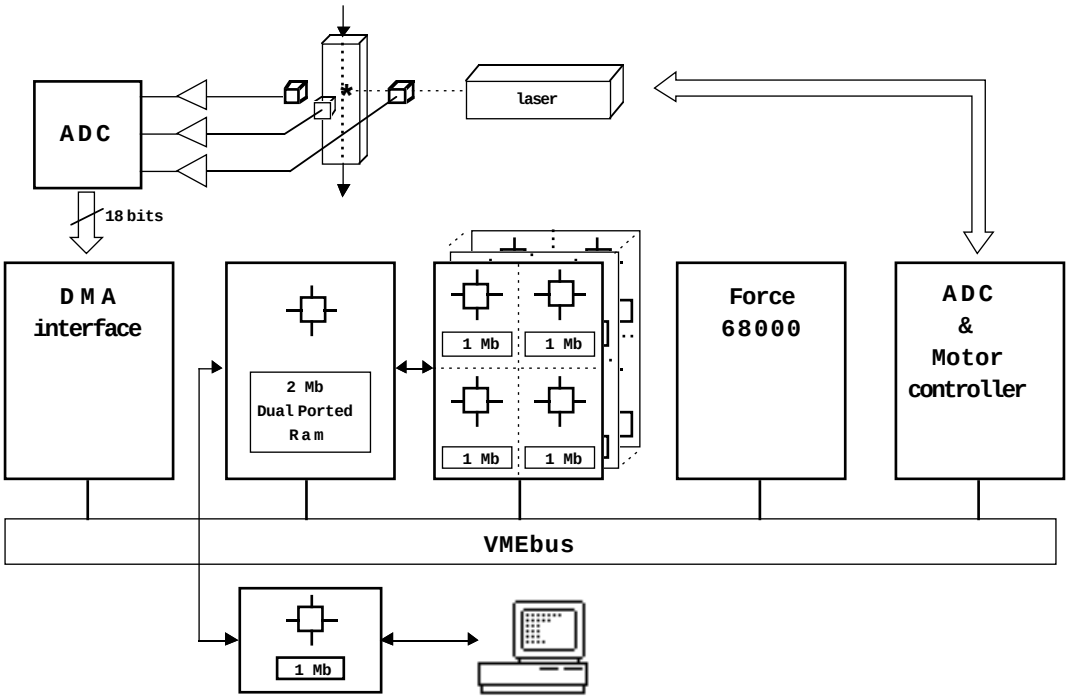

Figure 1. Hardware configuration of the acquisition and analysis system.

2.1 Hardware description

The data acquisition and analysis system is designed around the VME-bus specification (Figure 1). The use of this interfacing system makes it possible to build the system out of easily available standard building blocks thus ensuring both expandibility, in terms of interfaces that are needed to cover new applications, and the possibility to scale computational power.

The data acquisition is achieved by means of a special purpose data acquisition module. Analog signals from three light scattering detectors are each pre-amplified and then converted to a 6 bit digital value. This 18 bit pattern is used to address a 512 kilobyte memory area on a VME T800 transputer module with 2 Mb of Dual-Ported dynamic RAM (Parsytec BBK-V2). By means of a Read-Modify-Write scheme using Direct Memory Access (DMA) the contents of the 16 bit memory-element corresponding with each pattern occurrence is incremented by one (up to a maximum of 2^{16} - 1) thus creating a histogram of data in the transputer module's memory. This histogram represents the light scattering information from all cells that were detected during a separation experiment. It can be visualized by a box of 64^3 elements (Figure 5.1). The use of Dual-Ported memory on the transputer module allows the DMA interface to access memory locations without interfering with accesses that are performed by the main transputer in the remaining 1.5 Mb.

The computational power of the complete system is significantly enhanced by a second transputer module containing 4 extra T800 transputers, each with 1 Mb of dynamic RAM (Parsytec VMTM). Communication from this board to the main transputer takes place via the externally available link-connectors while the 4 transputers on one board can communicate with each other using the on-board C004 link configuration devices. The computational power can be further enhanced by placing more boards, provided this is supported by the software.

Added to the VME system is a processor module designed around the Motorola 68000 processor (Force SYS68K). An implementation of an autofocussing algorithm running on this processor uses an Analog to Digital Converter (ADC) to digitize the

scattering signal from one of the detectors and a motor controller to move the optics into focus. The 68000 communicates with the main transputer via a link-interface that is accessible from the VME bus.

The VME-system communicates with a Macintosh II host computer via a transputer link between the main transputer on the VME-system and a T800 transputer that is inserted into the NuBus of the Macintosh (Parsytec TPM-MAC). This transputer is used as the development platform and, during experiments, takes the function of interface between the data acquisition and analysis system and the user-interface running on the Macintosh.

All programs are written in occam 2 and developed under the Mac Programmer's Workshop using an adapted version of the Inmos D705-B development tools ("IServer tools") [6,7].

2.2 Software description

Occam 2 forces a programmer to structure his applications in a way that differs from conventional languages [6]. Because of the lack of dynamic variables and abstract data structures in the language it is sometimes hard to make an abstraction from underlying reality. The structure that remains, the process, is in many cases the only, but often quite elegant escape from this demerit (Figure 2).

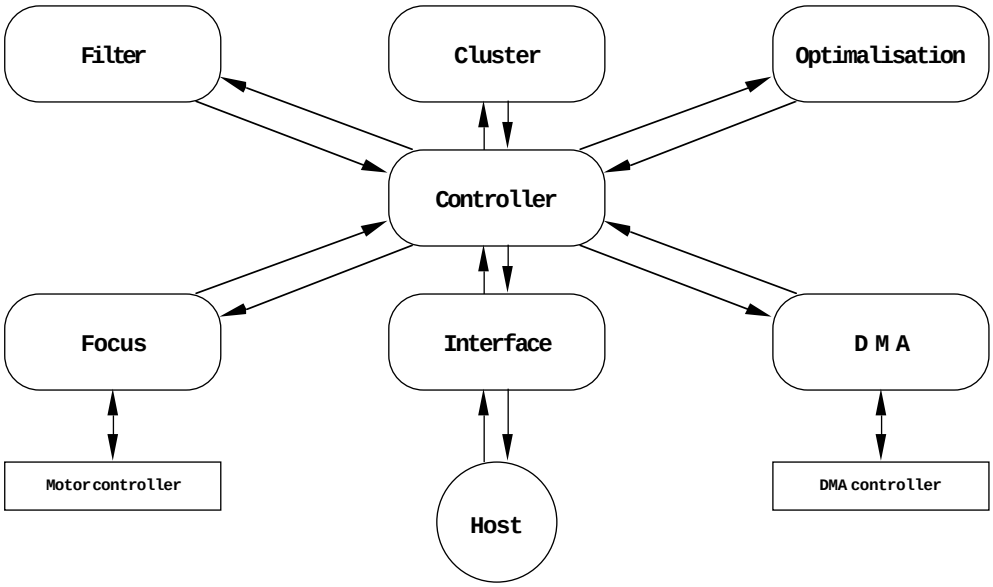

Figure 2. Process configuration of the analysis system.

In order to access the heterogeneous hardware, without interference to the data analysis software, an abstraction is indispensable. On conventional (read: sequential) systems this is often achieved by means of device-libraries that contain primitives to access and manage the underlying hardware. However, on a parallel platform this scheme is difficult to implement because of the lack of dynamic variables and abstract data structures. Therefore, the concept of device processes provides a logical alternative [8]. These processes allow individual actions from both hard- and software independent of any other process active. Interfacing devices to user-programs in this way provides a high degree of transparency to the programmer and flexibility towards the user of the end-product.

Communication between user-programs and device processes takes place via the conventional occam channels. Actions to be taken by a device process are sent via a special device-protocol which consists of a token representing the action to be taken,

followed by optional parameters. In addition to this command channel, a status channel is available enabling a user-program to request information about the current status of a device.

By introducing structure into occam 2 programs by means of processes, an application can be seen as a set of data analyses and support modules. These modules merely need to be "connected" to the user's programs prior to compile-time using a single configuration file. This supports the construction of a data processing pipeline as we describe in the following section.

3. PARALLEL DATA ANALYSES PIPELINE

As described in previous sections the resulting data from a typical experiment can be represented by a 3-parameter box (3 x 64 channels). Each x-y-z combination of parameters contains an integer number of detected events (see Figure 5.1).

In this section we describe the functional and implementational aspects of parallel algorithms that transform this complex dataset in $\Re^3$ into a set of functions describing the parameterized populations contained in the dataset.

First stochastic noise is removed by a discrete filtering procedure resulting in a smooth dataset. Next, the filtered data is transported to a parallel clustering algorithm that produces an estimation of the parameters describing the (sub)populations contained in the dataset. Finally a parallel iterative optimization algorithm manipulates the parameters from the (sub)populations to produce an optimal fit to the original data. This is schematically shown in the next figure:

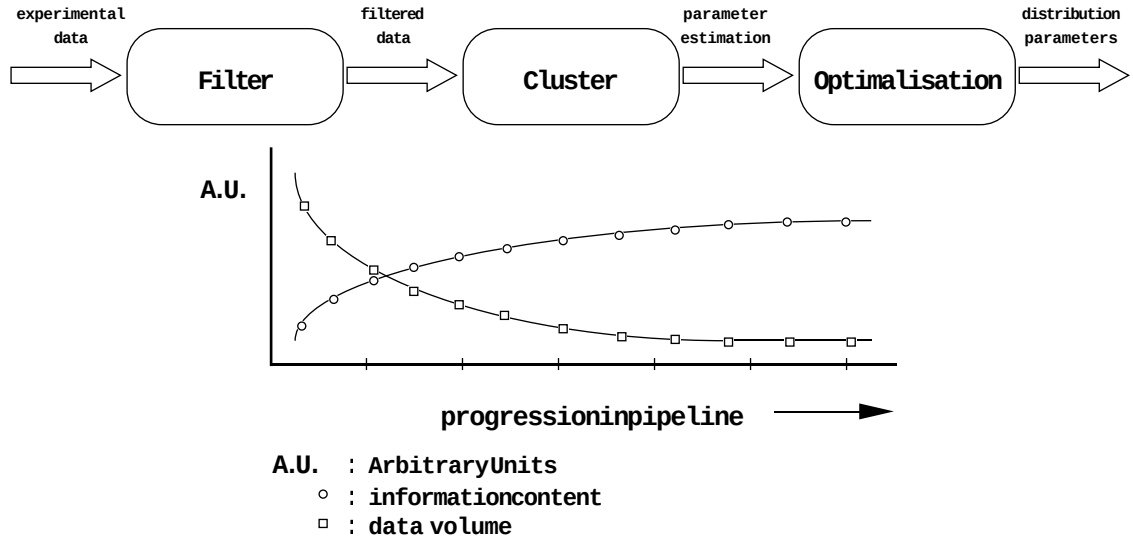

Figure 3: the parallel data analyses pipeline

3.1 Parallel N-dimensional digital filter

Apart from the instrumental noise that is known to be present in the analog signals from the detectors and amplifiers, quantizing the analog data during the conversion by an Analog to Digital Converter (ADC) also adds stochastic noise. In previous work we have shown that the frequency content of the data is extremely regular [2]. This resulted in a digital filtering procedure that removes stochastic noise and guarantees consistency of the data contents.

In this section we present a parallelization scheme for the Parallel Fourier Deconvolution that constitutes the filtering kernel.

The parallel filter performs a three dimensional Fast Fourier Transform (FFT) on the raw data by first distributing the data as two parameter planes over 64 processes. Depending on the number of transputers available, a number of these processes run time-sliced on the same transputer. In each process a real_to_complex FFT procedure transforms the x-direction. We are concerned with a low-pass filter, so the (high) noise frequencies to be removed are easily located in the transformed arrays (they reside in the middle of the array). Therefore, this part of the data can be skipped in the following steps of the transformation (Figure 4.1).

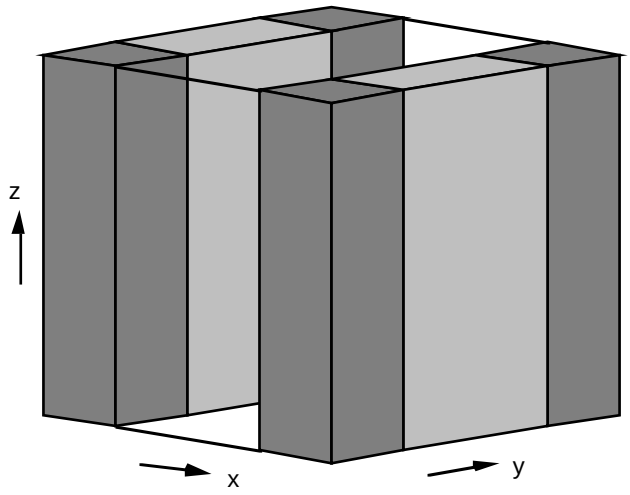

Figure 4.1 By filtering directly after the first transformation, the non-shaded area does not have to be transformed in the proceeding steps. In the third step, only the dark shaded area has to be transformed.

Also, the used input data of the filter is real, i.e. the imaginary part is zero. As a consequence the transformed data is now Hermitian. This implies that half of the data can be skipped due to the symmetry in the data (Figure 4.2) [9].

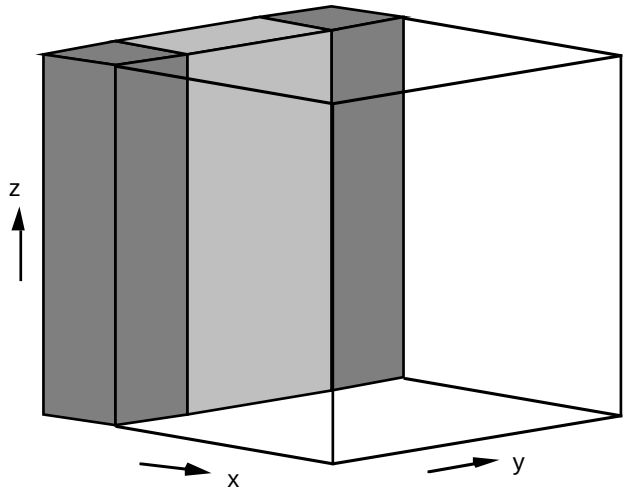

Figure 4.2 By skipping the upper half of the data after the first transformation, the second transformation has to be performed through the whole shaded box and the third transformation only through the dark shaded boxes.

The channels that are left are then transformed in the y-direction by a complex_to_complex FFT procedure. For the transformation in the z-direction, a reordering must take place as each process does not have all the necessary data (Figure 4.3). It can be shown that communication overhead in this stage is significantly reduced by only exchanging relevant data elements on a bidirectional ring topology [10].

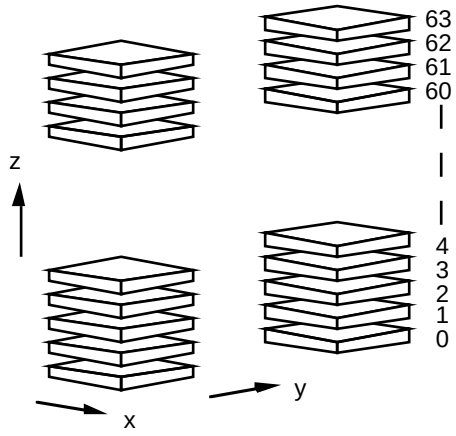

Figure 4.3 Databox transformed in 2 dimensions with filtered noise frequencies.

Finally the filtered data should be transformed back to the spatial domain. First the data is inverse transformed in the z-direction by a complex_to_complex iFFT procedure. After reordening, an iFFT procedure transforms the data in the y-direction followed by a Hermitian inverse_complex_to_real HiFFT procedure.

The result of this filter is shown in the following illustrations:

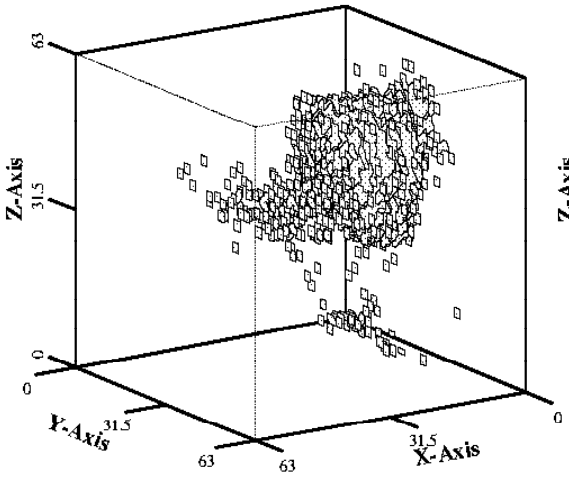

Figure 5.1 Raw data

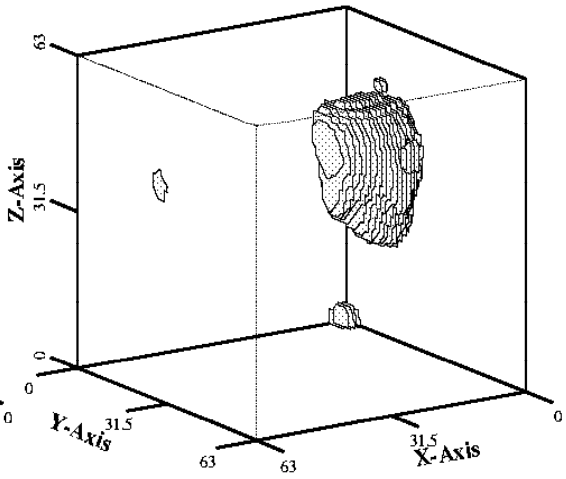

Figure 5.2 Filtered data

Measurements have shown an almost linear speedup of the filter and an efficiency of approximately 0.7.

3.2 Parallel cluster analyses

The process of cluster analyses tries to find a number of classes in a larger dataset where members of a class have a high degree of similarity to each other while the different classes are relatively distinct. Many methods of finding these clusters have been reported in literature [11,12,13], here we describe a specialized algorithm that assumes that each individual cell population can be described by a multivariate normal distribution [3].

For a clustering operation in four dimensions (three parameters), the output of the filter procedure is taken (which at the end of the process leaves a two parameter plane on each of the 64 processes) and transported to a two parameter clustering procedure. Each

of the 64 clustering processes makes an estimation of the parameters describing the populations in its two parameter plane by scanning for local maxima in both x- and y-direction. The result of this is a skeleton of maxima for each detected distribution. Local maxima are then eliminated (i.e. considered not to be part of a distribution) according to the following criteria:

- each distribution that has a density less than a pre-defined value is discarded,
- each distribution that has a width less than a pre-defined value is discarded,
- two distributions that are closer to each other than a pre-defined value are "merged" to one distribution.

After this, the percentage of overlap between neighboring distributions and their relative orientation is estimated. This overlap is used to determine the "purest" part of a distribution. Finally, the initial parameters for every distribution can be determined by fitting the maxima to normal functions.

Clustering results in a dramatic data reduction in that each of the distributions is described by a skeleton of maxima secluded by an integration area. 98.9 Percent of the distribution will be contained into this integration area by making the area twice the standard deviation. This is sufficient and efficient for further processing of the parameters. Optimization of the initial parameters is achieved by means of an iterative optimization process as described in the next section.

3.3 Statistic optimization

Maximum Likelihood (ML) is a statistical technique in which the parameters that describe a density function are estimated. Here, we apply a special Expectation Maximization (EM) algorithm that determines the ML estimates of multivariate normal distributions using an iterative function [3].

A mixture of multivariate distributions can be represented by

$$P(\bar{r}|\Phi) = \sum_{i=0}^{m-1} \alpha_i p_i(\bar{r}|\phi_i) \; ; \; \bar{r} = (x,y,z)^T \in \Re^3. \qquad (1)$$

Where $0 \le x,y,z \le 64$ and the number and fraction of the populations are represented by m and α. Each multivariate distribution is parameterized by

$$\Phi = (\alpha_0, \alpha_1, \ldots, \alpha_{m-1}; \phi_0, \phi_1, \ldots, \phi_{m-1}), \qquad (2)$$

where

$$\phi_i = (\bar{\mu}_i, \bar{\bar{\Sigma}}_i) \; . \qquad (3)$$

$\bar{\mu}_i$ denotes the vector of the mean and $\bar{\bar{\Sigma}}_i$ the (co-)variance matrix of distribution 'i' in $\Re^3$:

$$\bar{\bar{\Sigma}}_i = \begin{bmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{xz} \\ \sigma_{yx} & \sigma_{yy} & \sigma_{yz} \\ \sigma_{zx} & \sigma_{zy} & \sigma_{zz} \end{bmatrix}, \qquad (4)$$

where σ_{pq} is the variance (p=q) or the covariance ($p \neq q$) term of the cluster.

From Equation (1) the log-likelihood function Ln L(Φ) can be derived [3]. By setting the partial derivatives of this function to zero, the parameter values which maximize this function can be calculated:

$$\overline{\overline{\nabla}}_{(\alpha_i,\bar{\mu}_i,\overline{\overline{\Sigma}}_i)} L(\Phi) = 0 \qquad (5)$$

Solving this equation results in the following iteration schemes for α_i, $\bar{\mu}_i$, and $\overline{\overline{\Sigma}}_i$:

$$\alpha_i^{(r+1)} = f(\alpha_i^{(r)}, \bar{\mu}_i^{(r)}, \overline{\overline{\Sigma}}_i^{(r)}) \qquad (6a)$$

$$\bar{\mu}_i^{(r+1)} = f(\alpha_i^{(r)}, \bar{\mu}_i^{(r)}, \overline{\overline{\Sigma}}_i^{(r)}) \qquad (6b)$$

$$\overline{\overline{\Sigma}}_i^{(r+1)} = f(\alpha_i^{(r)}, \bar{\mu}_i^{(r+1)}, \overline{\overline{\Sigma}}_i^{(r)}) \; . \qquad (6c)$$

For the iteration we use the integration areas as estimated by the clustering process. However, the iteration schemes imply that the data decomposition in two parameter planes, as was used in the filtering and clustering process, can no longer be applied. A different strategy can be applied to parallelize this problem, for the following reasons:

It is not expected that a lot of distributions will be present in a mixture [3]. Therefore, it is feasible to devote a processor to the iteration of the parameters for one distribution. Furthermore; by making $\overline{\overline{\Sigma}}_i^{(r+1)}$ dependent on $\bar{\mu}_i^{(r)}$ instead of $\bar{\mu}_i^{(r+1)}$ in equation (6c) we explicitly impose parallelism on the calculation of the parameters so that within each of the optimization processes the calculation of the three parameters can be farmed in three processes. This slightly reduces speed of convergence, but this is counterbalanced by the amount of parallelism obtained.

We have observed that this function converges in about 20 iteration steps where the function obtained self-stabilization for the relative fraction (α_i), the vector of the means ($\bar{\mu}_i$) and the elements of the (co-)variance matrix ($\overline{\overline{\Sigma}}_i$).

4. USER-INTERFACE

The user-interface for the complete system is built around the Hypercard programming environment [14]. The high-level scripting language of Hypercard allows an end-user to adapt and control the feel-and-look of the user interface and the operation of the data processing pipeline.

Several extensions have been made to Hypercard in the form of external commands and functions (XCMD/XFCN) [15] so that Hypercard programs can efficiently communicate with the data analyses system via the local transputer. Functions have been written to enable the analysis system to store data of an experiment on the Macintosh filesystem for further analysis. As it is important that data can be visualized on the user-interface in real-time, further extensions have been made to circumvent the inefficient drawing capabilities offered in Hypercard. This is obtained through the use of fast drawing routines. Several routines have been implemented to display projections of the data acquired from an experiment and others that display how the fitted parameters can be matched to this data.

5. CONCLUSION AND FURTHER RESEARCH

In this paper we describe a parallel system for the acquisition, analyses and presentation of multivariate data sets. We have explained how the system is composed of software modules that support the construction of a data processing pipeline. Further research within our group pursues alternative clustering methods using simulated annealing techniques, back-propagating feed-forward neural networks, k-means and ISOdata algorithms. The system will be enhanced by flow-measurement hardware and more effort will be put in the auto-focussing algorithms.

The complete system is currently evaluated at the department of Immunology of the Netherlands Cancer Institute in Amsterdam the Netherlands where it is used for guiding the separation of human peripheral blood cells from inhomogeneous cell populations.

We gratefully acknowledge the assistance of H.A. van Eerde in designing an efficient Parallel Fourier Deconvolution.

6. REFERENCES

1. SLOOT, P.M.A., CARELS, M.J., TENSEN, P. and FIGDOR, C.G. - Computer Assisted Centrifugal Elutriation Part I: Detection System and Data Acquisition Equipment, *Comp. Meth. Prog. Biomed.*, 24, 179 (1987).
2. SLOOT, P.M.A., TENSEN, P. and FIGDOR, C.G. - Spectral Analysis of Flow Cytometric Data: Design of a Special Purpose Low-Pass Digital Filter, *Cytometry*, 8, 545 (1987).
3. SLOOT, P.M.A., VAN DER DONK, E.H.M. and FIGDOR, C.G. - Computer Assisted Centrifugal Elutriation Part II: Multiparametric Statistical Analysis, *Comp. Meth. Prog. Biomed.*, 27, 37 (1987).
4. SLOOT, P.M.A. and FIGDOR, C.G. - Ternary representation of trivariate data, *Cytometry*, 10, 77 (1989).
5. USA Patent 4939081 (1990).
6. Inmos Ltd. - *occam 2 Reference Manual*. Prentice Hall International, UK (1988).
7. Inmos Ltd. - *occam 2 Toolset User Manual*. (1989).
8. MATTOS, P. - Program design for concurrent systems. Inmos Ltd., *Technical Note* 5 (1987).
9. BRACEWELL, R.N. - note on Hermitian data: private communication.
10. SLOOT, P.M.A. - Parallel Fourier Deconvolution on a Transputer Platform. In preparation.
11. GORDON, A.D. - *Classification*. Chapman and Hall, NY (1981).
12. ANDERBERG, M.R. - *Cluster Analysis for Applications*. Academic Press, NY (1973).
13. EVERITT, B. - *Cluster Analysis (2nd edition)*. Heinemann Educational Books, London, UK (1980).
14. Apple Computer Inc., *Hypercard Reference*. (1990)
15. BOND, G. - *XCMD's for Hypercard*. MIS press Inc. Portland, Oregon (1988).