NASA/CR-97-206285

# Digital PIV (DPIV) Software Analysis System

*James L. Blackshire*
*ViGYAN, Inc., Hampton, Virginia*

December 1997

NASA/CR-97-206285

# Digital PIV (DPIV) Software Analysis System

*James L. Blackshire*
*ViGYAN, Inc., Hampton, Virginia*

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23681-2199

Prepared for Langley Research Center
under Contract NAS1-19505

December 1997

# Digital PIV (DPIV) Software Analysis System

## Abstract

A software package was developed to provide a Digital PIV (DPIV) capability for RTG/FMAD/MSTB at NASA LaRC. The system provides an automated image capture, test correlation, and autocorrelation analysis capability for the Kodak Megaplus 1.4 digital camera system for PIV measurements. The package includes three separate programs that, when used together with the PIV data validation algorithm, constitute a complete DPIV analysis capability. The programs are run on an IBM PC/AT host computer running under either Microsoft Windows v3.1 or Windows 95 using a 'quickwin' format that allows simple user interface and output capabilities to the windows environment.

## Acknowledgments

The author wishes to thank several people for assistance and input to the software development. Foremost of all are Mr. William Humphreys and Mr. Scott Bartram, who provided technical support and guidance with PIV theory and practice. Their patience and helpfulness are greatly appreciated. The author also wishes to acknowledge Dr. C.S. Yao, Mr. Keith Pascal, and Dr. Mark Wernet for helpful insight into development of the autocorrelation code development. Finally, I gratefully acknowledge the support provided by the Measurement Science and Technology Branch at NASA Langley Research Center, Hampton, VA 23681, under contract No. NAS1-19505.

# Contents

# Digital PIV (DPIV) Software Analysis System

## 1.0 Introduction

The Digital PIV (DPIV) image capture, test correlation, and autocorrelation analysis algorithms provide the capability for using the Kodak Megaplus 1.4 digital camera system for PIV measurements. The algorithms include three separate programs that, when used with the PIV data validation algorithm, constitute a complete DPIV analysis capability. The programs described in this document include 1) the digital image capture program DPIV.exe, 2) the DPIV test correlation program COR.EXE, and 3) the DPIV analysis program ANALYZE.EXE. The data validation program VALIDATE.EXE is described in NASA Contractor Report 201701. Each of the three algorithms are completely automated with the exception of user input to a configuration file prior to analysis execution for update of various system parameters. The programs are run on an IBM PC/AT host computer running under Microsoft Windows v3.1 and Windows 95 using a 'quickwin' format that allows simple user interface and output capabilities to the windows environment.

## 2.0 Hardware and Software Interface Description

The hardware used in the DPIV PC/AT system include two basic subsystems. They include 1) the Kodak Megaplus v1.4 digital camera and Epix frame grabber 4MEG video subsystem for transfer of the digital image data to PC memory, and 2) the Alacron FT200-AT array processor subsystem for calculating the 128x128 two-dimensional autocorrelations.

Three hardware libraries are linked in for control of the Epix framegrabber and Alacron array processor systems; two for the Epix system and one for the Alacron, respectively. The Epix libraries include m4obm7wl.lib and pxipm7wl.lib. The Alacron links in w3slv860.lib which actually points to the dll -> w3slv860.dll. In addition, the Alacron array processor system is run in slave processor mode which requires code to be run on the Alacron board to be compiled and linked on that system using the Portland Group Compiler and Linker supplied by Alacron.

The hardware and software versions used in the software development for each subsystem include:

| | | **DATE** |
|---|---|---|
| 1) | Kodak Megaplus Digital Camera System v1.4 | 1992 |

|  |  |  |
|---|---|---|
|  | Epix Framegrabber Object Library m4obm7wl.lib v1.7 | 1992 |
|  | Epix Framegrabber Image Processing Lbrary pxipm7wl.lib v1.7 | 1992 |
| 2) | Alacron FT200-AT array processor board | 1995 |
|  | Alacron FT200-AT RT860 Slave Mode Library w3slv860.lib v1.7 | 1995 |
|  | Portland Group Compiler package v2.3 | 1995 |

## 3.0 Software Description

### 3.1 <u>General Software Description</u>

The software is broken down into three separate programs with different functionality. The first program is called DPIV.EXE and provides the capability for capturing multiple high-resolution digital images with the Kodak Megaplus camera system. Digital images are captured at 1320x1035 pixel resolution, 8-bit gray-levels, and stored away in raw binary format to file. The user provides three pieces of information in the configuration file DPIV.CFG, including 1) the Kodak Megaplus Camera exposure level in milliseconds, 2) the number of files to save away, and 3) the base filename for the output files. The files are stored away with incrementing file extensions from *.000 to *.MAX, where MAX is the number of files the user wanted stored away. The files store away at a speed of about 7 seconds per file. The EPIX framegrabber 4mip.exe program is typically used just prior to execution of the DPIV.EXE program to determine the appropriate camera exposure level, and to evaluate image field-of-view, flair, and seed levels present. Once the configuration is setup and saved away, the user runs the DPIV.EXE program in an automated fashion.

The second DPIV program is called COR.EXE. This program provides the capability to test correlate the DPIV binary images just captured. It performs a 2D FFT autocorrelation process on the digital image field in 128x128 pixel subregions of the full 1320x1035 image field. The program displays the 128x128 image subregion on an RS-170 monitor, performs the spatial autocorrelation using the Alacron array processor system, and displays the autocorrelation image field results on the monitor for that subregion. The user provides various parameters to the program through the configuration file COR.CFG. This information includes two types including 1) correlation enhancement parameters, and 2) image file information. The correlation enhancement parameters include such things as image threshold cutoff levels and search box limits. The file information includes the input filename and the number of test correlations to perform. The maximum number of correlations to perform is 285. This number is the result of the 128x128 subimage correlation size and the 50% oversampling that is applied to the analysis. The oversampling shifts the correlation region 64 pixels for each adjacent image region horizontally and vertically. This results in a 19x15 correlation vector grid available for analysis or 285 possible correlations. The user can choose from between 1 and 285 test correlations, of which the system begins correlating in the top left corner of the field-of-view, and continues correlating adjacent image regions horizontally and then down vertically until the MAX number of test correlations provided by the user is reached. The primary use of this program is to optimize the correlation patterns and Signal-to-Noise Ratios of the correlated field by adjusting the correlation parameters. This is a trial and error process and is used primarily to setup the analysis parameter set for the third program ANALYZE.EXE.

The third program, ANALYZE.EXE, is the main analysis program for the DPIV system. It is an automated program that performs the 2d FFT autocorrelation analysis of subregions of the captured DPIV binary images, as described in the COR.EXE procedures above. The user again provides two basic pieces of information in the configuration file ANALYZE.CFG, including 1) correlation enhancement parameters, and 2) image file information. The correlation enhancement parameters should be identical to the optimized parameters determined in the test correlation process described above. The file information needed by the program includes three things including 1) the base filename for the input DPIV files, 2) the output analyzed vector base filename, and 3) the number of incremented files to analyze. The program outputs the top three correlations (and the relative intensity of each) for each of the available 285 image field positions in ASCII text format. This again is determined by the 128x128 pixel subregion sizes and the 50% oversampling, which produces a 19x15 spatially resolved vector grid as the output for each file. The process takes about 1 minute per input image file. The program reads in the configuration file information, and automatically analyzes the number of files input by the user, incrementing from file extension *.000 to *.MAX, as described above.

Each of the three software packages is modular in nature and applies the various hardware and software control functions in a sequential manner. The order of program execution for the DPIV.EXE program typically involves the following steps:

1) system initialization
2) read in configuration file parameters
3) capture high-res DPIV images and write to files
4) free up memory and close hardware

The order of program execution for the COR.EXE program typically involves the following steps:

1) system initialization
2) read in configuration file parameters
3) initialize the i860 array processor and load i860 program
4) calculate the FFT weight functions on the i860
5) initialize the EPIX framegrabber and Kodak Megaplus systems
6) send analysis parameters to i860
7) Begin main analysis loop which involves:
   a) capture subimage using framegrabber
   b) transfer position information to i860
   c) transfer image data to i860
   d) calculate the 2-d autocorrelation of the subimages which involves:
      I.) compute and apply active threshold on raw image data
      II.) check for totally white or black image field
      III.) apply default peak values
      IV.) calculate forward 2d fft
      V.) autocorrelate the 2d fft field
      VI.) center the final correlation field
      VII.) calculate the inverse 2d fft
      VIII.) zero out DC peak region
      IX.) scale output correlation field to 0-255
      X.) apply search box thresholding
      XI.) send correlation image field back to main program
   f) retrieve the final correlation image and display on monitor
10) free up memory and close hardware

The order of program execution for the ANALYZE.EXE program is similar to the COR.EXE program and typically involves the following steps:

1) system initialization
2) read in configuration file parameters
3) initialize the i860 array processor and load i860 program
4) calculate the FFT weight functions on the i860
5) initialize the EPIX framegrabber and Kodak Megaplus systems

6) send analysis parameters to i860
7) Begin main analysis loop which involves:
    a) capture subimage using framegrabber
    b) transfer position information to i860
    c) transfer image data to i860
    d) calculate the 2-d autocorrelation of the 285 subimages which involves:
        I.) compute and apply active threshold on raw image data
        II.) check for totally white or black image field
        III.) apply default peak values
        IV.) calculate forward 2d fft
        V.) autocorrelate the 2d fft field
        VI.) center the final correlation field
        VII.) calculate the inverse 2d fft
        VIII.) zero out DC peak region
        IX.) scale output correlation field to 0-255
        X.) apply search box thresholding
        XI.) calculate the top three centroid positions
        XII.) write the results to output file
    f) retrieve the final correlation number 285 and display on monitor
10) free up memory and close hardware

## 3.2 <u>Detailed Software Description</u>

The DPIV.EXE program is made up of two ‘c’ source files, and three resource files. The program is of a Windows ‘quickwin’ type which allows it to be run in Windows 3.1 or Windows 95, but allows minimal user interaction. Interaction and control of various program execution parameters is provided by a configuration file as described in the previous sections. The user edits the configuration file before program execution, and when the program is run, the configuration parameters are read in from that file. The program then displays the parameters on the monitor in a quickwin window and begins the analysis with no further user input.

The source file DPIV.C is the main program, and provides the image capture and storage capability. It’s contents will be detailed in the following section. The source file DPIVw.C develops and displays the quickwin window and will not be detailed here. The user is referred to the commented listing provided. The three additional resource files will also not be detailed here.

The source file DPIV.C contains four modular functions that perform the various initialization and control operations for the system. The main() function begins and ends program execution and performs memory allocation, initialization, and control functionality. Its first task is to call the read_config() function which reads the configuration file parameters into global variables for use later in the program. The main() function then continues on and prints the configuration parameters in the quickwin windows, followed by a single data array memory allocation call. The function do_open() is then called to open and initialize the EPIX framegrabber system. This function calls two EPIX framegrabber hardware control functions: pxd_m4open() and pxd_chkfault(). The function pxd_m4open() reads in a configuration file called 'videk' which contains various Kodak Megaplus camera configuration parameters. The function also initializes the framegrabber board, and calls pxd_chkfault() if there is a problem. If there is a problem at this point, the function do_open() returns an error message, allowing the user an opportunity to exit the program. If all is OK, the program continues on and captures a test image using the EPIX functions pxd_videkset(), pxd_videkdo(), and pxd_video(). These functions setup the Kodak Megaplus camera for high resolution capture mode, set the capture type and exposure level, and setup the display, respectively. The program then enters a looping sequence which involves:

1) capturing a high resolution Kodak Megaplus image,
2) setup and open the output filename for binary write using fopen(),
3) read the image data from the EPIX framegrabber buffer using pxd_iopen() and pxd_io() a line at a time, and
4) write the binary image field a line a time out to file using the function fwrite().

Once all of the data has been read and written to file, the memory is freed up and the hardware is closed.

The COR.EXE and ANALYZE.EXE programs are similar in nature and will be described here together. Detailed listings of the source code files for the programs are presented at the end of this document. Each program is made up of three 'c' source files, and three resource files. The program is of a Windows 'quickwin' type which allows it to be run in Windows 3.1 or Windows 95, but allows minimal user interaction. Interaction and control of various program execution parameters is provided by a configuration file as described in the previous sections. The user edits the configuration file before program execution, and when the program is run, the configuration parameters are read in from that file.  The program then displays the parameters on the monitor in a quickwin window and begins the analysis with no further user input.

The source files COR.C and ANALYZE.C are the primary ‘control’ programs, and the source files CORp.C and ANALYZp.C are ‘slave’ programs run on the i860 array processor.  Each of these will be detailed in the following section.  The source files CORw.C and ANALYZw.C develop and display the quickwin windows and will not be detailed here.  The user is referred to the commented listing provided.  The three additional resource files for each program will also not be detailed here.

The source files COR.C and ANALYZE.C each contain six modular functions. The main() function begins and ends program execution and performs memory allocation, initialization, and control functionality.  Its first task in each program is to call the read_config() function which reads the configuration file parameters into global variables for use later in the program.  The main() function then continues on and prints the configuration parameters in the quickwin windows, followed by data array memory allocation calls.

The Alacron i860 array processor board is then initialized and opened with the functions alopen() and aldev(), respectively.  The i860 slave programs ‘corp’ and ‘analyzp’ are then loaded by each respective program into i860 memory using the function almapload().  Memory is then initialized and allocated using the function i860malloc() and malloc() functions.  A call to the i860 slave  function dowts() is then made using the master function alcall().  This function calculates the FFT weight functions needed for the autocorrelation routine.  The function alwait() prevents further program execution until the alcall() function returns from the i860.

The EPIX 4MEG video framegrabber and Kodak Megaplus systems are then initialized and opened using the do_open() and pxd_m4open() functions.  An initialization image is then captured using the pxd_videkset() and pxd_videkdo() functions.

The correlation analysis parameter set for each program is then sent to the i860 using the functions alsetla() and alsetba() functions.  This includes 9 long integer values and a character string containing the output filename for the COR.EXE program, and 12 long integer values and a character string containing the output filename for the ANALYZE.EXE program.  Each program then calls the Alacron functions alsetla(), which in turn call the i860 slave functions params(), to transfer the correlation parameter sets to i860 memory.  The programs then diverge slightly with the COR.EXE program  calling alcall() and alwait(), in turn, which call the i860 slave function sendparams() which sets up the correlation parameters in i860 memory.  The test correlation program then captures a test image using the EPIX functions pxd_videkdo() and pxd_video().  The

ANALYZE.EXE program in the mean time has called the function onetenhund() to set up the output file extension number for the current file being read in. It has also used various string manipulation functions to assemble the full input and output filenames. Each of the programs then call the function fopen() to set up the input DPIV data file for binary read.

Image data is then read in from file and transferred to PC memory, and then transferred from the PC memory to the i860 memory in a piece-wise manner. This is based on the fact that the EPIX function pxd_io() and Alacron function alsetba() are limited to transferring 32000 bytes of information at a time. This requires the transfer of image data in forty-one pieces 25 rows by 1320 pixels long. The image data read from file and transferred to PC memory is accomplished with the read_array() function, which in turn calls the fread() function for transfer of the binary image data to a temporary PC memory buffer. Transfer from the PC memory to i860 memory is then done with the Alacron alsetba() function. The relative read-in and write-out image locations is sent with each chunk of image data as it is sent to the i860 memory arrays. A call to the i860 slave function senddata() is then required to reassemble the image field to a full 1320x1035 from its forty-one transferred pieces.

Once the image and analysis parameter sets have been sent and reassembled in i860 memory, the two programs diverge slightly again. The COR.EXE program sets up for display of the image subregion using the EPIX function pxd_video() and the function delay(). It then enters a looping scheme to first display the image field, calculate the autocorrelation field, and then display the result. The display of the current image field is done by calling the i860 function doshow() which reads the current 128x128 image and sends it back to PC memory, and the framebuffer for display. This display process also involves calling the functions algetba(), array_to_buf(), and delay(). The main() function then calls the i860 function dofft() to perform the autocorrelation of the subimage, followed by a retrieval of the correlation image field information for display using the functions just described. This looping sequence is repeated for the user-defined number of test correlations.

The ANALYZE.EXE program in the mean time has called the function alsetba() to send the output filename to i860 memory for the analyzed vector data set. This is followed by a call to the i860 slave function goparam() which reassembles the correlation parameter set and output filename in i860 memory. The i860 slave function dofft() is then called to perform the autocorrelation analyses of the entire 285 image subregions of the 1320x1035 image field.

The dofft() autocorrelation process for each program begins with a determination of the weighted mean pixel level for the 128x128 subimage region being evaluated:

$$\text{Weighted Mean} = \frac{\left(\sum_{i=0,j=0}^{i=128,j=128} I_{pixel}\right) \times Weight}{16384}$$

where *I* is the 8-bit pixel intensities (0-255), and*weight* is the user input weighting amount usually set to between 0.65 and 1.25. If a pixel value is found to be greater than the weighted mean value it is set to zero, and if found less than or equal to the weighted mean value it is set to its compliment value. This produces an image field of bright white particle image dots on a black background.

A count of bright and dark levels in the current image field is kept so that if excessive bright or dark areas are present the frame can be zeroed out for easier data validation efforts later. This condition typically occurs in boundary regions, image flair regions, or non-seeded regions. On a similar theme, a default correlation peak set is introduced into the image field to aid in data validation efforts later. This is done by setting a series of six pixels to a intensity level of 20 corresponding to a set of three particle image pairs that will correlate if the entire image field is black. The user elects to introduce these default peaks and chooses their relative locations. The minimal 20 pixel intensity level causes no adverse affects on a legitimate PIV image field, which has particle image peak intensities near 255 and contrast levels in excess of 50:1.

Following the active threshold (weighted mean) implementation, the forward 2-dimensional Fourier transform is applied to the data set using the Alacron provided function cfft2d(). The 2d FFT is then autocorrelated with itself:

$$\Im[R(\alpha,\beta)] = \Im[f^*(x,y) \otimes f(x,y)] = |F(p,q)|^2$$

which basically reduces to squaring the real and imaginary parts of the FFT and adding them together:

$$|F(p,q)|^2 = A_F(p,q)e^{i(px+qy)} \times A_F(p,q)e^{-i(px+qy)} = \{\mathrm{Re}[F(p,q)]\}^2 + \{\mathrm{Im}[F(p,q)]\}^2$$

where F(p,q) is the complex 2d forward FFT of the image field. The correlated image field is then centered in the image field by taking advantage of the translation property of the Fourier transform which sets the origin of the Fourier transform of f(x,y) to the center of its corresponding NxN frequency square by multiplying f(x,y) by $(-1)^{x+y}$. This is coded up by multiplying the x and y frequency components values of f(x) and f(y) by $(-1)^{x}$and $(-1)^{y}$ respectively. Mathematically, this is represented by:

$$f(x,y)(-1)^{x+y} \Leftrightarrow F(u-N/2, v-N/2)$$

The inverse Fourier transform is then obtained to produce the desired PIV autocorrelation pattern, which takes the form of a self-correlation peak centered in the image field, and symmetric satellite correlation peaks about the center of the image field representing the average particle image displacement present in the field.

Following the successful completing of theautocorrelation processing steps, a determination of the locations of the satellite peaks needs to be conducted. This first involves applying a filter to the self correlation peak centered in the image field. The DC peak extent value provided by the user is used to zero-out values extending from the center of the image field out to the peak extent level. This basically zeroes out a square box pattern centered around pixels (64,64). The correlation field is then scaled to 8-bit levels (0-255) for display and storage later. This involves calculating the minimum and maximum pixel intensity values in the image field and applying:

$$Scaled\ Pixel\ Intensity = \frac{Pixel\ Intensity + (-1.0 * Minimum)}{Maximum + (-1.0 * Minimum)} \times 255$$

The user supplied search box values are then used to apply a hard threshold level within the search box boundaries. An additional 0-255 scaling operation is done before the hard threshold to enhance the dynamic range levels within the search box. The hard threshold level is then applied so that if the pixel value is less than the user threshold it is zeroed.

The COR.EXE program at this point simply sends the resulting correlation image field back to the main() PC memory and framegrabber buffer for display, concluding its execution. The ANALYZE.EXE program, however, remains in the i860 slave program for evaluation of the autocorrelation centroid values in the search box region. The custom set of image centroid functions and procedures begin by calling the i860 slave function main_search() from within the dofft() function. A series of three dynamic centroid storage arrays are used to store potential centroid values, of which the top three intensity levels will be saved away. The main_search() function checks the pixel intensity levels from the top left portion of the search box to the bottom right, and calls blob_search() if it finds a pixel above the threshold level. The blob_search() function in turn looks for connected neighbors that are above the threshold level and stores away the (x,y) positions and the intensity value for each pixel associated with a given blob. Once the entire blob extent has been determined, its centroid is evaluated using the i860 slave function find_centroid() which uses a weighted mean average routine to determine the x, y, and intensity levels of the blob. A comparison of its weighted mean intensity level is then made with the top three centroids, and if it beats any of the three, it is placed ahead of them in the final centroid arrays.

Returning to the main_search() function, and following the determination of the top three centroid positions within the search box, the centroid x and y positions are scaled relative to the center of the image field, to give the relative displacement away from the self-correlation peak (which is the average displacement of the particle pairs in the field). The magnification and laser pulse separation are then taken into account using the user defined values to transform the final centroid pixel displacements to velocity according to:

$$Velocity\ Level = \frac{(Pixel\ Displacement\ Level)*(Analysis\ Scale\ Factor)}{(Acuisition\ Magnification\ Level)*(Laser\ Pulse\ Separation)}$$

where the analysis scale factor, acquisition magnification level, and laser pulse separation are determined by, and supplied by the user. The x and y position is also scaled at this point in the program according to:

$$Scaled\ Position_{x,y} = \frac{[Position_{x,y}*(Analysis\ Scale\ Factor)]}{(Acquisition\ Magnification\ Level)} + Shift_{x,y}$$

where the $Shift_{x,y}$ represents the amount of x or y shift between image capture frames.

The top three autocorrelation centroid positions are then written out to file along with the 8-bit (0-255) correlation peak intensity level, and the x and y position. The output file format takes the following form of eleven columns of data across:

| **column 1** | **column2** | **column3** | **column4** | **column5** | **column6** | **column7** | **column8** | **column9** | **column10** | **column11** |
|---|---|---|---|---|---|---|---|---|---|---|
| **x position** | **y position** | **1st u** | **1st v** | **1st peak** | **2nd u** | **2nd v** | **2nd peak** | **3rd u** | **3rd v** | **3rd peak** |

where u and v are the u and v velocity components, respectively, and peak is the relative centroid correlation peak height recorded during the analysis process. The 1st, 2nd, and 3rd designations represent the strongest, 2nd strongest, and 3rd strongest correlation peak heights, respectively.

Following the autocorrelation analysis and data write-out to file, the ANALYZE.EXE program returns to the master program's control and the main() function. Memory is then freed up, hardware/output data files are closed, and program execution is terminated.

## 4.0 Software Use and Procedures

The following section is provided as a guide for using the three DPIV programs. The basic procedure for all three involves first editing the configuration file for each respective program and saving the resulting text file away, and then running the automated executable program which reads in the configuration file for needed information.

## 4.1 DPIV.EXE Use and Procedures

The first program, DPIV.EXE, captures and stores away high resolution Kodak Megaplus digital images. The first step usually involves using the EPIX framegrabber system's 4MIP.EXE program to view the image field and to make any needed adjustments regarding things such as flair, seed density, reflections, focus, particle image shift amount, timing adjustments, and for the purposes of the DPIV.EXE program the Kodak Megaplus camera exposure level that will be used for the capture sequence. The typical exposure level is between 20-1000 microseconds, with most cases falling in the 100-300 microsecond range.

The second step involves setting up the configuration file DPIV.CFG. An example configuration file input is provided below:

```
Software_Version_Number          1.2
Exposure_Level(ms)               100
Output_Datafile_Base_Name        may1sh
Number_of_Files_to_Save          30
```

DPIV digital image capture system configuration file: **DPIV.CFG**

The file supplies three system parameters needed for processing. These include 1) the Kodak Megaplus camera exposure level determined in step #1, 2) the output base filenames the user would like to use for the digital images to be stored, and 3) the number of images to store away.

Once the configuration file is properly setup, it is saved away, and the DPIV.EXE program is run. This process starts the automated digital image capture routine that captures and stores away the DPIV image files. The parameters that were read in from the configuration file are displayed in the 'quickwin' box supplied by the program, and the current file number being saved away is updated as the program executes. The system is currently capable of storing files away at about 7 seconds each.

## 4.2 COR.EXE Use and Procedures

The second program, COR.EXE, allows the user to test the autocorrelation patterns of a captured DPIV image field, and adjust various correlation parameters to enhance the correlation SNR levels.

It first involves setting up the configuration file COR.CFG, which provides three basic types of information to the program. An example configuration file input is provided below:

```
Software_Version_Number                  1.2
Image_Threshold_Scale_Level(0.65-1.25)   0.9
FP_DC_Peak_Extent(1-20)                  10
Search_Box_xmin(0-128)                   55
Search_Box_xmax(0-128)                   85
Search_Box_ymin(0-128)                   20
Search_Box_ymax(0-128)                   45
Default_Peak_xpos(0-64)                  2
Default_Peak_ypos(0-64)                  2
Centroid_Threshold_Level(0-255)          250
Input_DPIV_Filename                      may1sh.005
Number_of_Test_Correlations              15
```

DPIV test correlation program configuration file: **COR.CFG**

The first piece of information that needs to be edited or updated is the input DPIV filename that the user would like to evaluate. (The full filename is required for this program, unlike the DPIV.CFG and ANALYZE.CFG files which use only the file basenames.) The second piece of information required is the number of test correlations the user would like to view. This can range from 1 to 285. The subimage and corresponding test correlations always begin in the upper left corner of the full 1320x1035 high resolution image and increment to the right and then downward in a 19x15 grid. The image fields and correlation fields are displayed as 128x128 image fields on the RS170 monitor one after the other, and from one position to the next. The correlations take about 4-5 seconds to cycle through for each, so that if the user chose 15 test correlations to view, the process would take a little over a minute to execute totally. If all 285 available subimage regions were viewed the process would take approximately 15-20 minutes.

The main utility of this program, however, involves changing the various correlation parameters to enhance the Signal-to-Noise Ratio of the resulting particle correlation patterns. There are nine variables that can be changed and adjusted, including two image thresholding variables, a self-correlation extent variable, two default peak correlation variables, and four correlation search box variables.

The image threshold scaling factor applies a threshold to the raw image before FFT processing based on the average pixel intensity (weighted mean) in the image. The user picks a value typically between 0.9 to 1.05 to scale the raw image field cutoff level and enhance the image contrast level before FFT processing begins. The centroid threshold level limits the centroid search to pixel intensity values above a certain level. A typical correlation peak has a peak intensity level above 240 (out of a possible 255), but in low seed density images, or noisy image fields, the peak intensity levels can be closer to 200. The user should pick a value that allows strong particle correlation patterns to extend out 2-10 pixel units radially from the correlation peak center.

The self-correlation and search box variables, tend to block out or limit extraneous correlation pattern influences that may swamp-out or saturate weak correlation patterns in low SNR instances. The self-correlation peak extent variable zeroes-out a square box region in the center of the correlation image field, effectively blocking out the self-correlation peak position. The user determines, in pixel units, the radial extent of the DC peak correlation pattern by trial and error, and chooses a level that blocks out the entire self correlation pattern. The extent typically ranges from 2-14 pixels out from the center, and is based on particle sizes, magnification levels, aberrations, and other imaging parameters.

The centroid search box extent variables place a limit on the region the centroid algorithm will look for valid image centroid images. The four variables are referenced to the upper left corner (0,0) of the 128x128 pixel field, where the center of the image field is (64,64). If the user picks (0,0), (0,128), (128,0) and (128,128) for the four search box variables, respectively, the entire FFT image field will be available for the centroid evaluation process. A typical search box region, however, extends only about 20-30 pixels in each direction. Both the DC peak extent and search box extents are displayed in the FFT correlation image field for evaluation purposes. The user should be liberal with the extents of the search box, and should test several different image fields and positions in the flow, to allow for a wide dynamic range of correlation peak positions.

The final variable set involves the insertion of default peak positions, in the event that there is a totally white, or totally black image field. The default peaks are usually chosen to be just inside one of the corners of the search box region, so that they correlate and are detected by the centroiding algorithm. The default peak values show up as known correlation values in the final data set and help in the data validation processing steps. The user can decide to not use the default peak values by choosing values that correlation out of the search box region, or by choosing (0,0).

The use of the COR.EXE program is typically an iterative process, involving the adjusting of a variable in the configuration file, and seeing how it affects the correlation pattern. The configuration file is edited each time, saved away, and then the COR.EXE program is run to view the correlation patterns. This procedure continues until the user is happy with the correlation SNR levels for a given image file, and then for several image files in a given analysis set.

## 4.3 ANALYZE.EXE Use and Procedures

The final DPIV program, ANALYZE.EXE, automatically analyses multiple DPIV image files based on the input of its configuration file ANALYZE.CFG. An example configuration file input is provided below:

```
Software_Version_Number                     1.2
Image_Threshold_Scale_Level(0.65-1.25)      0.8
FP_DC_Peak_Extent(1-20)                     14
Search_Box_xmin(0-128)                      50
Search_Box_xmax(0-128)                      80
Search_Box_ymin(0-128)                      20
Search_Box_ymax(0-128)                      45
Default_Peak_xpos(0-64)                     50
Default_Peak_ypos(0-64)                     50
Centroid_Threshold_Level(0-255)             240
Pixel_Scale_Factor(microns/pixel)           5.0
Laser_Pulse_Separation(microsec)            10.0
Input_DPIV_Binary_Base_Filename(s)          try1
Output_Vector_Base_Filename(s)              atry1
Number_of_Files_to_Analyze                  2
```

DPIV Analysis program configuration file: **ANALYZE.CFG**

Four types of information are input to the file including: 1) the input and output DPIV raw and analyzed file basenames, 2) the number of files to be analyzed, 3) the system magnification and laser pulse separation times, and 4) the correlation parameter set determined by the COR.EXE program described above.

The first piece of information that needs to be edited or updated is the input DPIV file basename(s) that the user would like to analyze, and what the output analyzed file basename(s) should be. The second piece of information required is the number of files to analyze. The analysis process takes about 1 minute per file, so a typical set of 30 file would take ½ hour to analyze.

The third piece of information needing input involves the system pixel scaling factor (magnification level), and the laser pulse separation timing. These values were probably determined before the data was taken, and scale the output vector position, and vector length data to the appropriate levels.

The final type of information that needs to be edited in the configuration file involves updating the correlation parameter set that was determined previously with the COR.EXE program. These variables include the threshold, search box, DC peak extent, and default correlation values found in that process to optimize the SNR of the correlation pattern.

Once the configuration file has been updated and saved away, the ANALYZE.EXE program is run. This process starts the automated digital PIV analysis routine that reads in DPIV image fields, autocorrelates the image, and stores away the output vector field to the user supplied output file names. The parameters that were read in from the configuration file are displayed in the 'quickwin' box supplied by the program, and the current file number being analyzed is updated as the program executes. The approximate time left in the processing is also displayed on the screen.

## 5.0 <u>Software Development Issues</u>

The three Digital PIV software packages were written in a Microsoft 'quickwin' format, which allows the program to be run in Windows 3.1 or Windows 95, but which doesn't allow any user interaction with the program while it is running. Each of the packages has a number of files needed for successful compiling and linking. The DPIV.EXE program requires the following five files:

| | | | |
|---|---|---|---|
| 1) | DPIV.c | - | The program's main control source file |
| 2) | DPIVw.c | - | The quickwin source file |

|  |  |  |  |
|---|---|---|---|
| 3) | DPIV.def | - | The definition file for the program |
| 4) | DPIV.rc | - | The resource file for the program |
| 5) | DPIV.ico | - | The bitmap icon for the program |

In addition, a command line batch file has been developed to simplify the compile, link, and resource compiler stages of the build process. This files name is DPIV.bat. The program was compiled with Microsoft C compiler version v8.00, linker version v5.5, and resource compiler v3.11.

Editing of the code to add new functionality, or to change existing capabilities is relatively straight forward. The modular nature of the code allows this. The quickwin source file DPIVw.c, definition file DPIV.def, resource file DPIV.rc, and icon file DPIV.ico should not require any changes and should be left alone if possible.

A copy of the compile/link batch file listing is provided below:

```
cl -c -AL -Gsw -Zpe -DC_MSC -Dmain=Cmain dpiv.c dpivw.c
link /NOD /NOE dpiv+dpivw,,,libw Llibcew m4obM7WL pxipM7WL,dpiv.def
rc -K dpiv.rc dpiv.exe
```

It calls the Microsoft compiler, linker, and resource compiler, with various switches setup for a quickwin application.

The COR.EXE program requires the following six files:

|  |  |  |  |
|---|---|---|---|
| 1) | COR.c | - | The program’s ‘master’ control source file |
| 2) | CORp.c | - | The program’s i860 array processor ‘slave’ source file |
| 3) | CORw.c | - | The quickwin source file |
| 4) | COR.def | - | The definition file for the program |
| 5) | COR.rc | - | The resource file for the program |
| 6) | COR.ico | - | The bitmap icon for the program |

The command line batch file for this program is called COR.bat. The program was also compiled with Microsoft C compiler version v8.00, linker version v5.5, and resource compiler v3.11.

Editing of the code to add new functionality, or to change existing capabilities should again be relatively straight forward due to the modular nature of the code. But again, the quickwin source file CORw.c, definition file COR.def, resource file COR.rc, and icon file COR.ico should not require any changes and should be left alone if possible.

A copy of the compile/link batch file listing for COR.EXE is provided below:

```
pgcc -Mreentrant -o corp corp.c i860lib.a pgcnl.a libm.a
cl -c -AL -Gsw -Zpe -DC_MSC -Dmain=Cmain cor.c corw.c
link /NOD /NOE cor+corw,,,w3slv860 libw Llibcew m4obM7WL pxipM7WL,cor.def
rc -K cor.rc cor.exe
```

It calls the Portland Group Compiler for compiling on the i860 processor board, followed by the Microsoft compiler, linker, and resource compiler, with various switches setup for a quickwin application.

The ANALYZE.EXE program requires the following six files:

|  |  |  |  |
|---|---|---|---|
| 1) | ANALYZE.c | - | The program's 'master' control source file |
| 2) | ANALYZp.c | - | The program's i860 array processor 'slave' source file |
| 3) | ANALYZw.c | - | The quickwin source file |
| 4) | ANALZYE.def | - | The definition file for the program |
| 5) | ANALYZE.rc | - | The resource file for the program |
| 6) | ANALYZE.ico | - | The bitmap icon for the program |

The command line batch file for this program is called ANALYZE.bat. The program was compiled with Microsoft C compiler version v8.00, linker version v5.5, and resource compiler v3.11.

Editing of the code to add new functionality, or to change existing capabilities is once again a relatively straight forward. The quickwin source file ANALYZEw.c, definition file ANALZYE.def, resource file ANALYZE.rc, and icon file ANALZYE.ico should, however, not require any changes and should be left alone if possible.

A copy of the compile/link batch file listing is provided below:

```
pgcc -Mreentrant -o analyzp analyzp.c i860lib.a pgcnl.a libm.a
cl -c -AL -Gsw -Zpe -DC_MSC -Dmain=Cmain analyze.c analyzw.c
link /NOD /NOE analyze+analyzw,,,w3slv860 libw Llibcew m4obM7WL pxipM7WL,analyze.def
rc -K analyze.rc analyze.exe
```

It also calls the Portland Group Compiler for compiling on the i860 processor board, and then the Microsoft compiler, linker, and resource compiler, with various switches setup for a quickwin application.

This software package was developed under contract to NASA. Requests for copies of the source and executable codes described in this report should be directed in writing to one of the following:

NASA Langley Research Center  
Technology Applications Group  
Mail Stop 118  
Hampton, Virginia 23681-0001

or

Vigyan, Inc.  
Aeronautical Research Group  
30 Research Drive  
Hampton, Virginia 23666-1325

| REPORT DOCUMENTATION PAGE | Form Approved OMB No. 0704-0188 |
|---|---|

| | | |
|---|---|---|
| **1. AGENCY USE ONLY** (Leave blank) | **2. REPORT DATE** December 1997 | **3. REPORT TYPE AND DATES COVERED** Contractor Report |

| | |
|---|---|
| **4. TITLE AND SUBTITLE** Digital PIV (DPIV) Software Analysis System | **5. FUNDING NUMBERS** C NAS1-19505 WU 538-03-12-04 |
| **6. AUTHOR(S)** James L. Blackshire | |
| **7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)** Vigyan, Inc. 30 Research Drive Hampton, Virginia 23666 | **8. PERFORMING ORGANIZATION REPORT NUMBER** |
| **9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)** National Aeronautics and Space Administration Langley Research Center Hampton, VA 23681-2199 | **10. SPONSORING/MONITORING AGENCY REPORT NUMBER** NASA/CR-97-206285 |

**11. SUPPLEMENTARY NOTES**
Langley Technical Monitor: Dr. Richard R. Antcliff
Final Report

| | |
|---|---|
| **12a. DISTRIBUTION/AVAILABILITY STATEMENT** Unclassified-Unlimited Subject Category 36 Distribution: Nonstandard Availability: NASA CASI (301) 621-0390 | **12b. DISTRIBUTION CODE** |

**13. ABSTRACT** (Maximum 200 words)

A software package was developed to provide a Digital PIV (DPIV) capability for NASA LaRC. The system provides an automated image capture, test correlation, and autocorrelation analysis capability for the Kodak Megaplus 1.4 digital camera system for PIV measurements. The package includes three separate programs that, when used together with the PIV data validation algorithm, constitutes a complete DPIV analysis capability. The programs are run on an IBM PC/AT host computer running either Microsoft Windows 3.1 or Windows 95 using a 'quickwin' format that allows simple user interface and output capabilities to the windows environment.

| | |
|---|---|
| **14. SUBJECT TERMS** Digital PIV, DPIV, Particle Image Velocimetry | **15. NUMBER OF PAGES** 27 |
| | **16. PRICE CODE** A03 |

| | | | |
|---|---|---|---|
| **17. SECURITY CLASSIFICATION OF REPORT** Unclassified | **18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified | **19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified | **20. LIMITATION OF ABSTRACT** |