Kenyon Copy

DIGITAL SIGNAL PROCESSING TECHNIQUES FOR THE MEASUREMENT OF OCULAR COUNRTERROLLING

by

Yoshihiro Nagashima

Kogakushi, Tokyo Institute of Technology (1977)

Kogakushushi, Tokyo Institute of Technology (1979)

Submitted in Partial Fulfillment of the

Requirements for the Degree of

MASTER OF SCIENCE

IN

AERONAUTICS AND ASTRONAUTICS

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 1985

Signature of Author ______________________________

Department of Aeronautics and Astronautics, May 1985

Certified by ______________________________

Robert V. Kenyon, Ph.D., Thesis Supervisor , Assist.Prof.

Accepted by ______________________________

Professor Harold Y. Wachman, Chairman Dept. Grad. Committee

DIGITAL SIGNAL PROCESSING TECHNIQUES FOR THE MEASUREMENT OF OCULAR COUNTERROLLING

by

Yoshihiro Nagashima

Submitted to the Department of Aeronautics and Astronautics
on May 10, 1985 in partial fulfillment of the requirement
for the Degree of Master of Science in Aeronautics and Astronautics

ABSTRACT

Measurement of ocular rolling is valuable for understanding space sickness because the torsion of the eyes response to the gravity vector change or visual scenery rotation is one of the few indications of otolith function.

There are a number of methods for measuring ocular torsion. But each of them has some drawbacks. Several digital signal processing methods have been investigated and it is known that the subpicture method is the most accurate calculation method of ocular torsion. This thesis was undertaken to develop practical softwares using this subpicture method for analyzing image rotation.

In the subpicture method, two 64*64 byte dimension subpictures are taken from a 256*256 byte dimension reference picture. A region of interest which has 20*20 byte dimension is defined at the center of the subpictures.

This region of interest is crosscorrelated with another subpicture taken from a data image picture. The correlation peak gives calculation data for the image rotation.

Two practical softwares were developed for RT-11 System at the Man-Vehicle Laboratory and VAX System at Tufts University Image Analysis Laboratory.

Supervisor: Robert V. Kenyon, PhD
Title: Assistant Professor in the Department of Aeronautics and Astronautics

ACKNOWLEDGEMENTS

I wish to express my appreciation to Robert V. Kenyon, Assistant Professor of the Department of Aeronautics and Astronautics, Laurence R. Young, Director of the Man-Vehicle Laboratory and Associate Professor of the Department of Aeronautics and Astronautics, and Professor Harold Y. Wachman, Chairman Departmental Graduate Committee, with special acknowledgement to my thesis supervisor, Robert Kenyon. I would like to thank for the great support which was provided by David Zahniser of Tufts University Image Analysis Laboratory.

I would like to acknowledge the frequent assistance by Sherry Modestino, Bob Renshaw, Mark Shelhamer, Remy Malan, Bryan Sullivan, Marilyn Williams and the other members of the Man-Vehicle Laboratory.

This thesis is dedicated for the promotion of friendship between the United States of America and my home country, Japan, and to my parents who have always believed in me.

Table of Contents

1. Introduction

Space sickness has been a problem for Space Shuttle pilots and payload specialists when they conduct experiments for many days on the earth orbit in weightlessness or micro-gravity. Space sickness or motion sickness is a peculiar function of the vestibular system.

Figure 1.1 and 1.2 show the human vestibular system. The labyrinthine-vestibular sensory system is the main apparatus for the maintenance of equilibrium and awareness of the body's position in relation to its environment. It serves to transduce forces associated with linear and angular movements of the head into nerve impulses that reflexly control movement and posture. Five sensory organs are included: the two saccular and utricular macules sense gravito-inertial forces; the three semicircular canals sense angular accelerations of the head. Each macule supports otoliths, the position of which is altered by linear motion and by the effects of gravity. The cristae (the sensory receptors of the semicircular canals) are attached to their walls and are activated by motions of the fluid within the canals. Currents can also be induced in this fluid by warming and cooling the tympanic membrane. The otoliths and cristae deform hair cells of the sensory membranes, which induce nerve impulses. The latter are transmitted by ganglion cells (located in the internal auditory canal) and reach the brainstem via the vestibular nerve.[5]

The main reason for space sickness is the mismatch between visual information from eyes and gravitational information from the vestibular system. For instance, if you tilt your head around the line of sight on the earth in 1G gravitational field, the eye balls also tilt and send rotated visual information to the cerebrum. But, at the same time, the vestibular system also feels the gravity vector change by the tilting of the head, and sends the gravitational information to the cerebrum. This visual and gravitational information match with each other on the earth in 1G field, but do not match each other on the earth orbit in 0G field, due to the vestibular system not feeling the gravity change by the head tilt in 0G field. This interaction between visual system and vestibular system causes the space sickness.

An interesting relation of the vestibular system to the eye movement is ocular counterrolling or ocular torsion. If you tilt your head around the line of sight, the eye does not rotate as much as the head. Rotating visual information, such as a rotating dome in front of the eye, can cause the ocular torsion even if there is no gravity vector change.

The analysis of this ocular counterrolling is important for knowing the function of the vestibular system and for the analysis of space sickness.

There are a number of methods of measuring ocular torsion, however, each of the methods has some drawbacks. The most accurate methods are those that require attachment of a device such as a contact lens directly to the eye. But these methods are not applicable for some experiments.[1] The non invasive video photographic methods generally require human analysis of images of the eye. This human photographic interpretation is costly and requires meticulous attention to quality control.

One of the latest developments in this area of counterrolling measurement has been that of a video-based monitoring system for torsional eye movement. Many digital image processing applications require that two images be registered with each other. The most commonly investigated sub-task in image registration is detection of translation, however, techniques for measuring rotation have also been examined. One application where there has been extensive use of image registration is the evaluation of LANDSAT imagery.[1]

Mehdi Hatamian et al (1983) presented a video-based technique for measuring the torsional movement of the eye (counterroll) by processing video images of the eyeball. Entire system was based on the fact that most variance in iris image density occurs in the angular direction in a polar coordinate system centered on the pupil. Crosscorrelation between sequences was obtained by sampling the digitized image of the eye ball at a fixed radius from the pupil center. The iris portion of the image was converted to an N*N format and then transformed using a two dimensional FFT (Radial Sampling). The results by software implementation of the algorithm showed the resulting error was less than 0.1 degree for the eye ball picture.[3]

Anthony Parker (1983) compared the advantages of three methods of radial sampling, sector tracking and landmark tracking and he has concluded that the landmark tracking method is easy to implement and has the sufficient accuracy.[1][2]

The thesis of this work is that digital signal processing techniques can be applied to images of the eye in order to measure ocular torsion and is to develop a practical software. Two practical softwares were developed for the RT-11 System at the MIT Man-Vehicle Laboratory and the VAX System at Tufts University Image Analysis Laboratory.

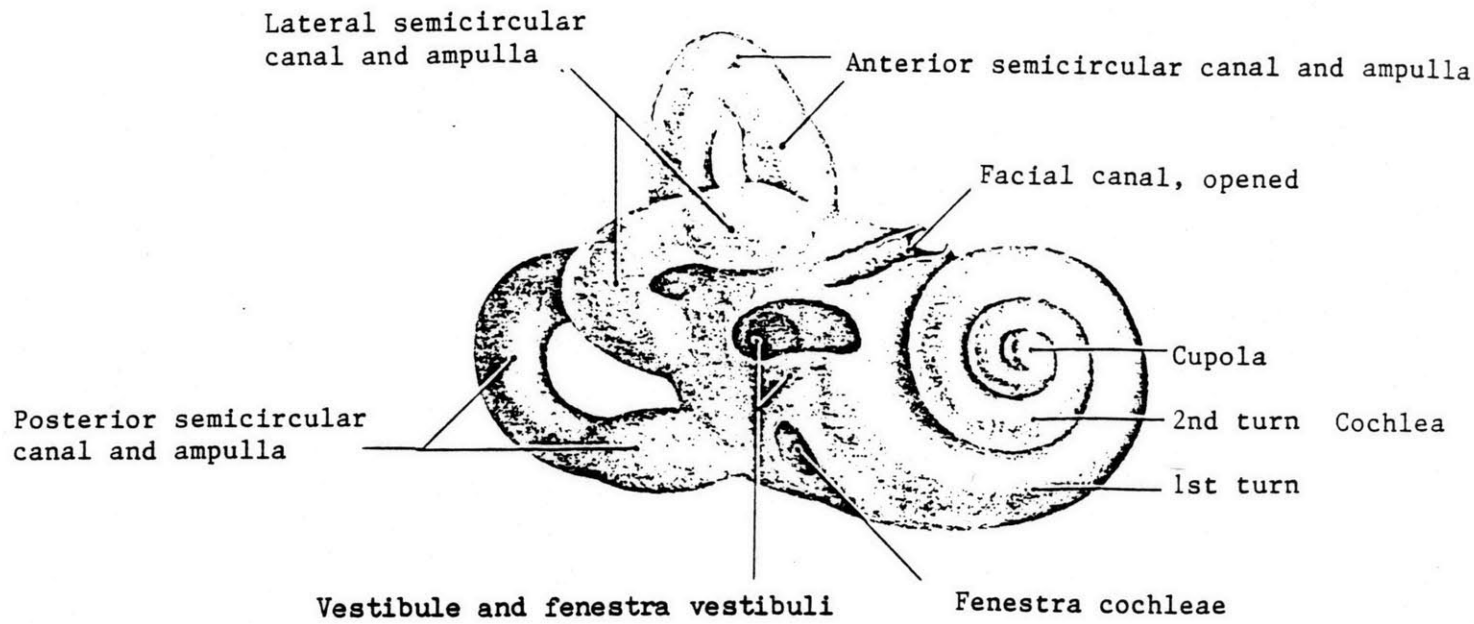


Fig. 1.1 Bony Labyrinth, Lateral View, Right Side [6]

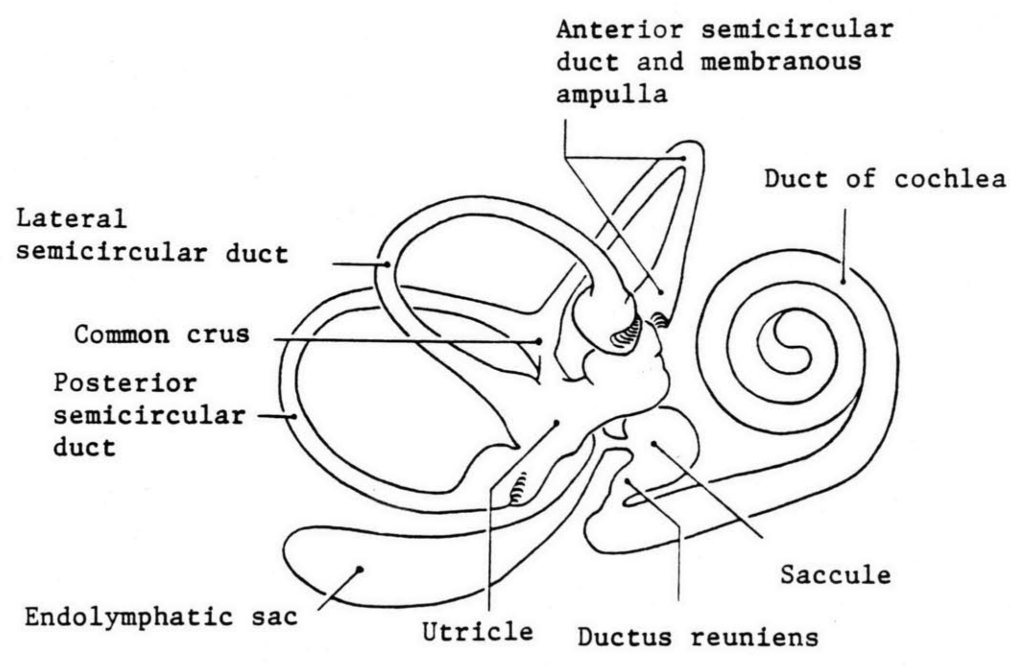


Fig. 1.2 Membranous Labyrinth, Lateral View, Right Side [6]

2. Rotation Measurement Algorithms

There have been several algorithms for the detection of data image rotation; (1) sector tracking, (2) radial sampling of FFT, and (3) subpicture method which was investigated in this thesis.

2.1 Sector Tracking Method - Theory of Action

In the sector tracking method, the reference image and the data image of the eye are radially sampled around the center of the pupil. The horizontal dimension of the sampled new images corresponds to the angular direction of the original images. The vertical dimension of the sampled new images corresponds to the radial direction of the original images.

The sampled new reference image and data image are Fourier transformed and filtered with the Mexican hat filter for boosting the moderately high frequencies. After inverse Fourier transformed, these radially sampled images are crosscorrelated. The translation between these two image matrices in the horizontal direction corresponds to the angular direction and provides a measure of the rotation angle.

Since this method requires radial sampling and independent measurement of the center position of the eye, it is slightly more difficult to implement than the subpicture method (Anthony Parker, 1983).

2.2 Radial Sampling - Theory of Action

The method of radial sampling of FFT takes advantage of the fact that the magnitudes of the Fourier transform are independent of translations in x and y directions of the image matrices. The magnitudes of the Fourier transform depend upon rotation of the image matrices.

A sector of the magnitudes of the Fourier transform is circularly sampled in the reference and the data images. The crosscorrelation of these sampled sector images is used to calculate the rotation angle of the two images. It was not possible to use this method for eye images because the strong peak of the crosscorrelation appeared in the non-iral part of the images (Anthony Parker, 1983).

2.3 Subpicture Method - Theory

Some of the basic subpicture method algorithms are shown in Figure 2.3.1 and Figure 2.3.2. The reference picture OCR1.PIC and the data picture OCR2.PIC have the dimension of 256*256.

The operator identifies two subpictures, A1.PIC and B1.PIC, which are subtracted from the reference picture OCR1.PIC. The dimension of these subpictures is 64*64 which enables short calculation time and FFT.

A region of interest which has 20*20 dimension is automatically defined by masking at the center of these subpictures, namely ROIA and ROIB. So, the operator must choose the x-y co-ordinates of these subpictures so that a clear landmark should be at the center of the subpicture and included in the 20*20 dimension of the region of interest.

Two other subpictures, A2.PIC and B2.PIC, are subtracted from the data picture OCR2.PIC at the same x-y co-ordinates as A1.PIC and B1.PIC. Two regions of interest, ROIA and ROIB, and two subpictures, A2.PIC and B2.PIC, are Fourier transformed and filtered with Mexican hat filter. The filtering operation boosts the moderately high frequencies and attenuates the low and very high frequencies. There are edge effects at the boundary of the image due to the filter. Therefore the outer three elements are masked to zero.

After being inverse Fourier transformed, ROIA is crosscorrelated with A2.PIC and ROIB is crosscorrelated with B2.PIC. The peak of each crosscorrelation is used as a measure of the translation of the region of interest between two subpictures.

The rotation angle calculation is simple.

(xa1, ya1) ;	x-y coordinates of subpicture A1.PIC
(xb1, yb1) ;	x-y coordinates of subpicture B1.PIC
(xa2, ya2) ;	x-y coordinates of crosscorrelation peak in the subpicture A2.PIC
(xb2, yb2) ;	x-y coordinates of crosscorrelation peak in the subpicture B2.PIC

The inclination of the line which connects (xa1, ya1) and (xb1, yb1) with respect to the horizontal x axis is given by

theta1=arctan((ya1-yb1)/(xa1-xb1))

The inclination of the line which connects (xa2, ya2) and (xb2, yb2) with respect to the horizontal x axis is given by

theta2=arctan((ya2-yb2)/(xa2-xb2))

Therefore, the rotation angle is given by

theta=theta1-theta2

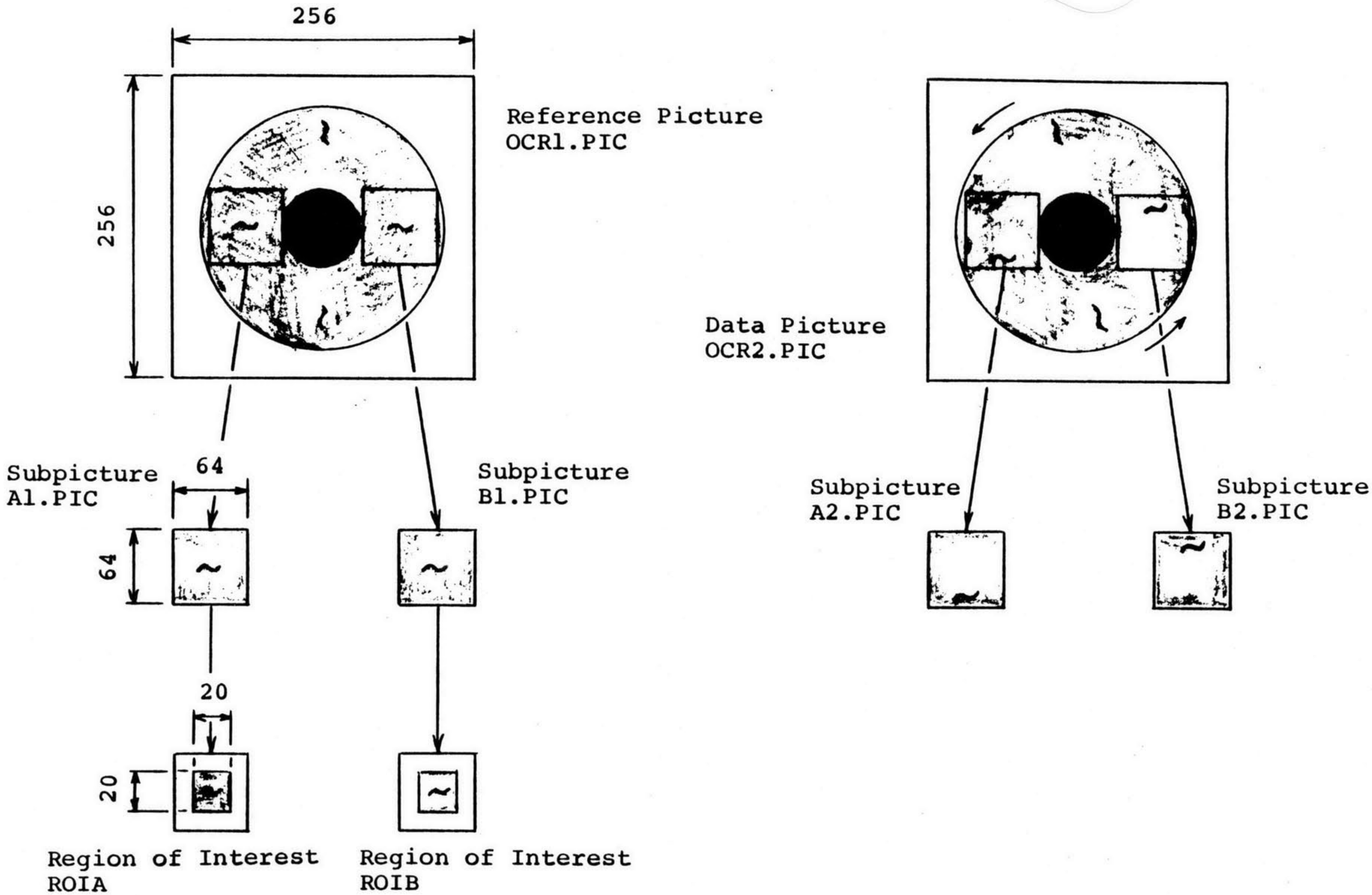


Fig. 2.3.1 Subpictures and Region of Interest

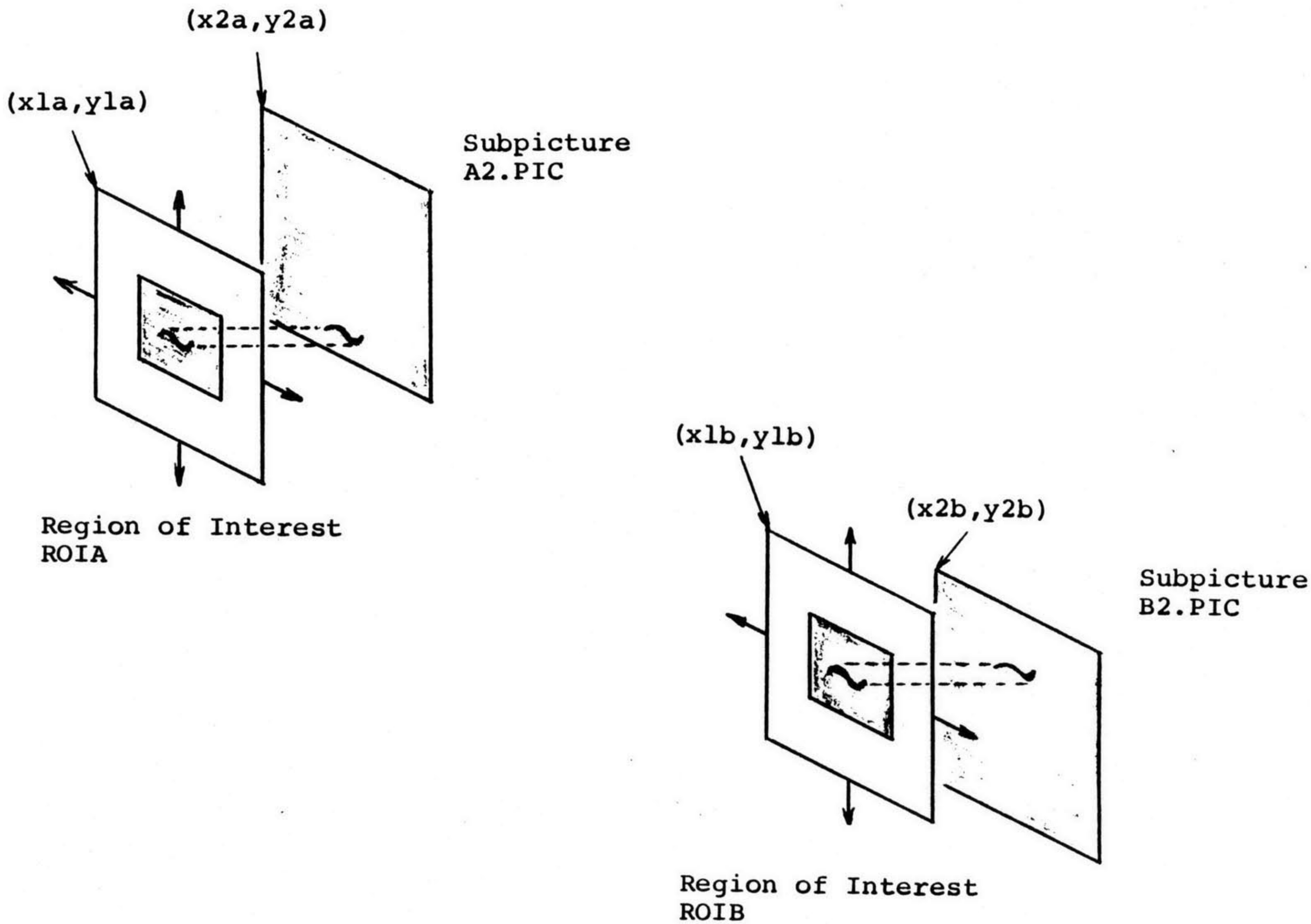


Fig. 2.3.2 Crosscorrelation of Subpictures

3. Preconditionig of Image - Mexican Hat Filter

The human observer can measure the translation of an object on the x-y plane which is perpendicular to the line of sight, or on the video screen. So, if the visual signal process of humans is simulated by computer algorithm, it should be possible for software to measure the translation of an object on the video screen. A successful process was to use the Mexican hat filter (Anthony Parker, 1983). Therefore, the same filter was used in this thesis.

This filter is the second derivative of a Gaussian function and given by

1/(sigma**2*sqrt(2*pi))*exp(-r**2/(2*sigma**2))

The name, Mexican hat, comes from the shape of this filter in the spacial domain.

The effect of the Mexican hat filter is to boost the moderately high frequencies and reduces the magnitude of low frequency component and very high frequency component. Figure 7.2 MH.TH and MHF.TH shows the Mexican hat filter in position domain and frequency domain respectively. The filtering effect is shown in Figure 7.2 A1.TH2, B1.TH2, A2.TH2 and B2.TH2.

4. Extracting Torsion Measurement - Crosscorrelation

Crosscorrelation is a well known operation defined by

Rf1f2(i',j')=sam(i,j) (f1(i,j) f2(*)(i-i',j-j'))

where

f1(i,j) is the first picture

f2(i',j') is the second picture

(*) is the conjugation operation

This crosscorrelation is normalized by the energy of the pictures

R'f1f2(i',j')=Rf1f2(i',j')/sqrt(ef1,ef2)

where

ef1 is the energy of the first image defined by

ef1=sum(i,j) (f(i,j) f(*)(i,j))

ef2 is the energy of the second image defined by

ef2=sum(i',j') (f(i',j') f(*)(i',j'))

In this thesis, the normalization factor is calculated by squaring the second image, multiplying the Fourier transform of the squared image by the transform of the mask, taking inverse Fourier transform and then taking the square root. The crosscorrelation and the normalization factor are shown in Figure 7.2 ONA.TH2 and ONB.TH2.

In this case, the image was filtered so as to increase the energy in the detail in the iris. For this reason, the peak in the crosscorrelation funtion falls in a region where the energy is near the maximum. The peak is apparent even in the unnormalized crosscorrelation (Anthony Parker 1983). When the peak is in a low energy region, it is often smaller than the side lobes. In that case, the normalization operation will make it possible to identify the peak when it could not otherwise be identified.

But, if there is a strong reflection of any image on the iris, such as reflection of a rotating dome, it might be possible that there are two crosscorrelation peaks, one is for the eye image and the other is for the reflection movement. So, the image data should be as clear as possible.

5. Hardware Used to Inplement the Ocular Torsion Algorithm

5.1 Hardware of Man-Vehicle Laboratory

Figure 5.1 shows the computer system at the Man-Vehicle Laboratory, which is built around a Digital Equipment Corporation, PDP-11/34 central processing unit with 256 kilobytes (kB) of main buffer memory, two 2.4 megabytes (MB) RK05 disk drives, a Microterm terminal ERGO 301, a WV-200P video camera, a NV-9300 video tape player, a WV-5300 video monitor and a Printronix MVP printer. The operating system used on this system is the RT-11 version 4.

The calculating power and speed are low because the buffer memory size is rather small and the system doesn't have an array processor. For instance, 256*256 byte dimension of matrix data array or 64*64 complex dimension of matrix data array cannot be defined in the buffer memory. A large program cannot be run and it was necessary that a large program must be divided into several parts. Two dimensional image data was manipulated every few rows of the image matrix and written in the disk memory with the record size of 128 (512 bytes).

5.2 Hardware of Tufts University Image Analysis Laboratory

Figure 5.2 shows the hardware system of Tufts Image Analysys Laboratory computer system. The hardware system is VAX-11/780 with 4 mega-byte buffer memory, and the operating system is VMS Version 4.1. This system has the floating point accelerator.

The calculation power and speed are much higher than RT-11

System, because the calculation can be done in the buffer memory and doesn't need I/O operation between the buffer and the disc memory. While the program for RT-11 System needs 75 minutes to analyze an image, the program for VAX System needs only 2 minutes with I/O operation and only 30 seconds without I/O operation between the buffer and the disc memory.

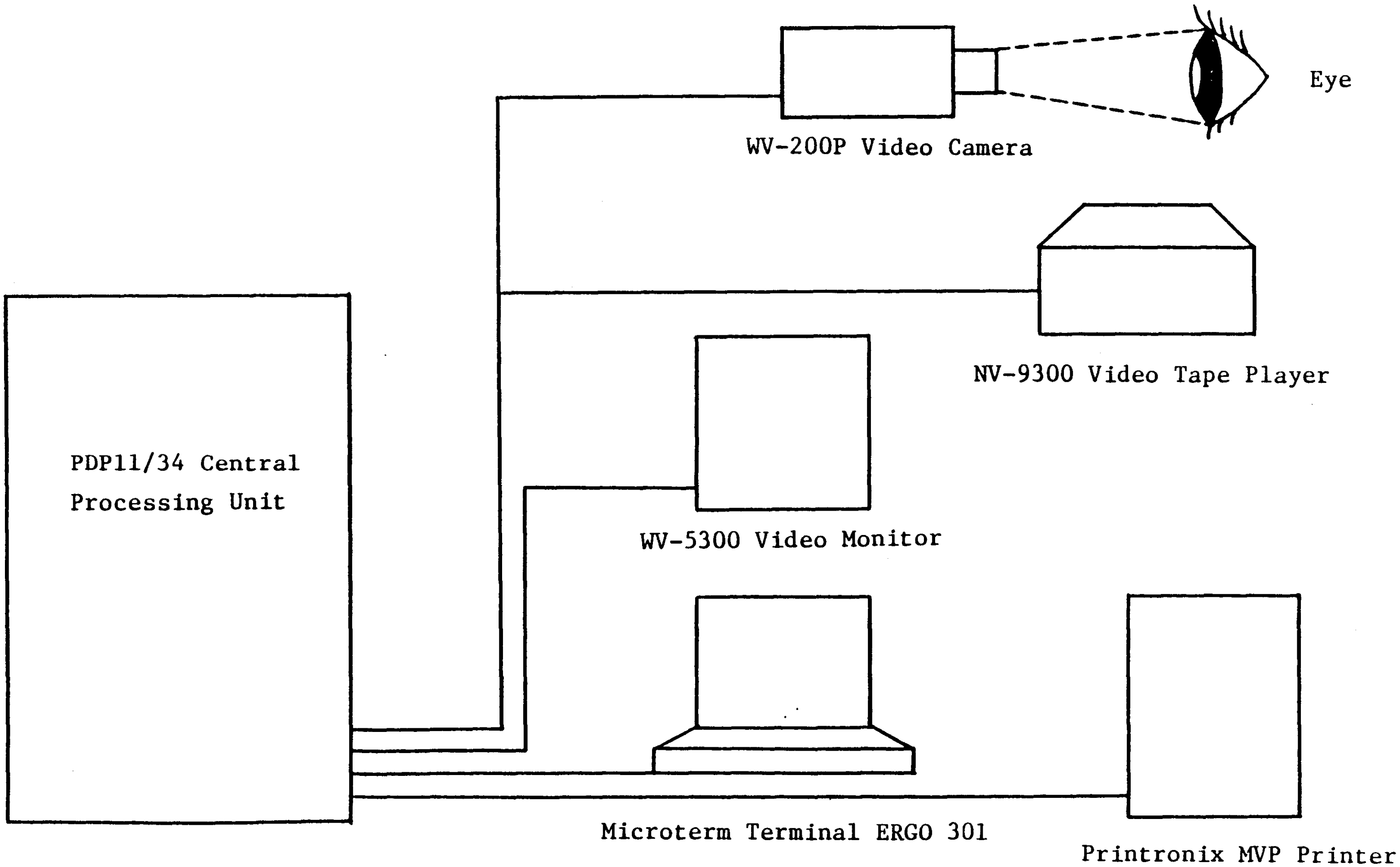


Fig. 5.1 Computer System at Man-Vehicle Laboratory

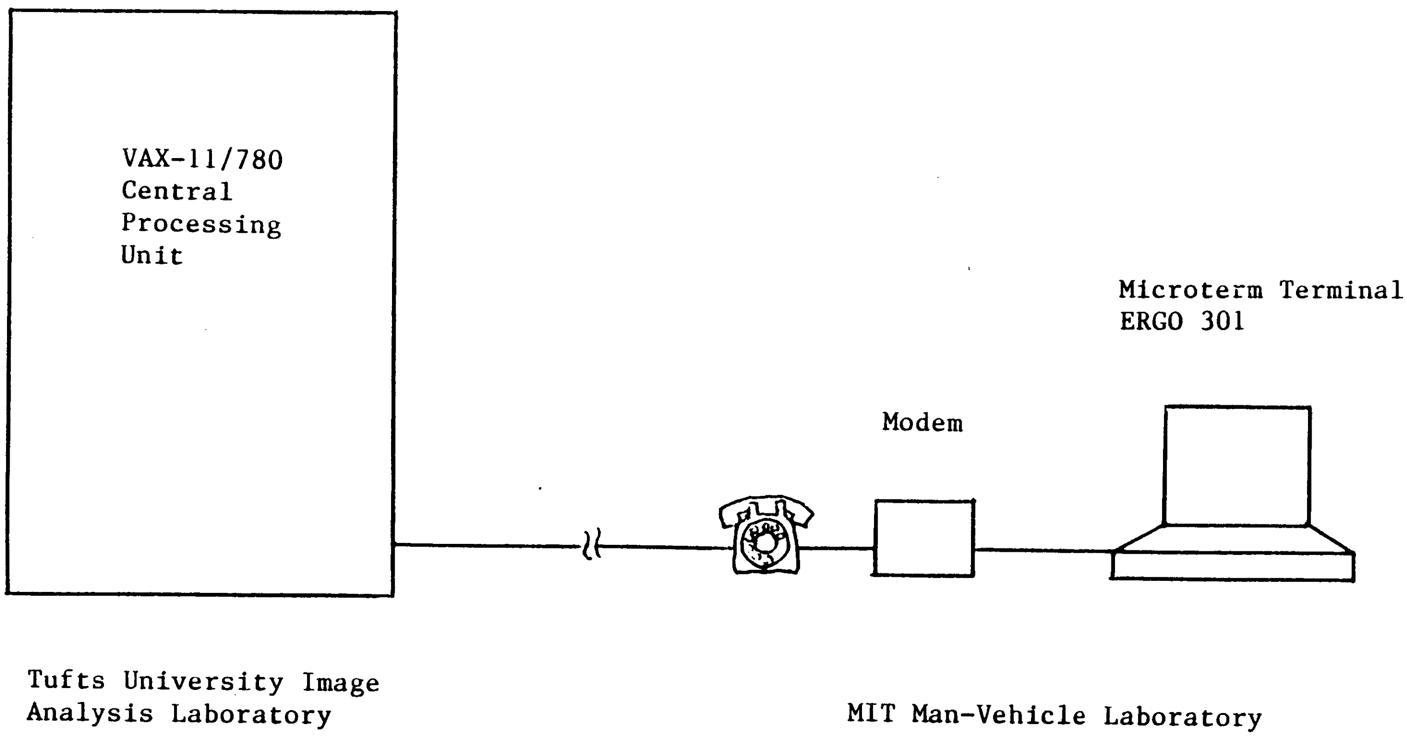


Fig. 5.2 Computer System at Image Analysis Laboratory

6. Data Format and Image Manipulation

6.1 Data Format

A frame grabber generates 256*256 dimension image data arrays. Each element is a byte which has 8 bit intensity capacity (-127 to 127). An image can be taken from the video camera or the video tape player and displayed on the video monitor.

Image data is transformed from bytes into complex numbers by adding 0 element of the complex part for FFT and digital filtering.

6.2 Image Acquisition and Manipulation

The following commands are useful for image manipulation.

SET VD:CNTRL=0	Reset the video screen
SET VD:CLEAR=n*10	Set whole display equal to n*10
SET VD:SCROLL=n*10	Load the scroll register
SET VD:LEFT=n*10	Set left bound of the screen at n*10
SET VD:RIGHT=n*10	Set right bound of the screen at n*10
SET VD:TOP=n*10	Set top bound of the screen at n*10
SET VD:BOTTOM=n*10	Set bottom bound of the screen at n*10
SET VD:CNTRL=10000	Connect the video monitor directly with the video camera or the video tape player
R RUNAVE	Because of the high noise level in

	the video frame grabber, multiple image had to be averaged to obtain a reasonable quality image. This command generates an averaged image on the video screen.
COPY VD:OCR1.PIC DAT:	Copy the image on the video screen named OCR1.PIC into DAT: disk

These commands can be refered in VDHELP.TXT in system disk of RT-11.

7. Software Description for RT-11 System

The programs for finding the image rotation are shown in Appendix 1. The used language for RT-11 System is RATFOR which can be preprocessed to standard Digital Equipment Corporation FORTRAN. A large program was divided into four parts; YOS11.RAT, YOS22.RAT, YOS3.RAT and YOS4.RAT. They were compiled with "/NOLINENUMBER" because of the small size of the central processor buffer memory.

YOS11.RAT is the main program for finding the image rotation angle. Input data files for this program are OCR1.PIC and OCR2.PIC. Output file is OCRBOX.DAT which includes the rotation angle of the pictures. YOS22.RAT, YOS3.RAT and YOS4.RAT are the programs of subroutines which must be linked to the mainprogram.

7.1 YOS11.RAT

YOS11.RAT is the main program for finding the image rotation angle. The data flow chart is shown in Figure 7.1. Picture data are shown in Figure 7.2. Input data files for this program are OCR1.PIC (a reference picture) and OCR2.PIC (a data picture) stored in logical unit DAT:. These files are 256*256 bite dimension raw pictures. The data files manipulated in this program are filed in the disc memory of the computer evey time when they are created or manipulated by subroutines because of the rather small size of the buffer memory with the recordsize 128 (512 bytes). This makes the calculation time so long as 75 minutes.

A picture descriptor block is added to the top of the files by the subroutine "pdb". The picture descriptor block is 512 bytes and has the information of the picture file such as the dimension of the picture, the maximum value of the picture elements and the x-y coordinates of the subpicture. The new files to which the picture descriptor block is added are OCR1.PDB and OCR2.PDB.

Subroutine "box" is called to define the x-y coordinates of the subpictures A1.PIC and B1.PIC. The operater can see the subpictures on the video screen. The x-y coordinate data is stored in the data file A.DAT for the subpicture A1.PIC and in the data file B.DAT for the subpicture B1.PIC by the subroutines "bopenf" to open the files and "rput" to store the data.

Mask file defines the region of interest and stored in file ONE.Z. This file is created by the subroutine "maksr". The dimension of the mask (64*64) must be the same as the subpictures. The intensity of the elements inside the mask is 1 and the intesity outside the mask is 0. Masking routine is that if the element of the mask is 0, the corresponding element of the output subpicture element is also 0. The location of the mask is the center of the subpicture (32,32). The size of the mask is (20*20). Therefore, the operater must choose the subpictures so that the subpictures may include a clear landmark of the iris.

OCR1.PDB and OCR2.PDB are transformed into complex value format by adding zero element of the imaginary part of the data by the subroutine "czcvt" so that Fourier transform and filtering can be done. From this part of the program, the picture data is manipulated in complex format.

Four subpictures are taken using x-y coordinates data, A.DAT and B.DAT. The names of the subpictures are A1.Z and B1.Z for the reference picture, and A2.Z and B2.Z for the data picture. Each element of these subpictures is a complex number and the dimension is (64*64).

Subroutine "masr8" is called to create the Mexican hat filter. This subroutine only creates 1/4 of the whole filter on the subpictures. The filter must be shifted at the center of the subpicture and rotated, because the filter is created only in the area where x-y coordinates are positive. Subroutine "rot11" rotates the filter and create the whole filter in the space domain.

Mexican hat filter is Fourier transformed by the subroutine "xform option 'FFT'". The file name is MHF.Z. The subpictures A1.Z, B1.Z, A2.Z and B2.Z are also Fourier transformed by the same subroutine and filtered with MHF.Z. Filtering operation is done by multiplying each element of the Fourier transform of subpictures by the element of the filter.

Subroutine "xform optiton 'Inverse FFT'" is called to inverse Fourier transform the subpictures A1.Z, B1.Z, A2.Z and B2.Z and edge masked by the edge mask MSK.Z. This mask file is created by the subroutine "maksr", and all three elements of the subpictures are masked. The masking operation is the same as the filtering operation, that is the muliplication of the each element of the mask and the subpictures. This is the end of the Mexican hat filtering operation.

Following is the crosscorrelation routine. The algorithm of this routine is explained in the section 4.

A2.Z and B2.Z are normalized and masked by the subroutine "normf" and squared by the subroutines "copy" and "twofil (option multiplication)". The masking file name is ONE.Z. The squared data subpictures are AA2.Z and BB2.Z. All these files, A2.Z, B2.Z, AA2.Z and BB2.Z, are Fourier transformed by the subroutine "xform (option FFT)".

The data subpictures, A2.Z and B2.Z, are multiplied by the complex conjugates of A1.Z1 and B1.Z1. AA2.Z and BB2.Z are also multiplied by the complex conjugates of ONE.Z1 and ONE.Z2. Rsulting data files are ONE.Z1 and ONE.Z2. A1.Z1, B1.Z1, ONE.Z1 and ONE.Z2 are inverse Fourier transformed. The square root files of ONE.Z1 and ONE.Z2 are taken by the subroutine "onef19". A1.Z1 and B1.Z1 are divided by ONE.Z1 and ONE.Z2 respectively. Rsulting files, A1.Z1 and B1.Z1, are the crosscorrelation functions. The square root of these files, ONE.Z1 and ONE.Z2, are the normalization factors. Detail explanation of the normalization factor is given in section 7.2.

Two final crosscorrelation functions, A1.Z1 and B1.Z1, are used to find the peak of the correlation and the translation of the region of interest by the subroutine "peak4". The actual rotation angle is calculated by the subroutine "calc20" and the data are stored in the file OCRBOX.DAT.

7.2 YOS22.RAT

YOS22.RAT is composed of main subroutines which are called by the main program YOS11.RAT. Basically, these main subroutines open files and the calculation data are stored in the files because of the small size of the buffer. 128 of recordsize (512 bytes) is used for I/O. This causes the calculation time to be slow.

subroutine pdb(cstr1,cstr2)

The subroutine pdb puts the picture descriptor block (512 bytes) at the top of the picture data file. " cstr1 " is an input data file name and " cstr2 " is an output data file name. Horizontal dimension is stored in the fifteenth block and vertical dimension is stored in the sixteenth block.

subroutine maksr(cstr,iszx,iszy,rmag,x0,y0,xc,yc)

This subroutine makes a mask file. The mask file is like a window. Each element inside the window has the value of " rmag " (usually rmag=1.0), while the elements outside the window are zero's such as

```
          zline(ix)=zmag
     else
          zline(ix)=(0.,0.)
```

The center of the window is (xc,yc). The size of the window is (2*x0)*(2*y0). (iszx,iszy) must be the same as the dimension of the subpicture which is to be masked (64*64). The window defines the region of interest in the subpictures. "cstr" is the mask file name such as MSK.Z or ONE.Z. Subroutine "ouplnz" is called to file the mask data into the disc memory line by line of the mask file elements.

subroutine subpic(cstr1,cstr2,cstr3,iszx2,iszy2)

This subroutine makes a subpicture file whose dimension is (iszx2*iszy2). In this thesis, the dimension is (64*64). The x-y coordinates data of the subpicture are stored in the file " cstr3 ". " cstr2 " is the subpicture file name taken from the original picture " cstr1 ". This program also puts the maximum value of the picture elements and the x-y coordinates of the subpicture as x-offset and y-offset in the picture descriptor block. The value of x-y offset is used by the subroutine " calc20 " for calculating the rotation angle.

subroutine maksr8(cstr,iszx,iszy,rmag,x0,y0,sigma)

This subroutine makes the Mexican hat filter centered at the upper left corner (0.0). Therefore, this filter must be shifted at the center of the subpicture and rotated. The value of the filter in the space domain is given by

```
zline(ix)=sigma_4_2_pi_inverse*(2.-var_inverse*rsq)*
          exp(-twovar_inverse*rsq) + i*0.     (i=root(-1))
[  zline(ix)=cmplx(sigma_4_2_pi_inverse*(2.-var_inverse*rsq)*
          exp(-twovar_inverse*rsq),0.)  ]
```

where

```
sigma_4_2_pi_inverse=1./(sigma**4*sqrt(2.*pi))
twovar_inverse=1./(2.*sigma**2)
var_inverse=1./sigma**2
```

Subroutine "ouplnz" is called to file the Mexican hat filter into the disc memory line by line of the filter elements.

subroutine rot11(cstr1,cstr2,ix,iy)

This subroutine shifts the Mexican hat filter, " cstr1 ", created by the subroutine maksr8 to (ix,iy) and rotates it around the center of the subpicture, since the filter file which is created by the subroutine "maksr" is only 1/4 of the whole filter at the upper left corner of the subpicture. The statement for the rotation is

```
zline2(mod(i-1+ix,iszx1)+1)=zline(i)
```

The data of "zline2" is written by the subroutine "ouplnz" line by line.

The new file " cstr2 " is the Mexican hat centered at (ix,iy).

subroutine xform(iop,cstr)

This subroutine does the two dimensional discrete Fourier transform. " iop=10 " is Fourier transform. " iop=11 " is inverse Fourier transform. Transformation is done row by row of the image data matrix by the subroutine "fft", and the matrix is transposed by the subroutine "transp". Fourier transform is done row by row again for two dimension. Another matrix transpose may be taken. But, in this subroutine, it is eliminated for saving time.

subroutine twofil(cstr1,cstr2,iop,rmin)

This subroutine operates multiplication of the two files, " cstr1 " and " cstr2 " such as

```
zrec2(i)=zrec1(i)*zrec2(i)
```

and multiplication of " cstr1 " by the conjugate of the file " cstr2 " such as

```
zrec2(i)=zrec1(i)*conjg(zrec2(i))
```

and division of the file " cstr1 " by " cstr2 ". The operation is done picture element by element. In order to avoid zero-divide, the minimum value of the second file elements is defined by " rmin ", that is

```
if(abs(real(zrec1(i))).ge.rmin)
     zrec2(i)=real(zrec2(i))/real(zrec1(i)) + i*0.  (i=root(-1))
     [  zrec2(i)=cmplx(real(zrec2(i))/real(zrec1(i)),0.)  ]
```

else

zrec2(i)=(0.,0.)

where

zrec1(i) is an element of the file "cstr1"

zrec2(i) is an element of the file "cstr2".

subroutine normf(cstr1,cstr2)

This subroutine is used for creating a picture file of the region of interest from the mask file " cstr1 " and the subpicture file " cstr2 ". At the first part of this program, mean value of the subpicture elements " zmean " and the normalization factor " rootsq_inverse " are calculated.

$$\text{rootsq\_inverse} = \frac{1.}{\sqrt{\text{sumsq-abs(zsum)/npel}}}$$

[rootsq_inverse=1./sqrt(sumsq-cabs(zsum)/npel)]

where

" sumsq " is the sum of the absolute value of all picture elements.

" zsum " is the sum of the all picture elements.

" npel " is the number of the picture elements (64*64=4096).

If the picture element of the mask file is zero, the element of the output file is also zero, which means the masking. If the picture element of the mask file is not zero, the element of the subpicture is normalized by " rootsq_inverse ".

subroutine onefil9(cstr)

This subroutine takes the square root of the real part of the picture element by the statement

$$zrecl(i) = \sqrt{max(0. , real(zrecl(i))} + i*0. \qquad (i=root(-1))$$

```
[  zrec1(i)=cmplx(sqrt(amax1(0.,real(zrec1(i)))),0.)  ]
```

where zrec1(i) is an element of the subpicture data file.
If the real part is negative, the result is set to zero in the output file.

subroutine peak4(cstr1,cstr2,n)

This subroutine calculates the x-y coordinates of the correlation peak, (rx,ry) from the crosscorrelation function " cstr1 ". The peak position is interpolated between the peak element and the next element by the high resolution cubic spline. This function was used successfully by Anthony Parker (1983), so the same function is used in this subroutine. Interpolation is " t0x,t0y ".

```
t0x=t0(temp(1),temp(2),temp(3))
```

where

```
temp(i)=f_of_t(t0y,f(i,1),f(i,2),f(i,3))
t0(f1,f2,f3)=(f1-f3)/(2.*(f1-2.*f2+f3))
f_of_t(t0,f1,f2,f3)=(f1-2.*f2+f3)/2.*t0**2+(f3-f1)/2.*t0+f2
```

f(i,j) is the element of the crosscorrelation function.

and

```
t0y=t0(temp(1),temp(2),temp(3))
```

where

```
temp(i)=f_of_t(t0x,f(1,i),f(2,i),f(3,i))
t0(f1,f2,f3)=(f1-f3)/(2.*(f1-2.*f2+f3))
```

f_of_t(t0,f1,f2,f3)=(f1-2.*f2+f3)/2.*t0**2+(f3-f1)/2.*t0+f2

f(i,j) is the element of the crosscorrelation function.

The subpicture location in the reference picture is stored in the picture descriptor block and is read in this program (xoffset,yoffset). The correlation peak location is the sum of these values.

peak location=(xoffset+rx+t0x , yoffset+ry+t0y)

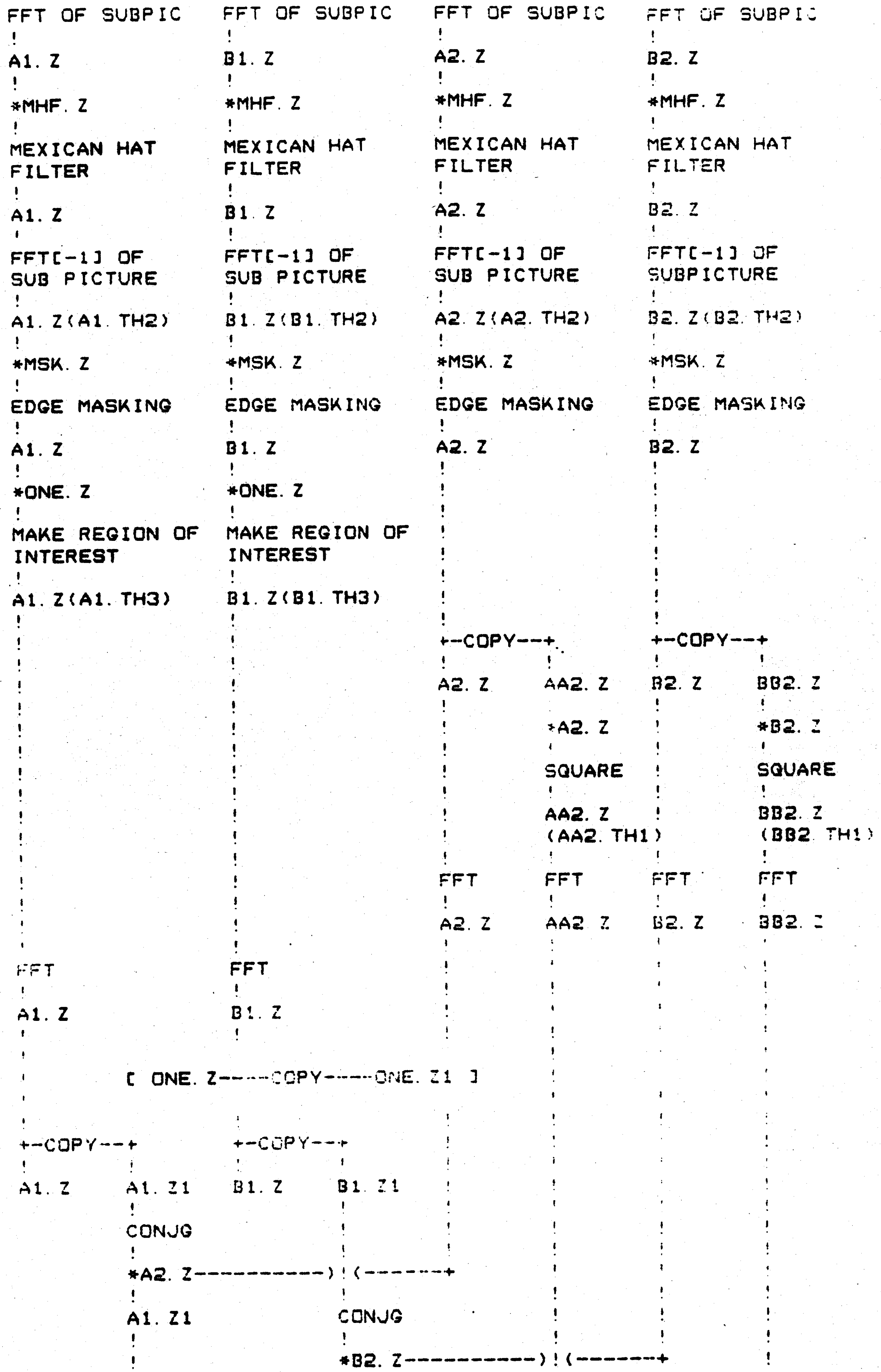
subroutine calc20(cstr,n1,n2,n3)

This subroutine calculates the rotation angle from the two correlation peak positions (x2a,y2a) and (x2b,y2b), and two subpicture locations (x1a,y1a) and (x1b,y1b).

rotation angle=arctan(y1a-y1b, x1a-x1b) - arctan(y2a-y2b, x2a-x2b)

```
******************************************************************************
THESDT.TXT

Thesis data explanation

Feb-5-95          Rev Non

[illegible]hiro Nagashima

******************************************************************************

       INPUT                                INPUT

       OCR1.PIC                             OCR2.PIC
       (256*256)                            (256*256)
       (REFERENCE PICTURE)                  (DATA PICTURE)
       !                                    !
       !                                    !
       ADD PICTURE DESCRIPTER               ADD PICTURE DESCRIPTER
       BLOCK                                BLOCK
       !                                    !
       OCR1.PDB                             OCR2.PDB
       !                                    !
       !       A.DAT(DEFINE SUB             !
       !               PICTURE A            !
       !               LOCATION)            !
       !       B.DAT(DEFINE SUB             !
       !               PICTURE B            !
       !               LOCATION)            !
       !                                    !
       !                                    !
       !       ONE.Z(MAKE MASK FILE)(ONE.TH)
       !       (64*64)                      !
       !       (REGION OF INTEREST IS       !
       !       ALSO DEFINED)                !
       !                                    !
       DATA TRANSLATE INTO COMPLEX          DATA TRANSLATE INTO COMPLEX
       NUMBER                               NUMBER
       !                                    !
       OCR1.Z                               OCR2.Z
       !                                    !
       !                                    !
       +---------------+                    +---------------+
       !               !                    !               !
       *A.DAT          *B.DAT               *A.DAT          *B.DAT
       !               !                    !               !
       MAKE SUB PIC    MAKE SUB PIC         MAKE SUB PIC    MAKE SUB PIC
       !               !                    !               !
       A1.Z(A1.TH1)    B1 Z(B1.TH1)         A2.Z(A2.TH1)    B2.Z(B2.TH1)
       (64*64)         (64*64)              (64*64)         (64*64)
       !               !                    !               !
       !
       !       [       MH.Z(MAKE MEXICAN HAT FILTER)(MH.TH)      ]
       !       [       !                                         ]
       !       [       FFT                                       ]
       !       [       !                                         ]
       !       [       MHF.Z(MHF.TH)                             ]
       !       [                                                 ]
       !       [       MSK.Z(MAKE EDGE MASK OF MEXICAN           ]
       !       [       HAT FILTER)(MSK.TH)                       ]
       !
       !                                                    !
```

Fig. 7.1 Data Flow Chart


FFT OF SUBPIC
A1. Z
*MHF. Z
MEXICAN HAT FILTER
A1. Z
FFT[-1] OF SUB PICTURE
A1. Z(A1. TH2)
*MSK. Z
EDGE MASKING
A1. Z
*ONE. Z
MAKE REGION OF INTEREST
A1. Z(A1. TH3)
FFT
A1. Z
FFT OF SUBPIC
B1. Z
*MHF. Z
MEXICAN HAT FILTER
B1. Z
FFT[-1] OF SUB PICTURE
B1. Z(B1. TH2)
*MSK. Z
EDGE MASKING
B1. Z
*ONE. Z
MAKE REGION OF INTEREST
B1. Z(B1. TH3)
FFT
B1. Z
FFT OF SUBPIC
A2. Z
*MHF. Z
MEXICAN HAT FILTER
A2. Z
FFT[-1] OF SUB PICTURE
A2. Z(A2. TH2)
*MSK. Z
EDGE MASKING
A2. Z
+--COPY--+
A2. Z
AA2. Z
*A2. Z
SQUARE
AA2. Z (AA2. TH1)
FFT
FFT
A2. Z
AA2 Z
FFT OF SUBPIC
B2. Z
*MHF. Z
MEXICAN HAT FILTER
B2. Z
FFT[-1] OF SUBPICTURE
B2. Z(B2. TH2)
*MSK. Z
EDGE MASKING
B2. Z
+--COPY--+
B2. Z
BB2. Z
*B2. Z
SQUARE
BB2. Z (BB2. TH1)
FFT
FFT
B2. Z
BB2. Z
[ONE. Z----COPY----ONE. Z1]
+--COPY--+
A1. Z
A1. Z1
CONJG
*A2. Z----------)
A1. Z1
+--COPY--+
B1. Z
B1. Z1
(-------+
CONJG
*B2. Z----------)
(-------+

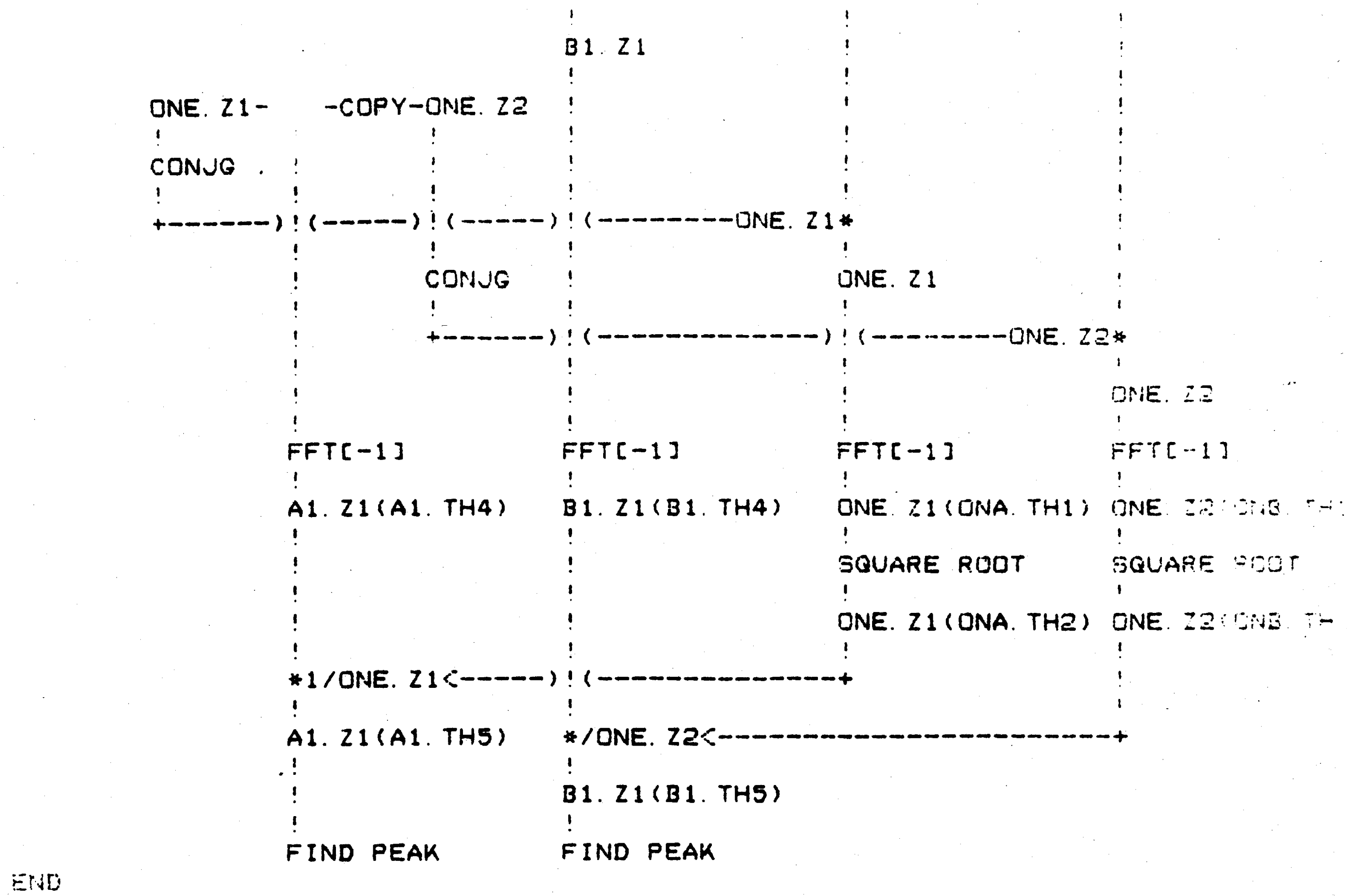
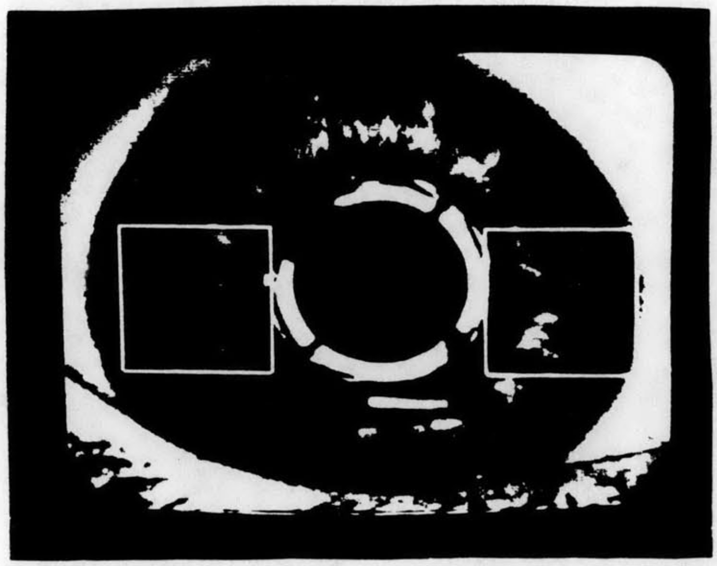
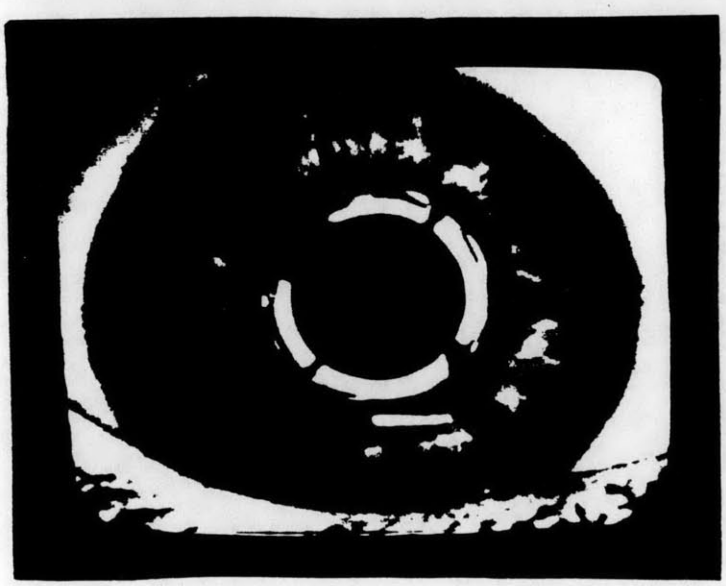


Fig. 7.1 Data Flow Chart

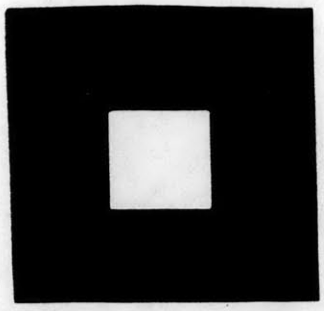


OCR1.PIC
Reference Picture and
Two Subpictures



OCR2.PIC
Data Picture

Fig. 7.2 Picture Data



ONE.TH
Mask File
Dimension is 64*64.
Mask dimension is 20*20.
The elements inside the mask are 1s (white), and outside the mask are 0s (black).

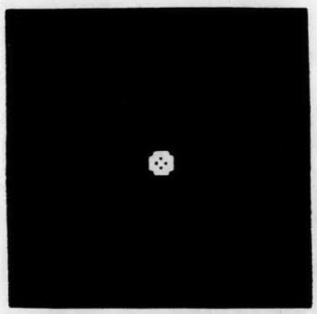
A1.TH1
Subpicture of reference picture, OCR1.PIC
Dimension is 64*64.

B1.TH1
Subpicture of reference picutre, OCR1.PIC
Dimension is 64*64.

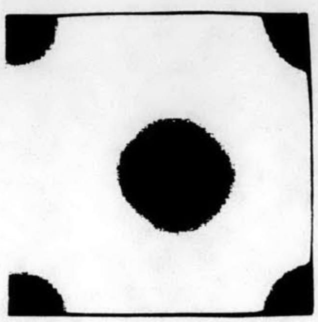
A2.TH1
Subpicture of data picture, OCR2.PIC
Dimension is 64*64.

Fig. 7.2 Picture Data

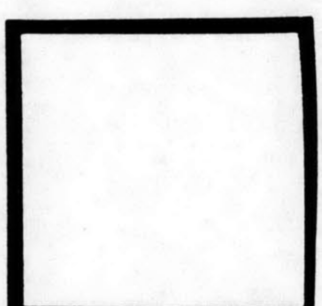
B2.TH1
Subpicture of data picture,
OCR2.PIC
Dimension is 64*64.



MH.TH
Mexican hat filter in
space domain. This is already
shifted to the center of the
subpicture, and rotated around
the center to get the whole
filter.

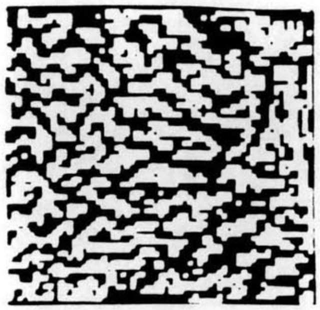


MHF.TH
Two dimensional Fourier
transform of Mexican hat
filter.

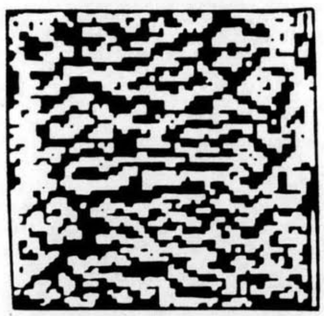


MSK.TH
Mask file for Edge Masking.
Dimension is 64*64.
Mask dimension is 58*58.
The elements inside the mask
are 1s(white), and outside
the mask are 0s(black).

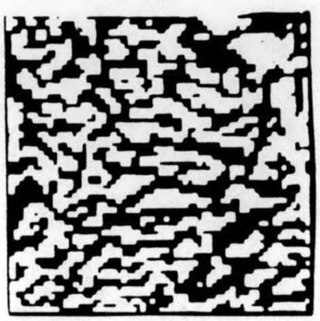
Fig. 7.2 Picture Data



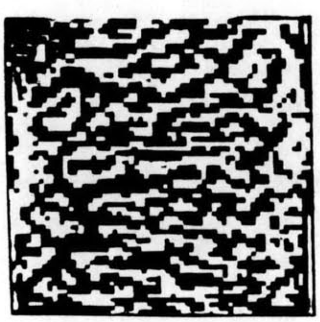
A1.TH2
Mexican hat filtered subpicture
of the reference picture.



B1.TH2
Mexican hat filtered subpicture
of the reference picture.

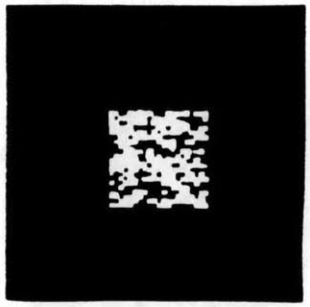


A2.TH2
Mexican hat filtered subpicture
of the data picture.

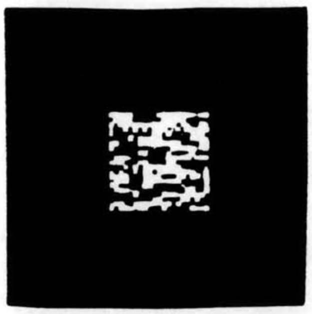


B2.TH2
Mexican hat filtered subpicture
of the data picture.

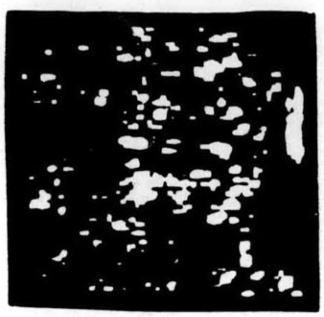
Fig. 7.2 Picture Data



A1.TH3
Region of interest.
This is the multiplication of the two files, the reference subpicture A (already filtered) and the mask file.
The dimension of the region of interest is 20*20.



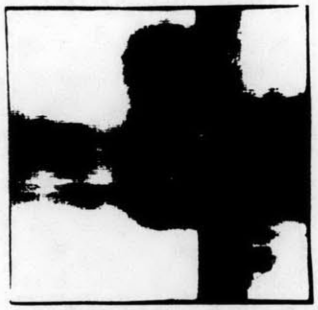
B1.TH3
Region of interest.
This is the multiplication of the two files, the reference subpicture B (already filtered) and the mask file.
The dimension of the region of interest is 20*20.



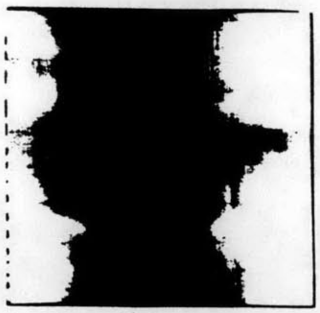
AA2.TH1
Square of A2.TH
This is used to get the crosscorrelation function.

BB2.TH1
Square of B2.TH
This is used to get the crosscorrelation function.

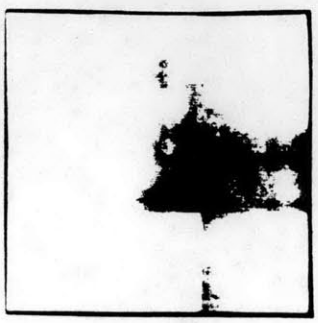
Fig. 7.2 Picture Data



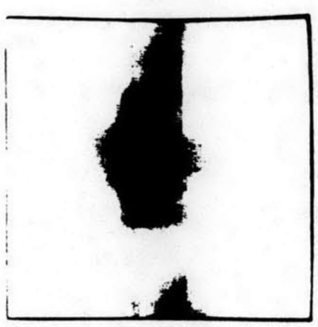
ONA.TH1
This is the multiplication of the conjugate of mask file and AA2.TH1, and used to get the normalization factor.



ONB.TH1
This is the multiplication of the conjugate of mask file and BB2.TH1, and used to get the normalization factor.



ONA.TH2
Square root of ONA.TH1
This is the normalization factor.



ONB.TH2
Square root of ONB.TH1
This is the normalization factor.

Fig. 7.2 Picture Data

8. Software Description for VAX System

The programs for finding the image rotation are shown in Appendix 2. The used language is RATFOR and the programs are compiled by RT-11 RATFOR compiler to FORTRAN, since Tufts VAX System doesn't have a RATFOR compiler.

The program is composed of four parts: YTFS1.FOR, YTFS2.FOR, YTFS3.FOR and YTFS4.FOR. The algorithm of the calculation is the same as the software for RT-11 System. But the image data manipulation is done in the buffer memory defining image data matrices, which enabled the program to be simple and calculation time to be short. This is the advantage of using the VAX System.

YTFS1.FOR defines subpicture location. YTFS2.FOR creates several picture files from the reference picture OCR1.PIC, which are necessary to use the main program. YTFS3.FOR is the main program for analyzing the image rotation. YTFS4.FOR is the program of subroutines which must be linked to the other programs.

8.1 YTFS1.FOR

This program defines subpicture location. At first, the operator must get a reference picture on the video screen and run this program. Subroutine "box" is called to define the x-y coodinates of the two subpictres, A.PIC and B.PIC. Subpicture dimension (64*64) is also defined in this program. The x-y coordinates data are stored in the files A.DAT for the subpicture A1.PIC and B.DAT for the subpicture B1.PIC. This program must be run in RT-11 System at the Man-Vehicle Laboratory and the data files must be transfered to the VAX System at Tufts University Image Analysis Laboratory.

8.2 YTFS2.FOR

This program creates several picture files from the reference picture OCR1.PIC, which are necessary to analyze the data picture OCR2.PIC using the program YTFS3.FOR. Inputs are the files of reference picture and x-y coordinates of subpictures, A.DAT and B.DAT. Output files are complex value transformed subpictures, A1.Z and B1.Z, a mask file, ONE.Z, which defines the region of interest in the subpictures, Fourier transform of Mexican hat filter, MHF.Z, and MSK.Z which is an edge masking file. These two dimensional data array matrices, ONEZ,A1Z,B1Z,MHZ,MHFZ,MSKZ, can be defined in the large buffer memory of the VAX System. This enables the very short analyzing time which was impossible for the RT-11 system of the Man-Vehicle Laboratory.

Subroutine "ymaksr" is called to create the mask file, ONE.Z. Dimension of this mask (64*64) must be the same as the subpictrues and defined by this calling routine. Magnitude of the mask, which is the magnitude of each picture element of this data file, is 1 so that there is not any change of the intensity of the data pictures. Dimension of the mask is also defined here and it is 20*20. The location of the mask is (32,32) which is the center of the subpicture. Therefore, the dimension of the region of interest is 20*20 and the location is the center of the subpictures. The operater should choose the subpicture location so that the region of interest includes a clear landmark of the iris.

Subroutine "ysbpic" is called to create the subpictures, A.PIC and B.PIC. This subroutine opens the data files of the x-y coordinates of the subpictures, A.DAT and B.DAT, and the reference picture, OCR1.PIC. A.PIC is created from OCR1.PIC using the coordinates A.DAT, and B.PIC is created from OCR1.PIC using the coordinates B.DAT. Subpicture data array matrices, A1Z and B1Z, are defined in the buffer memory of the computer.

Subroutine "ymksr8" is called to create the Mexican hat filter, MHZ. The dimension of the filter (64*64) must be the same as the subpictures. As the filter is created at the upper left corner, it must be shifted to the center of the subpicture. The shifting amount is (32,32).

Subroutine "yrot11" is called to rotate the filter around the center of the file (32,32), since the filter created by the subroutine "ymksr8" is 1/4 of the whole filter on the subpictures. The new data array matrix name is MHFZ.

Subroutine "yxform" is called to Fourier tramsfrom the Mexican hat filter MHFZ.

Subroutine "ykeep" is called to store the data array MHFZ in the data file MHF.Z. This is necessary because MHF.Z is used by YTFS3.FOR.

Subroutine "ymaksr" is called to create the edge mask array matrix MSKZ. The dimension of the edge mask is the same as the Mexican hat fiter. Intensity of the elements is 1 as well as the mask file MHK.Z. The location of the mask is the center of the filter (32,32) and all three elements of the edge are masked.

Subroutine "ykeep" is called to store the data array matrix MSKZ in the data file MSK.Z. This is necessary because MSK.Z is used by YTFS3.FOR.

Subroutine "yxform" is called to Fourier transform the subpicture data array matrices, A1Z and B1Z, which are already defined in the buffer memory of the computer.

Subroutine "y2fil" is called to filter the subpicture data array matrices, A1Z and B1Z, with the Mexican hat filter data array matrix, MHFZ. The filtering routine is done by multiplying each picture element by the data element of the filter.

The data array matrices, A1Z and B1Z, are inverse Fourier tranformed for the edge masking by the subroutine "yxform (option FFT[-1])" and multiplied by the edge masking array matrix, MSKZ, by the subroutine "y2fil (option multiplication)".

Subroutine "ynormf" is called to create the region of interest data array matrices, A1Z and B1Z, using the mask file array matrix, ONEZ. This routine is done by multiplying each subpicture element by the mask leement. Inside the region of interest, the intensity of each element of the mask is 1, and the intensity outside the region of interest is 0. The masking routine is that if the mask element is 0, the output matrix element is also 0. Each output element is normalized by the normalization factor. Detail explanation is given in section 8.4.

Region of interest data array matrices, A1Z and B1Z, and mask array matrix, ONEZ, are stored in the data files, A1.Z, B1.Z and ONE.Z, so that they can be used by other main programs such as YTFS3.FOR.

8.3 YTFS3.FOR

This is the main program for analyzing the image rotation using several data files created by YTFS2.FOR.

Input data files are OCR2.PIC which is 256*256 byte dimension raw image data file, A1.Z and B1.Z, which are the subpictures of the reference picture, A.DAT, B.DAT, MHF.Z and MSK.Z. The rotation angle data are stored in the output file OCRBOX.DAT.

The two dimensional data array matrices, A2Z,B2Z,MHFZ,MSKZ,AA2Z, BB2Z,ONEZ1,ONEZ2,A1Z1,B1Z1, can be defined in the large buffer memory of the VAX system. This enables the simple composition of the programs and the very short analyzing time for analysis because I/O operation between the buffer and the disc memory is not necessary, which was not possible for the RT-11 system of the Man-Vehicle Laboratory. The calculation takes 75 minutes for RT-11 System and only 30 seconds for VAX System to analyze an image.

Subroutine "ysbpic" is called to create the subpicture array matrices, A2Z and B2Z, form the data picture file, OCR2.PIC, which is opened by this subroutine. X-Y coordinate data files, A.DAT and B.DAT, are also opened by this subroutine and used for creating the subpicture data matrices.

Subpicture array matrices, A2Z and B2Z, are Fourier transfromed by the subroutine "yxform (option FFT)" for filtering.

Subroutine "filbuf" gets the Mexican hat filter data file and defines it as a two dimensional data array matrix, named MHFZ, in the buffer memory.

The subpicture data array matrices, AZ2 and B2Z, are multiplied, element by element, by the filter array matrirx, MHFZ, which is defined in the buffer memory by the subroutine "filbuf" from the filter data file, MHF.Z. This is the Mexican hat filtering operation.

After the filtering, subpicture matrices, A2Z and B2Z, are inverse Fourier transfromed by the subroutine "yxform (option FFT[-1])" and edge masked. The edge masking operation is also the multiplication of the subpicture data array matrix and the edge masking data array matrix, MSKZ, element by element. This is the end of the Mexican hat operation.

The following is the crosscorrelation operation. The algorithm of the crosscorrelation is explained in the section 4.

Subroutine "copy" is called to create the same subpicture data array matrices in the buffer memory in order to get the energy of the pictures. A2Z and B2Z are copyed in the array matrices, AA2Z and BB2Z, respectively. A2Z and AA2Z, and ,B2Z and BB2Z, are multiplied respectively to get the square value of the subpictures and Fourier transformed before the crosscorrelation.

Subroutine "filbuf" is called to get the subpicture data files of the reference picture, A1.Z and B1.Z, and defines the array matrices, A1Z1 and B1Z1, in the buffer. Complex conjugate matrices of the A1Z1 and B1Z1 are multiplied by the data subpicture matrices, A2Z and B2Z.

On the other hand, spuare value matrices of the data subpictures, AA2Z and BB2Z, are multiplied by the conjugate matrices of the mask file, ONEZ1 and ONEZ2, which are the same at this time. Resulting array matrices are ONEZ1 and ONEZ2. These array matrices, A1Z1,B1Z1,ONEZ1,ONEZ2, are inverse Fourier transformed. Square root matrices of ONEZ1 and ONEZ2 are taken, and A1Z1/ONEZ1 and B1Z1/ONEZ2 are the crosscorrelation functions. These operations such as the multiplication and the division are the calculation of element by element of the array matrices. This is the end of the crosscorrelation operation. Resulting crosscorrelation functions are A1Z1 for the subpicture A2.PIC and B1Z1 for the subpicture B2.PIC.

Subroutine "ypeak4" is called to identify the crosscorrelation peak of the function. The peak coordinates of A1Z1 and B1Z1 are (x2a,y2a) and (x2b,y2b) respectively on the matrix of the subpicture (64*64).

Therefore the coordinates of the subpictures are also necessary to get the peak location on the original 256*256 dimension coordinates.

Subroutine "datatr" transfers the data from the data files, A.DAT and B.DAT into the buffer, and gets the x-y coordinates data of the subpictures on the original picture matrix (256*256) into the buffer.

Subroutine "yclc20" calculates the actual rotation angle using the x-y coordinates of the subpictures on the original 256*256 matrix ((x1a,y1a) and (x2b,y2b)) and the crosscorrelation peak locations on the subpicture 64*64 matrices ((x2a,y2a) and (x2b,y2b)). Calculation algorithm is

angle=arctan(y1a-y1b,x1a-x1b) - arctan(y2a-y2b, x2a-x2b)

The rotation angle data is stored in the file OCRBOX.DAT.

8.4 YTFS4.FOR

This is the program of subroutines which are used by the main programs YTFS1.FOR, YTFS2.FOR and YTFS3.FOR. All of the data arrays manipulated in the subroutines are defined as two dimensional matrices of complex value. The data manipulation is done in the buffer memory of the computer and I/O operations are avoided as many as possible. Large buffer memory of the VAX system enabled these and very fast analyzing time and simple programs.

subroutine ymaksr(zsub,iszx,iszy,rmag,ix0,iy0,ixc,iyc)

The subroutine "ymaksr" creates a rectangle mask in the buffer memory for making the region of interest or the edge masking file. "zsub" is the name of the created mask such as ONEZ or MSKZ. "iszx" and "iszy" are the dimension of the mask data array matrix (64*64) which is the same as the subpictures. "rmag" is the magnitude of the mask inside the region of interest or the edge mask, and usually this is 1 so that there is not change of the intensity of the data files. "ix0" and "iy0" are the half size of the region of interest or the edge mask. "ixc" and "iyc" are the x-y coordinates of the center of the mask on the subpicture matrix.

Mask creating operation is

zsub(i,j)=1.   inside the mask

zsub(i,j)=0.   outside the mask

where zsub(i,j) is an element of the mask array matrix.

subroutine ysbpic(zsub,cstr1,cstr2)

This subroutine creates a subpicture array matrix in the buffer memory. "zsub" is the name of the subpicture array matrix such as A1Z or B1Z. "cstr1" is the name of the picture data file whose dimension is 256*256 byte dimension (not the complex value) such as OCR1.PIC or OCR2.PIC. "cstr2" is the file name of the x-y coordinates of the subpicture such as A.DAT or B.DAT.

Subroutine "bopenf" is called to open the file of the x-y coordinates of the subpicture center. The x-y coordinates data (x0,y0) is stored in the buffer memory by the subroutine "rget". Since these value are the center position of the subpicture on the matrix, the x-y coordinates of the upper left corner are necessary to create the subpicture, such as

ix0=x0-32.

iy0=y0-32.

The picture data file "cstr1" is opened in the buffer and data is read. But the recordsize of the picture file is 128, therefore only 512 bytes data is read. In this program, the matrix data array cpic(ii,jj) whose dimension is 256*256 byte is defined, so that the program is very much understandable. Each element of the matrix cpic(ii,jj) is a byte.

The matrix csub(k,l) is defined as the subpicture data array matrix whose dimension is 64*64 and each element of the matrix is still a byte.

The last routine of this subroutine is the byte-complex convert. Byte is converted into the integer value by the following statements.

```
equivalence (c,ic)                 ! c and ic are equivalent
data ic/0/                         ! clear high byte
c=csub(i,j)
isub(i,j)=ic
```

The integer value matrix is converted into a complex value matrix by adding a complex zero element to each real part of the data such as

```
zsub(i,j)=cmplx(float(isub(i,j)),0.)
```

where zsub(i,j) is the 64*64 complex dimension subpicture data array matrix.

subroutine ymksr8(zsub,iszx,iszy,rmag,x0,y0,sigma)

This subroutine is to create the Mexican hat filter matrix in the buffer. "zsub" is the name of the filter, iszx and iszy are the dimension of the filter which must be the same as the subpictures such as 64*64,rmag is a dummy variable,x0 and y0 are the center location of the filter and sigma is the standard deviation. The created filter is centered at the upper left corner (0,0). Therefore, this filter must be shifted at the center of the subpicture and rotated. The value of the filter in the space domain is given by

```
zsub(i,j)=sigma_4_2_pi_inverse*(2.-var_inverse*rsq)*
               exp(-twovar_inverse*rsq) + i*0.        (i=root(-1))
[  zsub(i,j)=cmplx(sigma_4_2_pi_inverse* \
     (2.-var_inverse*rsq)*exp(-twovar_inverse*rsq),0.)  ]
where
var_inverse=1./sigma**2
twovar_inverse=1./(2.*sigma**2)
sigma_4_2_pi_inverse=1./(sigma**4*sqrt(2.*pi))
```

subroutine yrot11(zsub1,zsub2,ix1,iy1)

This subroutine is to rotate the Mexican hat filter around the center of the matrix by the following statement.

```
zsub2(i,(mod(j-1+ix,64)+1))=zsub1(i,j)
where
zsub1(i,j) is the original Mexican hat filter
zsub2(i,j) is the rotated Mexican hat filter
```

Fourier transform is still necessary for the filtering.

subroutine yxform(iop,zsub)

This program is to two dimensional Fourier transfrom if the option number iop is 10, and is to two dimensional inverse Fourier transfrom if the option number is not 10. "zsub" is the data array matrix name which is to transformed.

zline(j) is defined by each row of the matrix zsub(i,j) and the subroutine "fft" operates the Fourier transform row by row. Subroutine "ytrnsp" is called to get the matrix transpose by the routine such as

zsub(i,j)=zsub(j,i)

and subroutine "fft" is called again and operated row by row for the two dimensional FFT. The matrix transpose may be taken again, but in this program, it is ignored for saving the calculation time.

subroutine y2fil(zsub1,zsub2,iop,rmin)

This subroutine is to operate multiplication, conjugate multiplication and division element by element of the picture array matrix whose dimension is 64*64.

The option number, iop, 20 is for the multiplication by the following statement.

zsub2(i,j)=zsub1(i,j)*zsub2(i,j)

The option number 21 is for the conjugate multiplication by the following statement.

zsub2(i,j)=zsub1(i,j)*conjg(zsub2(i,j))

The option number 23 is for the division by the following statement.

zsub2(i,j)=cmplx(real(zsub2(i,j))/real(zsub1(i,j)),0.)

If the real part of zsub1(i,j) is smaller than rmin1, zsub2(i,j)=(0.,0) to avoid the overflow.

where rmin1=rmin*"maximum absolute value of the real part of zsub2(i,j)"

subroutine ynormf(zsub1,zsub2)

This subroutine is to create the array matrix of the region of interest from the mask data matrix zsub1 and the subpicture data matrix zsub2. At the first part of this subroutine, mean value of the subpicture elements "zmean" and the normalization factor "rootsq_inverse" are calculated such as

$$\text{rootsq\_inverse}= \frac{1.}{\sqrt{\text{sumsq-abs (zsum)/npel}}}$$

[rootsq_inverse=1./sqrt(sumsq-cabs(zsum)/npel)]

where

"sumsq" is the sum of the absolute value of all picture elements of zsub2(i,j).

"zsum" is the sum of the all picture elements of zsub2(i,j).

"npel" is the number of the picture elements (64*64=4096).

If the picture element of the mask matrix is zero, the element of the output matrix is also zero, which means the masking. If the picture element of the mask array is not zero, the each element of the subpicture matrix is normalized by "rootsq_inverse" such as

```
zsub2(i,j)=(zsub2(i,j)-zmean)*rootsq_inverse
```

subroutine y1fil(zsub)

This subroutine is to get the square root of each element of the data matrix zsub(i,j) by the following statement.

```
zsub(i,j)=cmplx(sqrt(amax1(0.,real(zsub(i,j)))),0.)
```

subroutine ypeak4(zsub,x,y,rn)

This subroutine calculates the x-y coordinates of the correlation peak (x,y), from the crosscorrelation function zsub(i,j). The peak position is interpolated between the peak element and the next element by the following interpolation operations.

```
t0x=t0(temp(1),temp(2),temp(3))
```

where

```
temp(j)=f_of_t(t0y,f(j,1),f(j,2),f(j,3))
f_of_t(t0,f1,f2,f3)=(f1-f3)/(2.*(f1-2.*f2+f3))
dimension f(3,3),temp(3)
```

and

```
t0y=t0(temp(1),temp(2),temp(3))
```

where

```
temp(j)=f_of_t(t0x,f(1,j),f(2,j),f(3,j))
f_of_t(t0,f1,f2,f3)=(f1-f3)/(2.*(f1-2.*f2+f3))
dimension f(3,3),temp(3)
```

and where

```
f(j,i)=real(zsub2(i,jj))
jj=mod(x+(j-2)+63,64)+1
zsub2(i,j)=zsub(mod(y+i-2+63,64)+1,j)
```

zsub(i,j) is the element of the crosscorrelation function. (t0x,t0y) is the interpolation. (x,y) is the peak location of the crosscorrelation function. Therefore, calculated peak location is (x+t0x, y+t0y) on the subpicture coordinates. This is the high resolution cubic spline used by Anthony Parker (1983) successfully and the same function is used in this subroutine.

```
subroutine yclc20(x2a,y2a,x1a,y1a,x2b,y2b,x1b,y1b,cstr,m)
```

This subroutine is to calculate the image rotation angle from the two correlation peak positions (x2a,y2a) for the crosscorrelation function A1Z1 and (x2b,y2b) for the crosscorrelation function B1Z1, and two subpicture locations (x1a,y1a) for the subpicture A.PIC and (x1b,y1b) for the subpicture B.PIC. Calculation of the angle is a simple algorithm.

y2a1=y2a+y1a

x2a1=x2a+y2a

where (x2a1,y2a1) is the x-y coordinates of the peak for the subpicture A.PIC.

y2b1=y2b+y1b

x2b1=x2b+x1b

where (x2b1,y2b1) is the x-y coordinates of the peak for the subpicture B.PIC.

The inclination of the line between the centers of the subpictures A1.PIC and B1.PIC is given by

theta1=atan2(y1a-y1b,x1a-x1b)

The inclination of the line between the two peak locations of the crosscorrelations is given by

theta2=atan2(y2a1-y2b1,x2a1-x2b1)

The rotation angle is given by

theta=theta1-theta2

This data is stored in the data file "cstr".

9. Data Transmission

Picture data files must be transmitted from MIT to Image Analysis Laboratory at Tufts University for using the VAX System. Therefore, data format must be changed so that the data file can be transmitted. The program which transforms the picture data files created at MIT Man-Vehicle Laboratory into the transmittable format is shown in Appendix 3, and the program which transforms the new data format into the original data format is shown in Appendix 4.

9.1 MITFS.RAT

This program changes the data format which is transmittable from MIT Man-Vehicle Laboratory to the VAX System at Tufts University Image Analysis Laboratory.

Raw image data which is created at RT-11 System is an array of picture elements which can define a 256*256 dimension matrix. Each picture element is a byte which has eight bits. Therefore, the magnitude of each eight bit number indicates the intensity of the picture element and the range of magnitude is -127 to 127 (decimal). The data format as a binary number is eight bits such as

    ****

where

    * is 1 or 0.

On the other hand, the transmittable format is such that the first bit is 0, the second bit is 1, the third and the fourth bits are 0's, and the other four bits indicate the data, such as

0100****

Therefore, each picture element is divided into two parts. Higher four bits are shifted to the right by four bits and added to 01000000. Lower four bits are also added to 01000000. For instance, if the data of a picture element is 01101011, the resulting transformation is

01000110 for higher four bits

and

01001011 for lower four bits.

Magnitude range of the new data is 01000000 to 01001111 (binary number), or 100 to 117 (octal number), or 64 to 79 (decimal number).

All picture elements are transformed like this and a "line feed" (10 as a decimal number) and a "carriage return" (13 as a decimal number) are added every 64 bytes for the transmission.

9.2 TFSPIC.FOR

This program transforms the new data format into the original format. The transformation is the reverse operation of MITFS.RAT. 64 (01000000 as a binary number) is subtracted from every data element, shifted to the left by four bits for higher four digits and added to the next data which indicates the lower four bits of the picture element.

ASCII character chart is shown in Appendix 5.

10. Additional Experiments

10.1 Accuracy of the Algorithm

In order to examine the errors produced by the subpicture method without interference from other causes of error, digitally rotated images were artificially produced. These artifically produced images are quite useful for determining accuracy of the method since the true rotation are exactly known. When measuring rotsion on an experimental series of images, the precision of the measurement can be determined easily.

A well framed 256*256 image of a single eye of a blue-eyed subject was rotated digitally using the high resolution cubic spline interpolation algorithm. Images were produced at 1,2,3,4,5 and 6 degree rotations.

For each data picture, 9 pairs of subpictures, A2.PIC and B2.PIC were extracted around the subpictures shown in Figure 7.2 OCR1.PIC. Figure 10.1.1 shows the accuracy as the standard deviation with respect to the image rotation angle. This result shows that the subpicture method has a good accuracy compared to the required error limitation 0.1 degree between 0 and 4 degrees. The standard deviation is much larger in case of the 5 degree rotated image. This is because the subpicture method basically uses the subpicture image translation in x-y direction to calculate the rotation angle. It seems the calculation is limited to five degrees for this method, but this is enough for the eye rotation because the eye does not rotate more than about 5 degrees. Calculation was not possible for the six degree rotated image because of the same reason.

10.2 Comparison to the Result of Anthony Parker[1]

The algorithm of the subpicture method is a modification of the landmark tracking method which was developed by Anthony Parker [1]. The basic difference is that, in the landmark tracking method, the dimension of the region of interest is 64*64 and that this subpicture is crosscorrelated with the whole data picture (256*256 dimension), while in the subpicture method, the dimension of the region of interest is 20*20, and the region of interest is crosscorrelated with the subpicture (64*64). This made the calculation time much faster (75 minutes for the subpicture method, about 3 hours for the landmark tracking method).

The accuracy as the standard deviation shown in his thesis([1] p.98) can not be considered to be the real standard deviation for analyzing the actual data pictures, because the subpictures (of different size) were taken from the same part of the reference picture using the same x-y coordinates and crosscorrelated with the same area of the data picture, and so this data doesn't include the error for extracting the subpictures from the different part of the reference picture. And, in order to calculate the rotation angle, two crosscorrelation peak points are necessary. His data is the standard deviation of the translation of the crosscorrelation peak point of one subpicture in x-y direction using different size of subpictures (The unit is Picture Element). When two correlation peaks are used to calculate the rotation angle, the variance will be doubled. It is because of these reasons that the error shown in Figure 10.1.1 is larger than his result.

10.3 Analysis of Unclear Pictures

Figure 10.2.1 is the unclear eye image which has strong reflection on the contact lens. In this case, the eye rotated to the rotation of a dome in front of the eye. The reflection is the pattern inside the dome.

In this experiment, subpictures of the pattern on the contact lens were used. If there were no pattern on the lense, it might happen that the crosscorrelation would extract the movement of the reflection. So, it may be better to select the same bright colors of the pattern inside the dome, such as red and green, so that there is no difference in the brightness of the pattern when video-recorded in black and white.

Since the video camera was not on the line of sight, the subpicture method cannot measure the accurate rotation angle because the rotation angle is calculated by

rotation angle = tan(-1)[(y-coordinate of the subpicture)/ (x-coordinate of the subpicture)]

and x-coordinate of the subpicture is less than the real length since the eye is not watched perpendicularly.

But the crosscorrelation extracted the eye rotation using a pattern of the contact lense,in spite of the reflection. Figure 10.2.2 shows the eye movement time history.

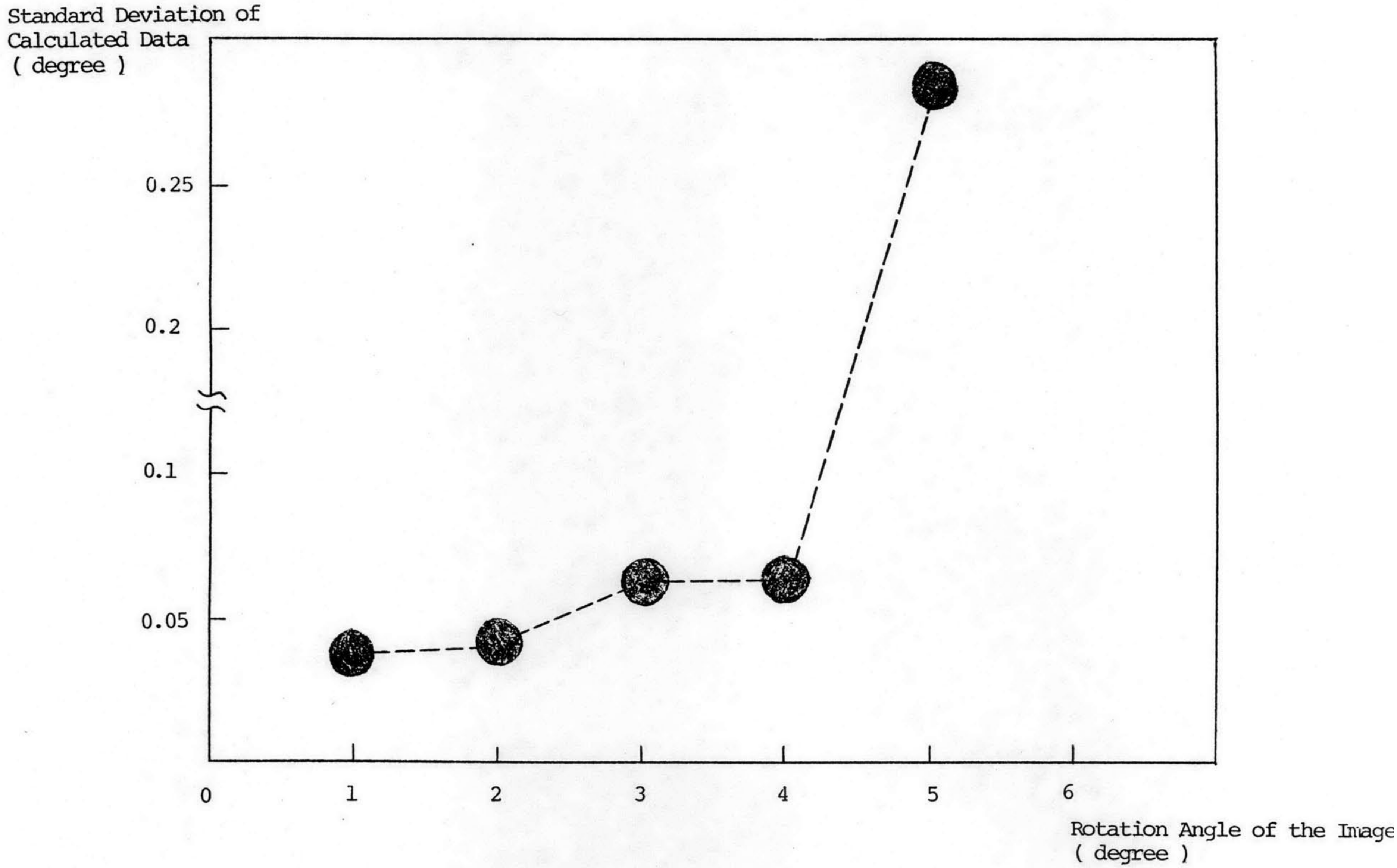


Fig. 10.1.1 Accuracy of the Algorithm

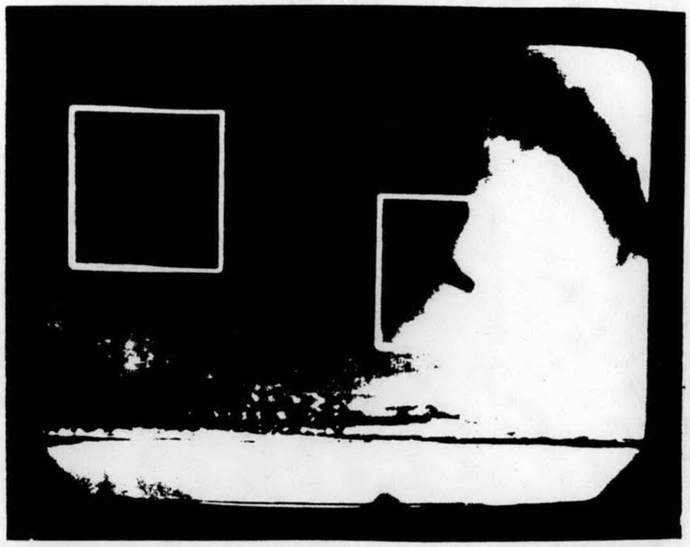


Fig. 10.2.1 Unclear Image

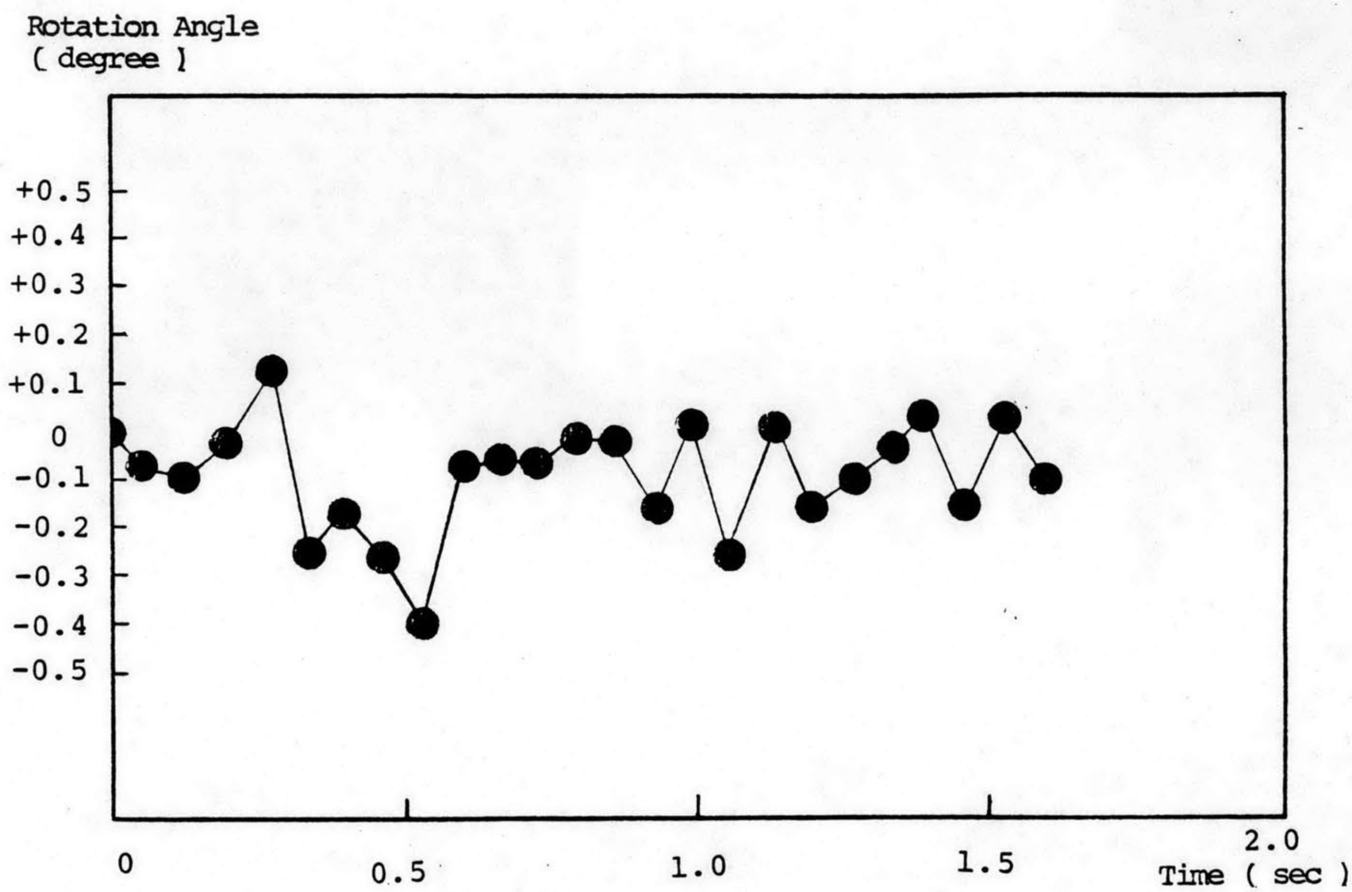


Fig. 10.2.2 Unclear Image Rotation Measurement
by the Subpicture Method

11. Conclusion

In this thesis, the measurement of image rotation was examined using digital image processing techniques. Ocular counterrolling image data was used since it is one of the few relatively direct indications of otolith functions. The digital image processing method can measure the rotation angle without using any device attached to the eye ball such as a contact lens, and is very reliable.

The purpose of this thesis was to develop a practical software for measureing ocular rolling. Two softwares were developed, the program for RT-11 System at the Man-Vehicle Laboratory and the program for VAX System at Tufts University Image Analysis Laboratory. RT-11 System takes 75 minutes and Tufts VAX System takes 30 seconds to analyze an image.

Five degrees is the maximum image rotation angle which can be calculated, because the subpicture method basically uses the translation of the subpictures and the crosscorrelation between the reference subpicture and the rotated data subpicture can not correlate if the angle is too large. The accuracy of the algorithm is less than 0.1 degree as the standard deviation for the image rotated by less than four degrees.

This algorithm could analyze the unclear image using the pattern drawn on the contact lens.

Appendix 1

Tutorial for using the programs for RT-11 System

If the edit of the program is necessary, put the program disc into RK2: , and edit the program. After this, compile the program according to the following procedure.

```
R RATFOR
* YOS11.RAT ( This is the name of the program which is edited. )
* ^c
FORT/NOLINENUMBER YOS11.FOR ( This is also the name of the edited
                               program. )
LINK YOS11,YOS22,YOS3,YOS4,BOX,SY:TPLIB/LINKLIB
BO RT11SJ
```

Then, put the reference picture OCR1.PIC and the data picture OCR2.PIC in RK1: , and get the reference picture on the video screen by the following command.

```
SET VD:CNTRL=0
COPY RK1:OCR1.PIC VD:
```

After this, assign RK1: to DAT: by the following command.

```
ASSIGN RK1: DAT:
```

Then, the data files are stored in DAT: .
Run the main program by the following command. (be sure that the program disc is in RK2:)

```
RUN RK2:YOS11
```

You will be asked to define the location of the subpictures. Move the box on the video screen by analog input and hit the space key to define the subpicture location.

```
/*****************************************************************************
*       PROGRAM YOS11.RAT
*       Ocular Counter Rolling Main Program
*       2-Nov-84         Rev. A
*       Yoshihiro Nagashima
*               Direct program file to find the rotation between two pictures
*               ( 1 and 2 ) from two correlations. Picture 1 is masked by
*               ROI-A and ROI-B, and then correlated with picture 2. From
*               the posions of the two regions of interest and the translation
*               from picture 1 to picture 2, the rotation is calculated.
*               The basic idea is that we are tracing two points on the eye.
*               With large rotation, the correlation will be affected by the
*               rotation of the data in the region of interest. Subpictures
*               are extracted from the two pictures centered around the region
*               of interest and the subpictures are used during correlation.
*
*       Inputs: ocr1.pic        raw picture 256*256 dimension  128pdb
*               ocr2.pic        raw picture 256*256 dimension  128pdb
*
*       Outputs:ocr1.pic
*               ocr2.pic
*               ocr1.z
*               ocr2.z
*               a1.z            subpicture A1 extracted from ocr1.z
*               a1.z1
*               a2.z            subpicture A2 extracted from ocr2.z
*               aa2.z           a2.z * a2.z
*               b1.z            subpicture B1 extracted from ocr1.z
*               b1.z1
*               b2.z            subpicture B2 extracted from ocr2.z
*               bb2.z           b2.z * b2.z
*               a.dat           x-y center data of subpicture A
*               b.dat           x-y center data of subpicture B
*               mh.z            Mexican Hat Filter
*               mhf.z           FFT of Mexican Hat Filter
*               msk.z           mask of MHF
*               one.z           mask data
*               one.z1          mask data
*               one.z2          mask data
*               ocrbox.dat      the last three data are theta ( radian ),
*                               theta ( degree ) and zero.
*       Caution:When you define the location of subpictures, please don't use
*               return key. Return key will make an error. Use space key, etc
*
*       r ratfor
*       * yos11.rat
*       * ^C
*       fort/nolinenumber yos11
*       link yos11,yos22,yos3,yos4,box,sy:tplib/linklib
*       bo rt11sj
*       run yos11
*****************************************************************************/
label NO_FILE,NO_FILE_2
define EOS1             0       # end of string 1
define EOS2             128
define VSV_LINES        25
define VSV_ADDR         172600
define BIT_MAP1         172620
define BIT_MAP2         172640
define CR               13
define LF               10
define SPACE            32
```

```
define CURSER_DISABLE   2000
define JSW              44      # address of job status word
define TT_SPECIAL_MODE  10100   # special mode bits in jsw
define DEL              127
define NULL             0
define MAX_ENTRIES      30
define FMT_STAT         format(g15.7)
define MAX_LINE         256     # Max picture dimension (pels/line)
define MAX_2_LINE       512     # Twice maximum picture dimension
define MAX_LINE_PLUS_1  257     # Maximun line plus 1
define RECORD_SIZE      512     # Largely determined by PDB size
define RECORD_SIZE_4    128     # Real*4 size
define RECORD_SIZE_8    64      # Complex*8 size
define PDB_RECORD       1       # Record number of picture descriptor block
define I_PDB_HORS       15      # Index of integer horizontal dimension
define I_PDB_VERS       16      # Index of integer vertical dimension
define I_PDB_XOFF       251     # Index of x offset in parent picture
define I_PDB_YOFF       252     # Index of y offset in parent picture
define I_PDB_OLD_DIM    250     # Index of demension of parent picture
define C_PDB_FORM       7       # Byte index in PDB of format flags
define PDB_FORM_WR      16      # Wrong reading bit in format byte i.e.(y,x)
define DATA_RECORD_1    2       # Record number of start of descriptor
define R_PDB_MAX_PEL    128     # Real index in PDB of max pel
define UNIT_1           1       # Logical unit for file 1
define UNIT_2           2       # Logical unit for file 2
define UNIT_3           3       # Logical unit for file 3
define PI               3.14159265358979
define VIRTUAL_BUFFERS  4       # Buffers equals blocks per line


define TP_IMPLICIT implicit byte(b-c),integer*2(i-n),real*4(a,e-h,o-y),\
                        real*8(d),complex*8(z)
define CHARACTER        byte
define COM_VSV          common /vsvc/line_num,bhold,b_no_vsv
define COM_TMP_STR      CHARACTER c_tmp_str(81); \
                        common /tmpstc/c_tmp_str
define COM_ASK          common /askc/idbase,b_indirect_command
define COM_SAVE_FILE    CHARACTER cstr2(81);\
                        common /sfc/lun,cstr2
define COM_OUTPUT       common /oupc/iunito,i_record_o,iptro,b_zero_pad
define COM_INPUT        common /inpc/iuniti,i_record_i,iptri,n_records_i
define COM_SIZE         common /szc/inpsz,ioupsz,ciform,coform, \
                                inp_oup_ratio,inp_oup_ratio_sq
define COM_ZREC_O       complex*8 zreco(RECORD_SIZE_8); \
                        common /zrecoc/zreco
define COM_PDB          CHARACTER cpdb(512); \
                        integer*2 ipdb(256); \
                        real*4 rpdb(128); \
                        equivalence (cpdb,ipdb,rpdb); \
                        common /pdbc/ipdb
define COM_ARRAYS       CHARACTER cline1(MAX_LINE); \
                        integer*2 line1(MAX_LINE),line2(MAX_LINE); \
                        complex*8 zline1(MAX_LINE); \
                        equivalence (cline1,line1,zline1); \
                        equivalence (line2(MAX_LINE),zline1(MAX_LINE)); \
                        common /arrayc/zline1
# |<-----------------------------------zline1------------------------------------>|
# |<-------line1------>|                                        |<-------line2------>|
# |<-cline1->|
define COM_Z_ARRAYS     complex*8 zline(MAX_2_LINE); \
                        complex*8 zline1(MAX_LINE),zline2(MAX_LINE); \
                        complex*8 zbuff(RECORD_SIZE_8,VIRTUAL_BUFFERS); \
```

```
                              complex*8 zrec1(RECORD_SIZE_8), zrec2(RECORD_SIZE_8);  \
                              real*4 rline1(MAX_2_LINE), rline2(MAX_2_LINE);  \
                              CHARACTER cline1(MAX_LINE), cline2(MAX_LINE);  \
                              CHARACTER crec1(RECORD_SIZE), crec2(RECORD_SIZE);  \
                              equivalence (zline, zline1, rline1, zbuff);  \
                              equivalence (zline(MAX_LINE_PLUS_1), zline2, rline2);  \
                              equivalence (zrec1, crec1), (zrec2, crec2);  \
                              common /zc/zrec1, zline, zrec2
define COM_1_ARRAY            complex*8 zline(MAX_LINE), zrec(RECORD_SIZE_8);  \
                              complex*8 zbuff(RECORD_SIZE_8, VIRTUAL_BUFFERS);  \
                              real*4 rline(MAX_2_LINE);  \
                              CHARACTER cline(MAX_LINE), crec(RECORD_SIZE);  \
                              equivalence (zline, rline, cline, zbuff);  \
                              equivalence (zrec, crec);  \
                              common /zc/zrec, zline
define COM_PIC_PARAM          equivalence (iszx, iszx1), (iszy, iszy1);  \
                              common /ppramc/iszx1, iszy1, iszx2, iszy2, scalex, scaley,  \
                                             unscalex, unscaley
define COM_GAM                common /gamc/iy, iframe, igamsz, i_oup_gam_ratio, b_byte_mtx
define ouplnz_sr              olnzsr
define ouplnz_eq_O            ouplnO



###########################################################################
###########################################################################
# main program
###########################################################################
###########################################################################
TP_IMPLICIT
COM_SIZE


call vsvfig(UNIT_1)
call vsvclr('all')

call vprint('Please wait a moment')
call pdb('DAT:OCR1.PIC', 'DAT:OCR1.PDB')
call vprint('okashiina')
call pdb('DAT:OCR2.PIC', 'DAT:OCR2.PDB')
call vprint('Please define subpicture A by the space key')
call box(ix, iy, 64, 64)            # define subpicture A location

x=float(ix-31)
y=float(iy-31)
call rprint('X of subpicture A is ', x)
call rprint('Y of subpicture A is ', y)
call bopenf('DAT:A.DAT', UNIT_1, 'new')
call rput('', x)
call rput('', y)
close(unit=UNIT_1)

call vprint('Please define subpicture B')
call box(ix, iy, 64, 64)            # define subpicture B location

x=float(ix-31)
y=float(iy-31)
call rprint('X of subpicture B is ', x)
call rprint('Y of subpicture B is ', y)
call bopenf('DAT:B.DAT', UNIT_1, 'new')
call rput('', x)
```

```
call rput('',y)
close(unit=UNIT_1)

call vprint('I am making a mask file')
call maksr('DAT:ONE.Z',64,64,1.,32.,32.,10.,10.)          # make mask
                                                   # mask name is ONE.Z
                                                   # mask dimension is 64*64
                                                   # magnitude is 1.
                                                   # ROI location is (31.,31.)
                                                   # ROI dimension is 10*10

ioupsz=256

call vprint('converting OCR1.PDB into OCR1.Z')
call inpset('DAT:OCR1.PDB')                  # convert OCR1.PDB into OCR1.Z
call oupset('DAT:OCR1.Z')
call czcvt
call ouppdb
call vsvclr('characters')

call vprint('converting OCR2.PDB into OCR2.Z')
call inpset('DAT:OCR2.PDB')                  # convert OCR2.PDB into OCR2.Z
call oupset('DAT:OCR2.Z')
call czcvt
call ouppdb
call vsvclr('characters')

call vprint('making subpicture A1.Z')
call subpic('DAT:OCR1.Z','DAT:A1.Z','DAT:A.DAT',64,64)
                                             # make subpicture A1.Z from OCR1.Z
                                             # location data is A.DAT
                                             # subpicture dimension is 64*64
call vprint('making subpicture B1.Z')
call subpic('DAT:OCR1.Z','DAT:B1.Z','DAT:B.DAT',64,64)
                                             # make subpicture B1.Z from OCR1.Z
                                             # location data is B.DAT
                                             # subpicture dimension is 64*64
call vprint('making subpicture A2.Z')
call subpic('DAT:OCR2.Z','DAT:A2.Z','DAT:A.DAT',64,64)
                                             # make subpicture A2.Z from OCR2.Z
                                             # location data is A.DAT
                                             # subpicture dimension is 64*64
call vprint('making subpicture B2.Z')
call subpic('DAT:OCR2.Z','DAT:B2.Z','DAT:B.DAT',64,64)
                                             # make subpicture B2.Z from OCR2.Z
                                             # location data is B.DAT
                                             # subpicture dimension is 64*64


##############################################################################
# mexican hat
##############################################################################

call vprint('making Mexican hat filter')
call maksrB('DAT:MH.Z',64,64,1.,32.,32.,0.7071068)       # make mexican hat filter
                                             # 64*64 dimension
                                             # magnitude is 1.
                                             # center is (32.,32.)
                                             # sigma is 0.7071068

call rot11('DAT:MH.Z','DAT:MHF.Z',-31,-31)         # rotation mask
                                                   # new data file is MHF.Z
```

```
call vprint('FFT of the Mexican hat filter')
call xform(10, 'DAT:MHF.Z')                      # option 10 is FFT
                                                 # output is MHF.Z ( FFT of mexican hat filter )
call vprint('making edge mask of filter')
call maksr('DAT:MSK.Z',64,64,1.,32.5,32.5,28.5,28.5)    # make edge mask of
                                                         # mexican hat filter
                                                         # mask dimension is 64*64

call vprint('FFT of subpictures')
call xform(10, 'DAT:A1.Z')                       # FFT of A1.Z
call xform(10, 'DAT:B1.Z')                       # FFT of B1.Z
call xform(10, 'DAT:A2.Z')                       # FFT of A2.Z
call xform(10, 'DAT:B2.Z')                       # FFT of B2.Z

call vprint('filtering subpictures A1.Z B1.Z A2.Z B2.Z with')
call vprint('Mexican hat filter')
call twofil('DAT:MHF.Z', 'DAT:A1.Z',20,0.0001)    # option 20 is z2=z1*z2
                                                  # z1 is MHF.Z
                                                  # z2 is A1.Z
call twofil('DAT:MHF.Z', 'DAT:B1.Z',20,0.0001)    # option 20 is z2=z1*z2
                                                  # z1 is MHF.Z
                                                  # z2 is B1.Z
call twofil('DAT:MHF.Z', 'DAT:A2.Z',20,0.0001)    # option 20 is z2=z1*z2
                                                  # z1 is MHF.Z
                                                  # z2 is A2.Z
call twofil('DAT:MHF.Z', 'DAT:B2.Z',20,0.0001)    # option 20 is z2=z1*z2
                                                  # z1 is MHF.Z
                                                  # z2 is B2.Z
call vprint('FFT[-1] of filtered subpictures')
call xform(11, 'DAT:A1.Z')                       # FFT[-1] of A1.Z
call xform(11, 'DAT:B1.Z')                       # FFT[-1] of B1.Z
call xform(11, 'DAT:A2.Z')                       # FFT[-1] of A2.Z
call xform(11, 'DAT:B2.Z')                       # FFT[-1] of B2.Z

call vprint('edge masking of subpictures')
call twofil('DAT:MSK.Z', 'DAT:A1.Z',20,0.0001)    # option 20 is z2=z1*z2
                                                         # z1 is MSK.Z
                                                         # z2 is A1.Z
call twofil('DAT:MSK.Z', 'DAT:B1.Z',20,0.0001)    # option 20 is z2=z1*z2
                                                         # z1 is MSK.Z
                                                         # z2 is B1.Z
call twofil('DAT:MSK.Z', 'DAT:A2.Z',20,0.0001)    # option 20 is z2=z1*z2
                                                         # z1 is MSK.Z
                                                         # z2 is A2.Z
call twofil('DAT:MSK.Z', 'DAT:B2.Z',20,0.0001)    # option 20 is z2=z1*z2
                                                         # z1 is MSK.Z
                                                         # z2 is B2.Z

###########################################################################
# end of mexican hat filter
###########################################################################


call vprint('making ROI from subpictures A1.Z & B1.Z')
call normf('DAT:ONE.Z', 'DAT:A1.Z')               # normalize A1.Z with ONE.Z
call normf('DAT:ONE.Z', 'DAT:B1.Z')               # normalize B1.Z with ONE.Z


###########################################################################
# ocrpc
###########################################################################
```

```
call vprint('correlating')
call copy('DAT:A2.Z','DAT:AA2.Z',65)              # copy recordsize is 65
call copy('DAT:B2.Z','DAT:BB2.Z',65)              # copy recordsize is 65

call twofil('DAT:A2.Z','DAT:AA2.Z',20,0.0001)     # op20 is z2=z1*z2
call twofil('DAT:B2.Z','DAT:BB2.Z',20,0.0001)     # op20 is z2=z1*z2

call vprint('FFT for correlating')
call xform(10,'DAT:A2.Z')                         # FFT
call xform(10,'DAT:B2.Z')                         # FFT
call xform(10,'DAT:AA2.Z')                        # FFT
call xform(10,'DAT:BB2.Z')                        # FFT

call vprint('I am tired')

call xform(10,'DAT:A1.Z')                         # FFT
call xform(10,'DAT:B1.Z')                         # FFT
call copy('DAT:ONE.Z','DAT:ONE.Z1',65)  # copy recordsize is 65
call xform(10,'DAT:ONE.Z1')                       # FFT

call copy('DAT:A1.Z','DAT:A1.Z1',65)
call copy('DAT:B1.Z','DAT:B1.Z1',65)
call vprint('Help me , Yoshi !')
call twofil('DAT:A2.Z','DAT:A1.Z1',21,0.0001) # op21 is z2=z1*conjg(z2)
call twofil('DAT:B2.Z','DAT:B1.Z1',21,0.0001) # op21 is z2=z1*conjg(z2)
call copy('DAT:ONE.Z1','DAT:ONE.Z2',65) # copy
call twofil('DAT:AA2.Z','DAT:ONE.Z1',21,0.0001) # z2=z1*conjg(z2)
call twofil('DAT:BB2.Z','DAT:ONE.Z2',21,0.0001) # z2=z1*conjg(z2)

call vprint('FFT[-1]')
call xform(11,'DAT:A1.Z1')                        # op 11 is FFT[-1]
call xform(11,'DAT:B1.Z1')                        # op 11 is FFT[-1]
call xform(11,'DAT:ONE.Z1')                       # op 11 is FFT[-1]
call xform(11,'DAT:ONE.Z2')                       # op 11 is FFT[-1]

call vprint('square root')
call onef19('DAT:ONE.Z1')                         # square root
call onef19('DAT:ONE.Z2')                         # square root

call vprint('almost done')
call twofil('DAT:ONE.Z1','DAT:A1.Z1',23,0.0001) # z2=z2/z1
call twofil('DAT:ONE.Z2','DAT:B1.Z1',23,0.0001) # z2=z2/z1

call vprint('finding peak')

call peak4('DAT:A1.Z1','DAT:OCRBOX.DAT',1)        # find peak

call ctrom('DAT:ONE.Z','DAT:OCRBOX.DAT','DAT:A.DAT',2)          # find center of mass

call peak4('DAT:B1.Z1','DAT:OCRBOX.DAT',3)                # find peak

call ctrom('DAT:ONE.Z','DAT:OCRBOX.DAT','DAT:B.DAT',4)          # find center of mass

call calc20('DAT:OCRBOX.DAT',1,3,5)              # calculation of angle

end
############################################################################
############################################################################
# end of main program
```

```
/*****************************************************************************
*       PROGRAM YOS22.RAT
*       Ocular Counter Rolling Main Subroutine Program
*       15-Aug-84
*       Yoshihiro Nagashima
*               Direct program file to find the rotation between two pictures
*               ( 1 and 2 ) from two correlations. Picture 1 is masked by
*               ROI-A and ROI-B, and then correlated with picture 2. From
*               the posions of the two regions of interest and the translation
*               from picture 1 to picture 2, the rotation is calculated.
*               The basic idea is that we are tracing two points on the eye
*               With large rotation, the correlation will be affected by the
*               rotation of the data in the region of interest. Subpictures
*               are extracted from the two pictures centered around the region
*               of interest and the subpictures are used during correlation.
*
*       Inputs: ocr1.pic        raw picture 256*256 dimension  128pdb
*               ocr2.pic        raw picture 256*256 dimension  128pdb
*
*       Outputs ocr1.pic
*               ocr2.pic
*               ocr1.z
*               ocr2.z
*               a1.z            subpicture A1 extracted from ocr1.z
*               a2.z            subpicture A2 extracted from ocr2.z
*               aa2.z           a2.z * a2.z
*               b1.z            subpicture B1 extracted from ocr1.z
*               b2.z            subpicture B2 extracted from ocr2.z
*               bb2.z           b2.z * b2.z
*               a.dat           x-y center data of subpicture A
*               b.dat           x-y center data of subpicture B
*               mh.z            Mexican Hat Filter
*               mhf.z           FFT of Mexican Hat Filter
*               msk.z           mask of MHF
*               one.z           mask data
*               one.z1          mask data
*               one.z2          mask data
*               ocrbox.dat      the last three data are theta ( radian ),
*                               theta ( degree ) and zero.
*       Caution When you define the location of subpictures, please don't use
*               return key. Return key will make an error. Use space key, etc.
*
*****************************************************************************/
label NO_FILE,NO_FILE_2
define EOS1             0       # end of string 1
define EOS2             128
define VSV_LINES        25
define VSV_ADDR         172600
define BIT_MAP1         172620
define BIT_MAP2         172640
define CR               13
define LF               10
define SPACE            32
define CURSER_DISABLE   2000
define JSW              44      # address of job status word
define TT_SPECIAL_MODE  10100   # special mode bits in jsw
define DEL              127
define NULL             0
define MAX_ENTRIES      30
define FMT_STAT         format(g15.7)
define MAX_LINE         256     # Max picture dimension (pels/line)
define MAX_2_LINE       512     # Twice maximum picture dimension
```

```
define MAX_LINE_PLUS_1   257       # Maximun line plus 1
define RECORD_SIZE       512       # Largely determined by PDB size
define RECORD_SIZE_4     128       # Real*4 size
define RECORD_SIZE_8     64        # Complex*8 size
define PDB_RECORD        1         # Record number of picture descriptor block
define I_PDB_HORS        15        # Index of integer horizontal dimension
define I_PDB_VERS        16        # Index of integer vertical dimension
define I_PDB_XOFF        251       # Index of x offset in parent picture
define I_PDB_YOFF        252       # Index of y offset in parent picture
define I_PDB_OLD_DIM     250       # Index of demension of parent picture
define C_PDB_FORM        7         # Byte index in PDB of format flags
define PDB_FORM_WR       16        # Wrong reading bit in format byte i e (y,x)
define DATA_RECORD_1     2         # Record number of start of descriptor
define R_PDB_MAX_PEL     128       # Real index in PDB of max pel
define UNIT_1            1         # Logical unit for file 1
define UNIT_2            2         # Logical unit for file 2
define UNIT_3            3         # Logical unit for file 3
define PI                3.14159265358979
define VIRTUAL_BUFFERS   4         # Buffers equals blocks per line
define TP_IMPLICIT implicit byte(b-c),integer*2(i-n),real*4(a,e-h,o-y), \
                           real*8(d),complex*8(z)
define CHARACTER         byte
define COM_VSV           common /vsvc/line_num,bhold,b_no_vsv
define COM_TMP_STR       CHARACTER c_tmp_str(81); \
                         common /tmpstc/c_tmp_str
define COM_ASK           common /askc/idbase,b_indirect_command
define COM_SAVE_FILE     CHARACTER cstr2(81);\
                         common /sfc/lun,cstr2
define COM_OUTPUT        common /oupc/iunito,i_record_o,iptro,b_zero_pad
define COM_INPUT         common /inpc/iuniti,i_record_i,iptri,n_records_i
define COM_SIZE          common /szc/inpsz,ioupsz,ciform,coform, \
                                    inp_oup_ratio,inp_oup_ratio_sq
define COM_ZREC_O        complex*8 zreco(RECORD_SIZE_8); \
                         common /zrecoc/zreco
define COM_PDB           CHARACTER cpdb(512); \
                         integer*2 ipdb(256); \
                         real*4 rpdb(128); \
                         equivalence (cpdb,ipdb,rpdb); \
                         common /pdbc/ipdb
define COM_ARRAYS        CHARACTER cline1(MAX_LINE); \
                         integer*2 line1(MAX_LINE),line2(MAX_LINE); \
                         complex*8 zline1(MAX_LINE); \
                         equivalence (cline1,line1,zline1); \
                         equivalence (line2(MAX_LINE),zline1(MAX_LINE)); \
                         common /arrayc/zline1
# |<-------------------------------zline1----------------------------------->|
# |<------line1------>|                                     |<-------line2------>|
# |-cline1->|
define COM_Z_ARRAYS       complex*8 zline(MAX_2_LINE); \
                          complex*8 zline1(MAX_LINE),zline2(MAX_LINE); \
                          complex*8 zbuff(RECORD_SIZE_8,VIRTUAL_BUFFERS), \
                          complex*8 zrec1(RECORD_SIZE_8),zrec2(RECORD_SIZE_8); \
                          real*4 rline1(MAX_2_LINE),rline2(MAX_2_LINE); \
                          CHARACTER cline1(MAX_LINE),cline2(MAX_LINE); \
                          CHARACTER crec1(RECORD_SIZE),crec2(RECORD_SIZE); \
                          equivalence (zline,zline1,rline1,zbuff); \
                          equivalence (zline(MAX_LINE_PLUS_1),zline2,rline2); \
                          equivalence (zrec1,crec1),(zrec2,crec2); \
                          common /zc/zrec1,zline,zrec2
define COM_1_ARRAY        complex*8 zline(MAX_LINE),zrec(RECORD_SIZE_8), \
                          complex*8 zbuff(RECORD_SIZE_8,VIRTUAL_BUFFERS), \
                          real*4 rline(MAX_2_LINE), \
                          CHARACTER cline(MAX_LINE),crec(RECORD_SIZE); \
```

```
                              equivalence (zline,rline,cline,zbuff); \
                              equivalence (zrec,crec); \
                              common /zc/zrec,zline
define COM_PIC_PARAM          equivalence (iszx,iszx1),(iszy,iszy1), \
                              common /ppramc/iszx1,iszy1,iszx2,iszy2,scalex,scaley, \
                                              unscalex,unscaley
define COM_GAM                common /gamc/iy,iframe,igamsz,i_oup_gam_ratio,b_byte_mtx
define ouplnz_sr              olnzsr
define ouplnz_eq_0            ouplnO




###############################################################################
# main subroutines
###############################################################################


subroutine pdb(cstr1,cstr2)
TP_IMPLICIT
CHARACTER cstr1(81)
CHARACTER cstr2(81)
COM_PDB
data ipdb/256*0/
%C
repeat[
        open(unit=UNIT_1,name=cstr1,type='OLD',access='DIRECT',\
             recordsize=RECORD_SIZE_4,associatevariable=iasv1,err=1)
        if(.false.) [
1                 call vsvclr('characters')
                  call vprint(cstr1)
                  call vprint('Input file not found')
                  next
                  ]
        break
        ]
iasv1=1
open(unit=UNIT_2,name=cstr2,type='NEW',access='DIRECT', \
     recordsize=RECORD_SIZE_4,associatevariable=iasv2)
ipdb(I_PDB_HORS)=256
ipdb(I_PDB_VERS)=256
iasv2=1
write(UNIT_2%'iasv2) ipdb
do i=1,128 [
        read(UNIT_1%'iasv1) ipdb
        write(UNIT_2%'iasv2) ipdb
        ]
close(unit=UNIT_1)
close(unit=UNIT_2)
#call vsvclr('characters')
return
end


subroutine mksr(cstr,iszx,iszy,rmag,x0,y0,xc,yc)          # make rectangle mask

# G(x)=(1/(sigma*sqrt(2*PI)))*exp(-x**2/(2*sigma**2))
# G''(x)=(1/(sigma**3*sqrt(2*PI)))*(x**2/sigma**2-1)*exp(-x**2/(2*sigma**2))
```

```
# G(x,y)=(1/(sigma**2*sqrt(2*PI)))*exp(-r**2/(2*sigma**2))
# grad(G(x,y))=(1/(sigma**4*2*PI))*(r**2/sigma**2-2)*exp(-r**2/(2*sigma**2))
TP_IMPLICIT
COM_1_ARRAY
COM_PDB
do i=1,256
        ipdb(i)=0
call opnfnw(cstr,UNIT_1,i_assoc_var,iszx,iszy)
zmag=cmplx(rmag,0.)
xmax=amin1(x0-1.,float(iszx)-x0)
ymax=amin1(y0-1.,float(iszy)-y0)
rmax=amin1(xmax,ymax)
for([iy=1;rmax=0.] ; iy.le.iszy ; iy=iy+1) [
        do ix=1,iszx [
                rsq=(float(ix)-x0)**2+(float(iy)-y0)**2
                r=sqrt(rsq)
                if(abs(float(ix)-x0).le.xc.and.abs(float(iy)-y0).le.yc)
                        zline(ix)=zmag
                else
                        zline(ix)=(0.,0.)
                rmax=amax1(rmax,abs(real(zline(ix))))
                     ]
        call ouplnz(UNIT_1,i_assoc_var,zline,iy,iszx,zrec)
                                                  ]
ipdb(I_PDB_HORS)=iszx
ipdb(I_PDB_VERS)=iszy
rpdb(R_PDB_MAX_PEL)=rmax
write(UNIT_1%'PDB_RECORD) ipdb
close(unit=UNIT_1)
return
end




subroutine subpic(cstr1,cstr2,cstr3,iszx2,iszy2)        # make subpicture
TP_IMPLICIT
COM_PDB
COM_1_ARRAY
i_old_dim=ipdb(I_PDB_OLD_DIM)
call bopenf(cstr3,UNIT_3,'old')
call rget(x0)              # ( x0,y0 ) is the center of subpic
call rget(y0)
call iprint('output x-dimension is ',iszx2)
call opnfod(cstr1,UNIT_1,i_assoc_var1,iszx1,iszy1)
call opnfnw(cstr2,UNIT_2,i_assoc_var2,iszx2,iszy2)
ix0=ifix(x0)
iy0=ifix(y0)
for([iy1=iy0;iy2=1;rmax=0.] ; iy2.le.iszy2 ; [iy1=iy1+1;iy2=iy2+1]) [
        call inplnz(UNIT_1,i_assoc_var1,zline,iy1,iszx1,zrec)
        call ouplnz(UNIT_2,i_assoc_var2,zline(ix0),iy2,iszx2,zrec)
        do i=ix0,ix0+iszx2-1
                rmax=amax1(rmax,abs(real(zline(i))))
        ]
ipdb(I_PDB_HORS)=iszx2
ipdb(I_PDB_VERS)=iszy2
if(i_old_dim.eq.0) [
        ipdb(I_PDB_OLD_DIM)=iszx2
        ipdb(I_PDB_XOFF)=ix0-1
        ipdb(I_PDB_YOFF)=iy0-1
        ]
else [
```

```
        ipdb(I_PDB_XOFF)=IXO-1
        ipdb(I_PDB_YOFF)=IYO-1
        ]
rpdb(R_PDB_MAX_PEL)=rmax
write(UNIT_2%'PDB_RECORD) ipdb
close(unit=UNIT_1)
close(unit=UNIT_2)
close(unit=UNIT_3)
return
end




subroutine maksr8(cstr, iszx, iszy, rmag, x0, y0, sigma)      # make mexican hat filter
TP_IMPLICIT
COM_1_ARRAY
COM_PDB
do i=1,256
        ipdb(i)=0
call opnfnw(cstr,UNIT_1,i_assoc_var,iszx,iszy)
zmag=cmplx(rmag,0.)
xmax=amin1(x0-1.,float(iszx)-x0)
ymax=amin1(y0-1.,float(iszy)-y0)
rmax=amin1(xmax,ymax)
var_inverse=1./sigma**2
twovar_inverse=1./(2.*sigma**2)
sigma_4_2_pi_inverse=1./(sigma**4*sqrt(2.*pi))
for([iy=1;rmax=0.] ; iy.le.iszy , iy=iy+1) [
        do ix=1,iszx [
                        rsq=(float(ix)-x0)**2+(float(iy)-y0)**2
                r=sqrt(rsq)
                        zline(ix)=cmplx(sigma_4_2_pi_inverse* \
                            (2.-var_inverse*rsq)*exp(-twovar_inverse*rsq),0.)
                rmax=amax1(rmax,abs(real(zline(ix))))
                ]
        call ouplnz(UNIT_1,i_assoc_var,zline,iy,iszx,zrec)
        ]
ipdb(I_PDB_HORS)=iszx
ipdb(I_PDB_VERS)=iszy
rpdb(R_PDB_MAX_PEL)=rmax
write(UNIT_1%'PDB_RECORD) ipdb
close(unit=UNIT_1)
return
end




subroutine rot11(cstr1,cstr2,ix,iy)
# rotate or translate a picture
# z2=circular_integer_translation(z1)
# z1 is cstr1
# z2 is cstr2
# x translation is ix
# y translation is iy
TP_IMPLICIT
```

```
COM_Z_ARRAYS
COM_PDB
COM_PIC_PARAM
call opnfod(cstr1,UNIT_1,i_assoc_var1,iszx1,iszy1)
call opnfnw(cstr2,UNIT_2,i_assoc_var2,iszx1,iszy1)
if(ix.lt.0)
        ix=iszx1+ix
if(iy.lt.0)
        iy=iszy1+iy
for(j=1 ; j.le.iszy1 ; j=j+1) [
        call inplnz(UNIT_1,i_assoc_var1,zline1,j,iszx1,zrec1)
        do i=1,iszx1
                zline2(mod(i-1+ix,iszx1)+1)=zline1(i)
        call ouplnz(UNIT_2,i_assoc_var2,zline2,mod(j-1+iy,iszy1)+1,  \
                   iszx1,zrec2)
        ]
ipdb(I_PDB_XOFF)=0
ipdb(I_PDB_YOFF)=0
write(UNIT_2%'PDB_RECORD) ipdb
close(unit=UNIT_1)
close(unit=UNIT_2)
return
end


subroutine xform(iop,cstr)
# FFT and FFT[-1]
# iop 10 is FFT
# iop 11 is FFT[-1]
TP_IMPLICIT
COM_Z_ARRAYS
COM_PDB
call opnfod(cstr,UNIT_1,i_assoc_var,iszx,iszy)
for([j=1;rmax=0.] ; j.le.iszy ; j=j+1) [
        call inplnz(UNIT_1,i_assoc_var,zline1,j,iszx,zrec1)
        if(iop.eq.10)
                call fft(zline1,iszx,'forward')
        else
                call fft(zline1,iszx,'inverse')
        call ouplnz(UNIT_1,i_assoc_var,zline1,j,iszx,zrec1)
        do i=1,iszx
                rmax=amax1(rmax,abs(real(zline1(i))))
        ]
call transp(UNIT_1,i_assoc_var,iszx)
for([j=1;rmax=0.] ; j.le.iszy ; j=j+1) [
        call inplnz(UNIT_1,i_assoc_var,zline1,j,iszx,zrec1)
        if(iop.eq.10)
                call fft(zline1,iszx,'forward')
        else
                call fft(zline1,iszx,'inverse')
        call ouplnz(UNIT_1,i_assoc_var,zline1,j,iszx,zrec1)
        do i=1,iszx
                rmax=amax1(rmax,abs(real(zline1(i))))
        ]
rpdb(R_PDB_MAX_PEL)=rmax
write(UNIT_1%'PDB_RECORD) ipdb
close(unit=UNIT_1)
return
end


subroutine twofil(cstr1,cstr2,iop,rmin)
# Opens two files as specified by the user  It performs and element by element
```

```
#   operation on the files.
# To avoid divide by 0, when real(z1) is less than rmin, the result is set to
# 0. rmin is defined as a fraction of the maximum value, rmax, in z1.
# iop 20 is z2=z2*z1
# iop 21 is z2=z1*conjg(z2)
# iop 23 is z2=z2/z1
TP_IMPLICIT
COM_Z_ARRAYS
COM_PDB
call opnfod(cstr1,UNIT_1,i_assoc_var1,iszx1,iszy1)
rmax1=rpdb(R_PDB_MAX_PEL)
call opnfod(cstr2,UNIT_2,i_assoc_var2,iszx2,iszy2)
if(iop.eq.23)
        rmin=abs(rmax1*rmin)
i_assoc_var1=DATA_RECORD_1
i_assoc_var2=DATA_RECORD_1
rmax=0.
nrecords=float(iszx2)*float(iszy2)/RECORD_SIZE_8+0.999
do j=1,nrecords [
        read(UNIT_1%'i_assoc_var1) zrec1
        read(UNIT_2%'i_assoc_var2) zrec2
        if(iop.eq.20)
                do i=1,RECORD_SIZE_8
                        zrec2(i)=zrec1(i)*zrec2(i)
        if(iop.eq.21)
                do i=1,RECORD_SIZE_8
                        zrec2(i)=zrec1(i)*conjg(zrec2(i))
        if(iop.eq.23)
                do i=1,RECORD_SIZE_8
                        if(abs(real(zrec1(i))).ge.rmin)
                                zrec2(i)=cmplx(real(zrec2(i))/real(zrec1(i)),0.)
                        else
                                zrec2(i)=(0.,0.)
        write(UNIT_2%'i_assoc_var2-1) zrec2
        do i=1,RECORD_SIZE_8
                rmax=amax1(rmax,abs(real(zrec2(i))))
        ]
rpdb(R_PDB_MAX_PEL)=rmax
write(UNIT_2%'PDB_RECORD) ipdb
close(unit=UNIT_1)
close(unit=UNIT_2)
return
end



subroutine normf(cstr1,cstr2)
TP_IMPLICIT
COM_Z_ARRAYS
COM_PDB
call opnfod(cstr1,UNIT_1,i_assoc_var1,iszx1,iszy1)
call opnfod(cstr2,UNIT_2,i_assoc_var2,iszx2,iszy2)
for([j=1;npel=0;zsum=(0.,0.);sumsq=0.] ; j.le.iszy2 ; j=j+1) [
        call inplnz(UNIT_1,i_assoc_var1,zline1,j,iszx1,zrec1)
        call inplnz(UNIT_2,i_assoc_var2,zline2,j,iszx2,zrec2)
        do i=1,iszx2
                if(real(zline1(i)).ne.0.) [
                        npel=npel+1
                        zsum=zsum+zline2(i)
                        sumsq=sumsq+cabs(zline2(i))
                        ]
        ]
```

```
if(npel.eq.0) [
        call vprint('keisan dekimasen')
        return
        ]
zmean=zsum/npel
rootsq_inverse=1./sqrt(sumsq-cabs(zsum)/npel)
for([j=1;rmax=0.] ; j.le.iszy2 ; j=j+1) [
        call inplnz(UNIT_1,i_assoc_var1,zline1,j,iszx1,zrec1)
        call inplnz(UNIT_2,i_assoc_var2,zline2,j,iszx2,zrec2)
        do i=1,iszx2
                if(real(zline1(i)).ne.0.)
                        zline2(i)=(zline2(i)-zmean)*rootsq_inverse
                else
                        zline2(i)=(0.,0.)
        do i=1,iszx2
                rmax=amax1(rmax,abs(real(zline2(i))))
        call ouplnz(UNIT_2,i_assoc_var2,zline2,j,iszx2,zrec2)
        ]
rpdb(R_PDB_MAX_PEL)=rmax
write(UNIT_2%'PDB_RECORD) ipdb
close(unit=UNIT_1)
close(unit=UNIT_2)
return
end



subroutine onef19(cstr)
TP_IMPLICIT
COM_Z_ARRAYS
COM_PDB
call opnfod(cstr,UNIT_1,i_assoc_var,iszx,iszy)
nrecords=float(iszx)*float(iszy)/RECORD_SIZE_8+.999
rmax=rpdb(R_PDB_MAX_PEL)
i_assoc_var=DATA_RECORD_1
rmax=0.
do j=1,nrecords [
                read(UNIT_1%'i_assoc_var) zrec1
                do i=1,RECORD_SIZE_8
                        zrec1(i)=cmplx(sqrt(amax1(0.,real(zrec1(i)))),0.)
                write(UNIT_1%'i_assoc_var-1) zrec1
                do i=1,RECORD_SIZE_8
                        rmax=amax1(rmax,abs(real(zrec1(i))))
                ]
rpdb(R_PDB_MAX_PEL)=rmax
write(UNIT_1%'PDB_RECORD) ipdb
close(unit=UNIT_1)
return
end



subroutine peak4(cstr1,cstr2,n)
TP_IMPLICIT
real*4 f(3,3),temp(3)
COM_PDB
COM_1_ARRAY
t0(f1,f2,f3)=(f1-f3)/(2.*(f1-2.*f2+f3))
f_of_t(t0,f1,f2,f3)=(f1-2.*f2+f3)/2.*t0**2+(f3-f1)/2.*t0+f2
call opnfod(cstr1,UNIT_1,i_assoc_var,iszx,iszy)
```

```
for([j=1;rx=0.;ry=0.;rn=0.;rmax=-1.E38] ; j.le.iszy ; j=j+1) [
        call inplnz(UNIT_1,i_assoc_var,zline,j,iszx,zrec)
        do i=1,iszx [
                r=real(zline(i))
                if(r.lt.rmax)
                        next
                if(r.eq.rmax) [
                        rn=rn+1.
                        rx=rx+float(i)
                        ry=ry+float(j)
                        next
                              ]
                rmax=r
                rn=1.
                rx=i
                ry=j
                    ]
                                                                         ]
if(rn.eq.0.)
        stop 'No maximum found'
rx=rx/rn
ry=ry/rn
ix=rx+.5
iy=ry+.5
if(rn.gt.1.) [
        call vprint('More than one peak found')
        call rprint('rn = ',rn)
                ]
else[
        for(j=1 ; j.le.3 ; j=j+1) [
                call inplnz(UNIT_1,i_assoc_var,zline, \
                        mod(iy+(j-2)-1+iszy,iszy)+1,iszx,zrec)
                do i=1,3 [
                        ii=mod(ix+(i-2)-1+iszx,iszx)+1
                        f(i,j)=real(zline(ii))
                         ]
                                      ]
        for([iters=1;t0x=0.;t0y=0.] ; iters.le.3 ; iters=iters+1) [
                do i=1,3
                        temp(i)=f_of_t(t0y,f(i,1),f(i,2),f(i,3))
                t0x=t0(temp(1),temp(2),temp(3))
                do i=1,3
                        temp(i)=f_of_t(t0x,f(1,i),f(2,i),f(3,i))
                t0y=t0(temp(1),temp(2),temp(3))
                                                                          ]
        rmax=temp(2)
        call rprint('x interpolation is ',t0x)
        call rprint('y interpolation is ',t0y)
        rx=rx+t0x
        ry=ry+t0y
    ]
if(ifix(rx).gt.iszx/2)
        rx=rx-float(iszx)
if(ifix(ry).gt.iszy/2)
        ry=ry-float(iszy)
rx=rx+float(ipdb(I_PDB_XOFF)).
ry=ry+float(ipdb(I_PDB_YOFF))
call rprint('Rmax is ',rmax)
call rprint('The x position is ',rx)
call rprint('The y position is ',ry)
call bopenp(cstr2,UNIT_2,3*(n-1))
call rput('',rx)
call rput('',ry)
```

```
call rput('',rn)
close(unit=UNIT_2)
close(unit=UNIT_1)
return
end



subroutine ctrom(cstr1,cstr2,cstr3,n)            # cstr3
# find center of mass
TP_IMPLICIT
COM_1_ARRAY
COM_PDB
call opnfod(cstr1,UNIT_1,i_assoc_var,iszx,iszy)
for([j=1;rx=0.;ry=0.;f=0.] ; j.le.iszy ;  j=j+1) [
        call inplnz(UNIT_1,i_assoc_var,zline,j,iszx,zrec)
        do i=1,iszx [
                r=real(zline(i))
                f=f+r
                rx=rx+r*float(i)
                ry=ry+r*float(j)
                ]
        ]
rx=rx/f
ry=ry/f
rx=rx+float(ipdb(I_PDB_XOFF))
ry=ry+float(ipdb(I_PDB_YOFF))
call rprint('The x center is ',rx)
call rprint('The y center is ',ry)
close(unit=UNIT_1)
call bopenf(cstr3,UNIT_1,'old')              ###
call rget(x1)                                ###
call rget(y1)                                ###
close(unit=UNIT_1)                           ###
call bopenp(cstr2,UNIT_2,3*(n-1))
call rput('',x1)                             ###
call rput('',y1)                             ###
call rput('',f)
close(unit=UNIT_2)
return
end



subroutine calc20(cstr,n1,n2,n3)
TP_IMPLICIT
CHARACTER cstr(81)
call bopenf(cstr,UNIT_1,'old')
for(i=1 ; i.le.3*(n1-1) ; i=i+1)
        call rget(r)
call rget(x2a)
call rget(y2a)
call rget(r)
call rget(x1a)
call rget(y1a)
call rget(r)
close(unit=UNIT_1)
call bopenf(cstr,UNIT_1,'old')
for(i=1 ; i.le.3*(n2-1) ; i=i+1)
        call rget(r)
```

```
call rget(x2b)
call rget(y2b)
call rget(r)
call rget(x1b)
call rget(y1b)
call rget(r)
close(unit=UNIT_1)
theta1=atan2(y1a-y1b, x1a-x1b)
theta2=atan2(y2a-y2b, x2a-x2b)
theta=theta1-theta2

call rprint('The angle (radians) is ', theta)
call rprint('The angle (degrees) is ', theta*180 /PI)
call bopenp(cstr, UNIT_1, 3*(n3-1))
call rput('', theta)
call rput('', theta*180. /PI)
call rput('', 0. )
close(unit=UNIT_1)
```

```
/******************************************************************************
*       PROGRAM YOS3.RAT.
*       Ocular Counter Rolling Small Subroutine Program
*       15-Aug-84
*       Yoshihiro Nagashima
*               Direct program file to find the rotation between two pictures
*               ( 1 and 2 ) from two correlations. Picture 1 is masked by
*               ROI-A and ROI-B, and then correlated with picture 2. From
*               the posions of the two regions of interest and the translation
*               from picture 1 to picture 2, the rotation is calculated.
*               The basic idea is that we are tracing two points on the eye.
*               With large rotation, the correlation will be affected by the
*               rotation of the data in the region of interest. Subpictures
*               are extracted from the two pictures centered around the region
*               of interest and the subpictures are used during correlation.
*
*       Inputs: ocr1.pic        raw picture 256*256 dimension
*               ocr2.pic        raw picture 256*256 dimension
*
*       Outputs:ocr1.pic
*               ocr2.pic
*               ocr1.z
*               ocr2.z
*               a1.z            subpicture A1 extracted from ocr1.z
*               a2.z            subpicture A2 extracted from ocr2.z
*               aa2.z           a2.z * a2.z
*               b1.z            subpicture B1 extracted from ocr1.z
*               b2.z            subpicture B2 extracted from ocr2.z
*               bb2.z           b2.z * b2.z
*               a.dat           x-y center data of subpicture A
*               b.dat           x-y center data of subpicture B
*               mh.z            Mexican Hat Filter
*               mhf.z           FFT of Mexican Hat Filter
*               msk.z           mask of MHF
*               one.z           mask data
*               one.z1          mask data
*               one.z2          mask data
*               ocrbox.dat      the last three data are theta ( radian ),
*                               theta ( degree ) and zero.
*       Caution:When you define the location of subpictures, please don't use
*               return key. Return key will make an error. Use space key, etc.
*
******************************************************************************/
label NO_FILE,NO_FILE_2
define EOS1             0       # end of string 1
define EOS2             128
define VSV_LINES        25
define VSV_ADDR         172600
define BIT_MAP1         172620
define BIT_MAP2         172640
define CR               13
define LF               10
define SPACE            32
define CURSER_DISABLE   2000
define JSW              44      # address of job status word
define TT_SPECIAL_MODE  10100   # special mode bits in jsw
define DEL              127
define NULL             0
define MAX_ENTRIES      30
define FMT_STAT         format(g15.7)
define MAX_LINE         256     # Max picture dimension (pels/line)
define MAX_2_LINE       512     # Twice maximum picture dimension
```

```
define MAX_LINE_PLUS_1   257       # Maximun line plus 1
define RECORD_SIZE       512       # Largely determined by PDB size
define RECORD_SIZE_4     128       # Real*4 size
define RECORD_SIZE_8     64        # Complex*8 size
define PDB_RECORD        1         # Record number of picture descriptor block
define I_PDB_HORS        15        # Index of integer horizontal dimension
define I_PDB_VERS        16        # Index of integer vertical dimension
define I_PDB_XOFF        251       # Index of x offset in parent picture
define I_PDB_YOFF        252       # Index of y offset in parent picture
define I_PDB_OLD_DIM     250       # Index of demension of parent picture
define C_PDB_FORM        7         # Byte index in PDB of format flags
define PDB_FORM_WR       16        # Wrong reading bit in format byte i.e. (y,x)
define DATA_RECORD_1     2         # Record number of start of descriptor
define R_PDB_MAX_PEL     128       # Real index in PDB of max pel
define UNIT_1            1         # Logical unit for file 1
define UNIT_2            2         # Logical unit for file 2
define UNIT_3            3         # Logical unit for file 3
define PI                3.14159265358979
define VIRTUAL_BUFFERS   4         # Buffers equals blocks per line
define TP_IMPLICIT implicit byte(b-c),integer*2(i-n),real*4(a,e-h,o-y),\
                                real*8(d),complex*8(z)
define CHARACTER         byte
define COM_VSV           common /vsvc/line_num,bhold,b_no_vsv
define COM_TMP_STR       CHARACTER c_tmp_str(81); \
                         common /tmpstc/c_tmp_str
define COM_ASK           common /askc/idbase,b_indirect_command
define COM_SAVE_FILE     CHARACTER cstr2(81);\
                         common /sfc/lun,cstr2
define COM_OUTPUT        common /oupc/iunito,i_record_o,iptro,b_zero_pad
define COM_INPUT         common /inpc/iuniti,i_record_i,iptri,n_records_i
define COM_SIZE          common /szc/inpsz,ioupsz,ciform,coform, \
                                    inp_oup_ratio,inp_oup_ratio_sq
define COM_ZREC_O        complex*8 zreco(RECORD_SIZE_8); \
                         common /zrecoc/zreco
define COM_PDB           CHARACTER cpdb(512); \
                         integer*2 ipdb(256); \
                         real*4 rpdb(128); \
                         equivalence (cpdb,ipdb,rpdb); \
                         common /pdbc/ipdb
define COM_ARRAYS        CHARACTER cline1(MAX_LINE); \
                         integer*2 line1(MAX_LINE),line2(MAX_LINE); \
                         complex*8 zline1(MAX_LINE); \
                         equivalence (cline1,line1,zline1); \
                         equivalence (line2(MAX_LINE),zline1(MAX_LINE)); \
                         common /arrayc/zline1
# |<--------------------------------zline1--------------------------------->|
# |<------line1------>|                                     |<-------line2------->|
# |<-cline1->|
define COM_Z_ARRAYS      complex*8 zline(MAX_2_LINE); \
                         complex*8 zline1(MAX_LINE),zline2(MAX_LINE); \
                         complex*8 zbuff(RECORD_SIZE_8,VIRTUAL_BUFFERS); \
                         complex*8 zrec1(RECORD_SIZE_8),zrec2(RECORD_SIZE_8); \
                         real*4 rline1(MAX_2_LINE),rline2(MAX_2_LINE); \
                         CHARACTER cline1(MAX_LINE),cline2(MAX_LINE); \
                         CHARACTER crec1(RECORD_SIZE),crec2(RECORD_SIZE); \
                         equivalence (zline,zline1,rline1,zbuff); \
                         equivalence (zline(MAX_LINE_PLUS_1),zline2,rline2); \
                         equivalence (zrec1,crec1),(zrec2,crec2); \
                         common /zc/zrec1,zline,zrec2
define COM_1_ARRAY       complex*8 zline(MAX_LINE),zrec(RECORD_SIZE_8); \
                         complex*8 zbuff(RECORD_SIZE_8,VIRTUAL_BUFFERS); \
                         real*4 rline(MAX_2_LINE); \
                         CHARACTER cline(MAX_LINE),crec(RECORD_SIZE); \
```

```
                           equivalence (zline,rline,cline,zbuff); \
                           equivalence (zrec,crec); \
                           common /zc/zrec,zline
define COM_PIC_PARAM       equivalence (iszx,iszx1),(iszy,iszy1); \
                           common /ppramc/iszx1,iszy1,iszx2,iszy2,scalex,scaley, \
                                        unscalex,unscaley
define COM_GAM             common /gamc/iy,iframe,igamsz,i_oup_gam_ratio,b_byte_mtx
define ouplnz_sr           olnzsr
define ouplnz_eq_O         ouplnO


##########################################################################
# other small subroutines
##########################################################################


subroutine copy(cstr1,cstr2,n)
dimension  x(128)
open(unit=1,access='DIRECT',recordsize=128,type='OLD',\
        name=cstr1)
open(unit=2,access='DIRECT',recordsize=128,type='NEW',\
        name=cstr2)
do i=1,n [
        read(1%'i) x
        write(2%'i) x
        ]
close(unit=1)
close(unit=2)
return
end

subroutine opnfnw(cstr,iunit,i_assoc_var,iszx,iszy)      # open new file
TP_IMPLICIT
COM_PDB
CHARACTER c_prompt(81)
CHARACTER cstr(81)
i=INDEX(cstr,'.')
init_size=DATA_RECORD_1-1+ \
                float(iszx)*float(iszy)/RECORD_SIZE_8+0.999
NO_FILE_2       open(unit=iunit,name=cstr,type='NEW',access='DIRECT', \
                      recordsize=RECORD_SIZE_4,initialsize=init_size, \
                      associatevariable=i_assoc_var,buffercount=2)
return
end




subroutine opnfod(cstr,iunit,i_assoc_var,iszx,iszy)

TP_IMPLICIT
COM_PDB
CHARACTER c_prompt(81)
CHARACTER cstr(81)
i=INDEX(cstr,'.')
open(unit=iunit,name=cstr,type='OLD',access='DIRECT', \
```

```
     recordsize=RECORD_SIZE_4, associatevariable=i_assoc_var,  \
     buffercount=2)
read(iunit%'PDB_RECORD) ipdb
iszx=ipdb(I_PDB_HORS)
iszy=ipdb(I_PDB_VERS)
return
end



subroutine vsvfig(iunit)
TP_IMPLICIT
COM_VSV
COM_TMP_STR
string c_prompt 'Is there a vsv01 ? \200'
%C
open(unit=iunit, name='SY:VSVFIG.DAT', type='OLD', carriagecontrol='LIST', \
                                                   err=NO_FILE)
read(iunit, 1) b_no_vsv
1 format(l1)
if(.false.) [
NO_FILE call GTLIN(c_tmp_str, c_prompt)
        if(cup(c_tmp_str(1)).eq.'Y')
                b_no_vsv=.false.
        else
                b_no_vsv=.true.
        open(unit=iunit, name='VSVFIG.DAT', type='NEW', carriagecontrol='LIST')
        write(iunit, 1) b_no_vsv
        ]
close(unit=iunit)
return
end


logical function cup*1(char)
TP_IMPLICIT
%C
if('a'.le.char.and.char.le.'z')
        cup=char-32
else
        cup=char
return
end


logical function beqflg*1(cstr1, cstr2)
TP_IMPLICIT
CHARACTER cstr1(2), cstr2(2)
%C
if(cup(cstr1(1)).eq.cup(cstr2(1)).and.cup(cstr1(2)).eq.cup(cstr2(2)))
        beqflg=.true.
else
        beqflg=.false.
return
end


subroutine vsvclr(iwhat)
TP_IMPLICIT
COM_VSV
```

```
%C
if(beqflg(iwhat,'vsv'))
        b_no_vsv=.false.
if(beqflg(iwhat,'no vsv'))
        b_no_vsv=.true.
if(beqflg(iwhat,'hold'))
        bhold=.true.
if(beqflg(iwhat,'continue'))
        bhold=.false.
# Clear characters and curser
if(beqflg(iwhat,'all').or.beqflg(iwhat,'characters')) [
        line_num=1
        if(b_no_vsv) [
                call PRINT(EOS1)
                call PRINT(EOS1)
                return
                ]
        call IPOKE(%"VSV_ADDR+4,0)                    # upper left
        do i=1,25 [
                do j=1,64
                        call IPOKEB(%"VSV_ADDR,SPACE)
                call IPOKEB(%"VSV_ADDR,CR)
                call IPOKEB(%"VSV_ADDR,LF)
                ]
        call IPOKE(%"VSV_ADDR,%"CURSER_DISABLE)
        call IPOKE(%"VSV_ADDR+4,0)                    # upper left
        ]
if(b_no_vsv)
        return
# Turn cursor on or off
if(beqflg(iwhat,'on'))
        call IPOKE(%"VSV_ADDR,0)
if(beqflg(iwhat,'off'))
        call IPOKE(%"VSV_ADDR,%"CURSER_DISABLE)
# Turn off bit maps
if(beqflg(iwhat,'all').or.beqflg(iwhat,'bit maps')) [
        call IPOKE(%"BIT_MAP1,IPEEK(%"BIT_MAP1).and..not.%"400)
        call IPOKE(%"BIT_MAP2,IPEEK(%"BIT_MAP2).and..not.%"400)
        ]
return
end



subroutine vsvset(line,icol)
TP_IMPLICIT
COM VSV
%C
line_num=line
if(b_no_vsv)
        return
call IPOKE(%"VSV_ADDR+4,256*(line-1)+icol-1)
return
end


subroutine cprint(cstr,cval,n)
TP_IMPLICIT
CHARACTER cstr(81),cval(81)
COM_TMP_STR
%C
for(k=1 ; k.le.80.and.cstr(k).ne.EOS1.and.cstr(k).ne.EOS2 ; k=k+1)
```

```
        c_tmp_str(k)=cstr(k)
c_tmp_str(k)=EOS2
call vprint(c_tmp_str)
for(i=1 ; i.le.n ; i=i+1)
        if(cval(i).ne.' ')
                break
for(j=1 ; i.le.n ; [i=i+1;j=j+1])
        c_tmp_str(j)=cval(i)
c_tmp_str(j)=cstr(k)
call vprint(c_tmp_str)
return
end



subroutine rprint(cstr,rval)
TP_IMPLICIT
CHARACTER cstr(81)
CHARACTER crval(15)
%C
encode(15,1,crval) rval
1 format(g15.7,$)
for(i=15 ; i.gt.1.and.(crval(i).eq.SPACE.or.crval(i).eq.'0') ; )
        i=i-1
call cprint(cstr,crval,i)
return
end



subroutine vprint(cstr)
TP_IMPLICIT
CHARACTER cstr(81)
COM_VSV
data b_no_vsv/.false./,bhold/.false./
%C
for([i=1;n=0] ; i.le.81 ; i=i+1) [
        if(cstr(i).eq.EOS2)
                break
        if(cstr(i).eq.EOS1) [
                n=n+1
                break
                ]
        if(cstr(i).eq.LF)
                n=n+1
        ]
if(line_num.gt.VSV_LINES.or.line_num+n.ge.VSV_LINES+2) [
        if(bhold) [
                i_oldjsw=IPEEK(%"JSW)
                call IPOKE(%"JSW,%"TT_SPECIAL_MODE.or.i_oldjsw)
                call vsvset(25,62)
                call vsvclr('on')
                repeat [
                        ]until(ITTINR().eq.CR)
                call vsvclr('off')
                i=ITTINR()      # Dump trailing <LF>
                call IPOKE(%"JSW,i_oldjsw)
                ]
        else
                call vsvclr('characters')
        call vsvset(1,1)        # line_num=1 as side effect
        ]
line_num=line_num+n
```

```
if(b_no_vsv) [
        call PRINT(cstr)
        return
        ]
for(i=1 ; i.le.80.and.cstr(i).ne.EOS1.and.cstr(i).ne.EOS2 ; i=i+1)
        call IPOKEB(%"VSV_ADDR,cstr(i))
if(cstr(i).eq.EOS1) [             # for 0 end of string, add <CR><LF>
        call IPOKEB(%"VSV_ADDR,CR)
        call IPOKEB(%"VSV_ADDR,LF)
        ]
return
end



subroutine ask(cstr,c_prompt)
TP_IMPLICIT
CHARACTER cstr(81),c_prompt(81)
COM_ASK
COM_TMP_STR
string crlf '\n\200'
# Simulate a delete with a backspace, space, backspace
string cdel '\b \b\200'
data idbase/10/,b_indirect_command/.true./
%C
call RCTRLO      # resets ^O
call z200(c_prompt,c_tmp_str)
# Presumably if ISPY(%"366) is negative, an indirect file is active; however,
#   this is very poorly documented in program request section of advanced
#   programers manuel.
if(b_indirect_command.and.ISPY(%"366).lt.0) [
        call GTLIN(cstr,c_tmp_str)              # get input from indirect file
        return
        ]
call vprint(c_tmp_str)
call vsvclr('on')                 # turn on curser
i_oldjsw=IPEEK(%"JSW)             # save old jsw
# TT special mode with immediate return
call IPOKE(%"JSW,%"TT_SPECIAL_MODE.or.i_oldjsw)
for(j=1 ; j.le.80 ; j=j+1) [
        repeat [                  # get past deletes and not ready
                i=ITTINR()
                if(i.eq.DEL.and.j.gt.1) [                 # delete?
                        j=j-1
                        call vprint(cdel)
                        ]
                ]until(i.gt.0.and.i.ne.DEL)
        if(i.eq.CR)               # CR ends input
                break
        else [
                cstr(j)=i
                cstr(j+1)=EOS2
                call vprint(cstr(j))
                ]
        ]
cstr(j)=EOS1
call vprint(crlf)                 # newline
call vsvclr('off')                # turn off curser
i=ITTINR()              # dump trailing <LF>
call IPOKE(%"JSW,i_oldjsw)                # restore old TT mode
return
end
```

```
subroutine z200(cstr1,cstr2)
TP_IMPLICIT
CHARACTER cstr1(81),cstr2(81)
%C
for(i=1 ; i.le.80.and.cstr1(i).ne.EOS1.and.cstr1(i).ne.EOS2 ; i=i+1)
        cstr2(i)=cstr1(i)
if(cstr1(i).eq.EOS2)
        cstr2(i)=EOS1
else
        cstr2(i)=EOS2
return
end



logical function bopenf*1(cstr,iunit,ctype)
TP_IMPLICIT
CHARACTER cstr(81),ctype(81)
COM_SAVE_FILE

lun=iunit
if(cstr(1).eq.NULL)
        call ask(cstr2,'File name ? ')
else
        call SCOPY(cstr,cstr2,80)
if(INDEX(cstr2,'.').eq.0)
        call CONCAT(cstr2,'.DAT',cstr2,80)
if(beqflg(ctype,'new'))
        open(unit=lun,name=cstr2,type='NEW',carriagecontrol='LIST')
if(beqflg(ctype,'old')) [
        open(unit=lun,name=cstr2,type='OLD',carriagecontrol='LIST',err=1)
        if(.false.) [
                1 bopenf=.false.
                return
                ]
        ]
if(beqflg(ctype,'unknown'))
        open(unit=lun,name=cstr2,type='UNKNOWN',carriagecontrol='LIST')
bopenf=.true.
return
end



logical function bopenp*1(cstr,iunit,n)
TP_IMPLICIT
CHARACTER cstr(81)
real*4 r(MAX_ENTRIES)
COM_SAVE_FILE

lun=iunit
if(cstr(1).eq.NULL)
        call ask(cstr2,'File name ? ')
else
        call SCOPY(cstr,cstr2,80)
if(INDEX(cstr2,'.').eq.0)
        call CONCAT(cstr2,'.DAT',cstr2,80)
if(n.gt.1) [
        open(unit=lun,name=cstr2,type='OLD',carriagecontrol='LIST',err=1)
```

```
        if(.false.) [
                1 bopenp=.false.
                return
                ]
        for(i=1 ; i.le.min0(n,MAX_ENTRIES) ; i=i+1)
                call rget(r(i))
        close(unit=lun)
        ]
open(unit=lun,name=cstr2,type='NEW',carriagecontrol='LIST')
for(i=1 ; i.le.min0(n,MAX_ENTRIES) ; i=i+1)
        call rput('',r(i))
bopenp=.true.
return
end


subroutine iprint(cstr,ival)
TP_IMPLICIT
CHARACTER cstr(81)
CHARACTER cival(6)
%C
encode(6,1,cival) ival
1 format(i6,$)
call cprint(cstr,cival,6)
return
end


subroutine oprint(cstr,ival)
TP_IMPLICIT
CHARACTER cstr(81)
CHARACTER coval(6)
%C
encode(6,1,coval) ival
1 format(o6,$)
call cprint(cstr,coval,6)
return
```

```
/******************************************************************************
*       PROGRAM YOS4.RAT
*       Ocular Counter Rolling Small Subroutine Program
*       15-Aug-84
*       Yoshihiro Nagashima
*               Direct program file to find the rotation between two pictures
*               ( 1 and 2 ) from two correlations. Picture 1 is masked by
*               ROI-A and ROI-B, and then correlated with picture 2. From
*               the posions of the two regions of interest and the translation
*               from picture 1 to picture 2, the rotation is calculated.
*               The basic idea is that we are tracing two points on the eye.
*               With large rotation, the correlation will be affected by the
*               rotation of the data in the region of interest. Subpictures
*               are extracted from the two pictures centered around the region
*               of interest and the subpictures are used during correlation.
*
*       Inputs: ocr1.pic        raw picture 256*256 dimension
*               ocr2.pic        raw picture 256*256 dimension
*
*       Outputs ocr1.pic
*               ocr2.pic
*               ocr1.z
*               ocr2.z
*               a1.z            subpicture A1 extracted from ocr1.z
*               a2.z            subpicture A2 extracted from ocr2.z
*               aa2.z           a2.z * a2.z
*               b1.z            subpicture B1 extracted from ocr1.z
*               b2.z            subpicture B2 extracted from ocr2.z
*               bb2.z           b2.z * b2.z
*               a.dat           x-y center data of subpicture A
*               b.dat           x-y center data of subpicture B
*               mh.z            Mexican Hat Filter
*               mhf.z           FFT of Mexican Hat Filter
*               msk.z           mask of MHF
*               one.z           mask data
*               one.z1          mask data
*               one.z2          mask data
*               ocrbox.dat      the last three data are theta ( radian ),
*                               theta ( degree ) and zero.
*       Caution:When you define the location of subpictures, please don't use
*               return key. Return key will make an error. Use space key, etc.
*
******************************************************************************/
label NO_FILE,NO_FILE_2
define EOS1                 0       # end of string 1
define EOS2                 128
define VSV_LINES            25
define VSV_ADDR             172600
define BIT_MAP1             172620
define BIT_MAP2             172640
define CR                   13
define LF                   10
define SPACE                32
define CURSER_DISABLE       2000
define JSW                  44      # address of job status word
define TT_SPECIAL_MODE      10100   # special mode bits in jsw
define DEL                  127
define NULL                 0
define MAX_ENTRIES          30
define FMT_STAT             format(g15.7)
define MAX_LINE             256     # Max picture dimension (pels/line)
define MAX_2_LINE           512     # Twice maximum picture dimension
```

```
define MAX_LINE_PLUS_1   257      # Maximun line plus 1
define RECORD_SIZE       512      # Largely determined by PDB size
define RECORD_SIZE_4     128      # Real*4 size
define RECORD_SIZE_8     64       # Complex*8 size
define PDB_RECORD        1        # Record number of picture descriptor block
define I_PDB_HORS        15       # Index of integer horizontal dimension
define I_PDB_VERS        16       # Index of integer vertical dimension
define I_PDB_XOFF        251      # Index of x offset in parent picture
define I_PDB_YOFF        252      # Index of y offset in parent picture
define I_PDB_OLD_DIM     250      # Index of demension of parent picture
define C_PDB_FORM        7        # Byte index in PDB of format flags
define PDB_FORM_WR       16       # Wrong reading bit in format byte i.e (y,x)
define DATA_RECORD_1     2        # Record number of start of descriptor
define R_PDB_MAX_PEL     128      # Real index in PDB of max pel
define UNIT_1            1        # Logical unit for file 1
define UNIT_2            2        # Logical unit for file 2
define UNIT_3            3        # Logical unit for file 3
define PI                3.14159265358979
define VIRTUAL_BUFFERS   4        # Buffers equals blocks per line
define TP_IMPLICIT implicit byte(b-c),integer*2(i-n),real*4(a,e-h,o-y), \
                         real*8(d),complex*8(z)
define CHARACTER         byte
define COM_VSV           common /vsvc/line_num,bhold,b_no_vsv
define COM_TMP_STR       CHARACTER c_tmp_str(81); \
                         common /tmpstc/c_tmp_str
define COM_ASK           common /askc/idbase,b_indirect_command
define COM_SAVE_FILE     CHARACTER cstr2(81);\
                         common /sfc/lun,cstr2
define COM_OUTPUT        common /oupc/iunito,i_record_o,iptro,b_zero_pad
define COM_INPUT         common /inpc/iuniti,i_record_i,iptri,n_records_i
define COM_SIZE          common /szc/inpsz,ioupsz,ciform,coform, \
                                  inp_oup_ratio,inp_oup_ratio_sq
define COM_ZREC_O        complex*8 zreco(RECORD_SIZE_8), \
                         common /zrecoc/zreco
define COM_PDB           CHARACTER cpdb(512); \
                         integer*2 ipdb(256); \
                         real*4 rpdb(128); \
                         equivalence (cpdb,ipdb,rpdb); \
                         common /pdbc/ipdb
define COM_ARRAYS        CHARACTER cline1(MAX_LINE), \
                         integer*2 line1(MAX_LINE),line2(MAX_LINE); \
                         complex*8 zline1(MAX_LINE); \
                         equivalence (cline1,line1,zline1), \
                         equivalence (line2(MAX_LINE),zline1(MAX_LINE)); \
                         common /arrayc/zline1
# |<-------------------------------zline1---------------------------------->|
# |<------line1------->|                                 |<--------line2------>|
# |<-cline1->|
define COM_Z_ARRAYS      complex*8 zline(MAX_2_LINE); \
                         complex*8 zline1(MAX_LINE),zline2(MAX_LINE); \
                         complex*8 zbuff(RECORD_SIZE_8,VIRTUAL_BUFFERS); \
                         complex*8 zrec1(RECORD_SIZE_8),zrec2(RECORD_SIZE_8); \
                         real*4 rline1(MAX_2_LINE),rline2(MAX_2_LINE); \
                         CHARACTER cline1(MAX_LINE),cline2(MAX_LINE); \
                         CHARACTER crec1(RECORD_SIZE),crec2(RECORD_SIZE); \
                         equivalence (zline,zline1,rline1,zbuff); \
                         equivalence (zline(MAX_LINE_PLUS_1),zline2,rline2); \
                         equivalence (zrec1,crec1),(zrec2,crec2); \
                         common /zc/zrec1,zline,zrec2
define COM_1_ARRAY       complex*8 zline(MAX_LINE),zrec(RECORD_SIZE_8); \
                         complex*8 zbuff(RECORD_SIZE_8,VIRTUAL_BUFFERS); \
                         real*4 rline(MAX_2_LINE); \
                         CHARACTER cline(MAX_LINE),crec(RECORD_SIZE); \
```

```
                                equivalence (zline,rline,cline,zbuff); \
                                equivalence (zrec,crec); \
                                common /zc/zrec,zline
define COM_PIC_PARAM            equivalence (iszx,iszx1),(iszy,iszy1); \
                                common /ppramc/iszx1,iszy1,iszx2,iszy2,scalex,scaley, \
                                                unscalex,unscaley
define COM_GAM                  common /gamc/iy,iframe,igamsz,i_oup_gam_ratio,b_byte_mtx
define ouplnz_sr                olnzsr
define ouplnz_eq_0              ouplnO


##########################################################################
# other small subroutines
##########################################################################


subroutine rput(cstr,r)
TP_IMPLICIT
CHARACTER cstr(81)
COM_SAVE_FILE

if(cstr(1).ne.NULL)
        call rprint(cstr,r)
write(lun,1) r
1 FMT_STAT
return
end


real function rget*4(r)
TP_IMPLICIT
COM_SAVE_FILE

read(lun,1) r
1 FMT_STAT
rget=r
return
end


subroutine czcvt
TP_IMPLICIT
COM_SIZE
COM_PDB
COM_ARRAYS

max=0
do j=1,ioupsz [
        call getlnc
        do i=1,ioupsz [
                zline1(i)=cmplx(float(line2(i)),0.)
                max=max0(max,line2(i))
                ]
        call oplnz1(zline1)
        ]
```

```
rpdb(R_PDB_MAX_PEL)=float(max)
return
end


subroutine getlnc

# Gets inp_oup_ratio, character (8 bit) lines from input file and compresses
#    them into one line.
# Returns compressed line as an integer*2 line in vector, line2  in COM_ARRAYS.

TF_IMPLICIT
COM_SIZE
COM_ARRAYS

do i=1,inpsz
        line2(i)=0
do i=1,inp_oup_ratio
        call inplnc(line2)
if(inp_oup_ratio.gt.1) [
        for(i=1;i2=1] ; i le ioupsz ; [i=i+1;i2=i2+1]) [
                line2(i)=line2(i2)
                do k=2,inp_oup_ratio [
                        i2=i2+1
                        line2(i)=line2(i)+line2(i2)
                        ]
                ]
        ]
return
end


subroutine inpset(cstr)
TF_IMPLICIT
COM_INPUT
COM_PDB
COM_SIZE
COM_GAM
CHARACTER cstr(31)
data iuniti/UNIT_1/

        ciform=cform(cstr)

        open(unit=iuniti,name=cstr,type='OLD',access='DIRECT',buffercount=2, \
                associatevariable=i_record_i,recordsize=RECORD_SIZE_4)
        read(iuniti'PDB_RECORD) ipdb
        inpsz=ipdb(I_PDB_HOPS)

        if(ciform.eq.'Z')
                n_records_i=DATA_RECORD_1-1+ \
                        float(inpsz)*float(ipdb(I_PDB_VERS))/RECORD_SIZE_8+0.999
        else
                n_records_i=DATA_RECORD_1-1+ \
                        float(inpsz)*float(ipdb(I_PDB_VERS))/RECORD_SIZE+0.999
        i_record_i=DATA_RECORD_1
        iptri=RECORD_SIZE+1             # Indicates record buffer is empty
return
end
```

```
subroutine inplnc(line2)

# crec buffers one record.  RECORD_SIZE and MAX_LINE can be <, >, =.

TP_IMPLICIT
COM_INPUT
COM_SIZE
integer*2 line2(MAX_LINE)
CHARACTER crec(RECORD_SIZE)
equivalence (c,ic)
data ic/0/                # Clear high byte

do i=1,inpsz [
        if(iptri.gt.RECORD_SIZE) [
                if(i_record_i.gt.n_records_i)    # Hide short y dimension
                        return
                read(iunit1%'i_record_i) crec
                iptri=1
                ]
        c=crec(iptri)                   # Treat bytes as 8 bit positive numbers
        line2(i)=line2(i)+ic
        iptri=iptri+1
        ]
return
end


subroutine oupset(cstr)
TP_IMPLICIT
COM_OUTPUT
COM_SIZE
COM_PDB
COM_GAM
CHARACTER cstr(81)
data iunito/UNIT_2/

ioform=cform(cstr)

# Special case VD., VE., VF., VG., and VH.
i=INDEX(cstr,'V')
is=DATA_RECORD_1-1+float(ioupsz)**2/RECORD_SIZE_B+0.999
open(unit=iunito,name=cstr,type='NEW',access='DIRECT', \
        recordsize=RECORD_SIZE_4,associatevariable=i_record_o, \
        buffercount=2,initialsize=is)
i_record_o=DATA_RECORD_1
iptro=1
ipdb(I_PDB_XOFF)=0
ipdb(I_PDB_YOFF)=0
ipdb(I_PDB_HORS)=ioupsz          # Put actual size in PDB
ipdb(I_PDB_VERS)=ioupsz

if(ioupsz.eq.2*inpsz) [
        b_zero_pad=.true.
        ioupsz=inpsz             # Hide the zero padding
        ]
else
        b_zero_pad=.false.
inp_oup_ratio=inpsz/ioupsz
inp_oup_ratio_sq=inp_oup_ratio**2
return
end
```

```
subroutine oplnzl(zline1)
TP_IMPLICIT
COM_OUTPUT
complex*8 zline1(MAX_LINE)

call ouplnz_sr(zline1)
if(b_zero_pad)
        call ouplnz_eq_O
return
end


subroutine ouplnz_sr(zline1)
TP_IMPLICIT
COM_SIZE
COM_OUTPUT
COM_ZREC_O
complex*8 zline1(MAX_LINE)

do i=1,ioupsz [
        zreco(iptro)=zline1(i)
        iptro=iptro+1
        if(iptro.gt.RECORD_SIZE_8) [
                write(iunito%'i_record_o) zreco
                iptro=1
                ]
        ]
return
end


subroutine ouplnz_eq_O
TP_IMPLICIT
COM_SIZE
COM_OUTPUT
COM_ZREC_O

do i=1,ioupsz [
        zreco(iptro)=(O.,O.)
        iptro=iptro+1
        if(iptro.gt.RECORD_SIZE_8) [
                write(iunito%'i_record_o) zreco
                iptro=1
                ]
        ]
return
end


logical function cform*l(cstr)
TP_IMPLICIT
CHARACTER cstr(81)

i=INDEX(cstr,'.')                  # Look for extension
if(i.eq.O)
```

```
        cform=' '                   # No extension
else
        cform=cstr(i+1)
return
end


subroutine ouppdb

# Clean up stuff

TP_IMPLICIT
COM_SIZE
COM_INPUT
COM_OUTPUT
COM_PDB

if(b_zero_pad)          # Only if complex
        do i=1,2*ioupsz
                call ouplnz_eq_0
close(unit=iuniti)
write(iunito%'PDB_RECORD) ipdb
close(unit=iunito)
return
end


subroutine inplnz(iunit,i_assoc_var,zline,iy,isz,zrec)
TP_IMPLICIT
complex*8 zline(MAX_LINE),zrec(RECORD_SIZE_8)

i_assoc_var=DATA_RECORD_1+(isz/RECORD_SIZE_8)*(iy-1)
iptr=RECORD_SIZE_8+1            # Read first record
do i=1,isz [
        if(iptr gt RECORD_SIZE_8) [
                read(iunit%'i_assoc_var) zrec
                iptr=1
                ]
        zline(i)=zrec(iptr)
        iptr=iptr+1
        ]
return
end


subroutine ouplnz(iunit,i_assoc_var,zline,iy,isz,zrec)
TP_IMPLICIT
complex*8 zline(MAX_LINE),zrec(RECORD_SIZE_8)
i_assoc_var=DATA_RECORD_1+(isz/RECORD_SIZE_8)*(iy-1)
iptr=1
do i=1,isz [
        zrec(iptr)=zline(i)
        iptr=iptr+1
        if(iptr gt RECORD_SIZE_8) [
                write(iunit%'i_assoc_var) zrec
                iptr=1
                ]
        ]
return
end
```

```
subroutine fft(z,n,direction)
integer*2 n,i,j,k,twok
real*4 s,direction
complex*8 z(n),u,w,temp
%C
if(direction.eq.'inverse'.or.direction.eq.'INVERSE') [
        s=-PI
        temp=cmplx(1./float(n),0.)
        do i=1,n
                z(i)=temp*z(i)
        ]
else
        s=PI
for([i=1;j=1] ; i.lt.n ; i=i+1) [                    # bit reversal
        if(i.lt.j) [                # switch only once
                temp=z(j)
                z(j)=z(i)
                z(i)=temp
                ]
        # Test bsts form high to low order.  If set, clear it and go on to next
        #   bit.  If clear, set it and stop.  I.e. bit reversed counting
        for(k=n/2 ; k.lt.j ; k=k/2)
                j=j-k
        j=j+k
        ]
# Number of stages equals log[2] of n
for([k=1;twok=2] ; twok.le.n ; [k=twok;twok=2*twok]) [
        u=(1.,0.)
        w=cexp(cmplx(0.,-s/float(k)))
        # Numper of bufferfiles equals k*n/twok = n/2
        do j=1,k [
                do i=j,n,twok [
                        temp=z(i+k)*u               # butterfly
                        z(i+k)=z(i)-temp            #     :
                        z(i)=z(i)+temp              #     :
                        ]
                u=u*w
                ]
        ]
return
end

subroutine transp(iunit,i_assoc_var,isz)
TP_IMPLICIT
COM_Z_ARRAYS
COM_PDB

for(ksz=isz/2 , ksz.ge.1 , ksz=ksz/2)               # Matrix size=ksz
        for(jk=1 ; jk.le.isz ; jk=jk+2*ksz)                 # Rows of matrices
                for(j=jk ; j.le.jk+ksz-1 ; j=j+1) [      # Row in matrix
                        call inplnz(iunit,i_assoc_var,zline1,j,isz,zrec1)
                        call inplnz(iunit,i_assoc_var,zline2,j+ksz,isz,zrec2)
                        do ik=1,isz,2*ksz                   # Columns of matrices
                                do i=ik,ik+ksz-1 [          # Column in matrix
                                        z=zline1(i+ksz)
                                        zline1(i+ksz)=zline2(i)
                                        zline2(i)=z
                                        ]
                        call ouplnz(iunit,i_assoc_var,zline1,j,isz,zrec1)
                        call ouplnz(iunit,i_assoc_var,zline2,j+ksz,isz,zrec2)
                        ]
```

```
cpdb(C_PDB_FORM)=cpdb(C_PDB_FORM).xor.PDB_FORM_WR        # wrong reading (y,x)
```

```
        .TITLE  BOX.MAC
        .GLOBL  BOX
        .MCALL  .TTINR
        ADSTAT  =       170400  ; A/D STATUS
        ADBUF   =       170402  ; A/D BUFFER
        XADDR   =       174002  ; X PIC ADDRESS
        YADDR   =       174003  ; Y PIC ADDRESS
        PICBUF  =       174000  ; DISPLAY BUFFER
;
BOX:    TST     (R5)+
        MOV     (R5)+,HORS      ; GET X POSITION ADDRESS
        MOV     (R5)+,VERT      ; GET Y POSITION ADDRESS
        MOV     @(R5)+,XL       ; GET X LENGTH
        MOV     @(R5),YL        ; GET Y LENGTH
        BIS     #10000,@#44     ; TT SPECIAL MODE (PSW BIT 12)
        BIS     #100,@#44       ; INHIBIT TT WAIT (FB MNTR)
START:  MOV     #001,@#ADSTAT   ; A/D CH 0
ADX:    BIT     #1,@#ADSTAT     ; DONE?
        BNE     ADX             ; NO
        MOV     @#ADBUF,R1      ; YES, GET X POS.
        MOV     #401,@#ADSTAT   ; A/D CH 1
ADY:    BIT     #1,@#ADSTAT     ; DONE?
        BNE     ADY             ; NO
        MOV     @#ADBUF,R3      ; YES, GET Y POS.
        ASR     R1              ; DIVIDE X BY 16
        ASR     R1
        ASR     R1
        ASR     R1
        ASR     R3              ; DIVIDE Y BY 16
        ASR     R3
        ASR     R3
        ASR     R3
        SUB     #255.,R3        ; REVERSE Y DIRECTION
        NEG     R3
;
;
        MOV     R1,@HORS        ; RETURN X POS.
        MOV     R3,@VERT        ; RETURN Y POS.
;
;
        MOV     R1,R2
        MOV     R3,R4
        MOV     XL,R0           ; XL
        ASR     R0              ; 1/2 XL
        SUB     R0,R1           ; LEFT EDGE
        ADD     R0,R2           ; RIGHT EDGE
        MOV     YL,R0           ; YL
        ASR     R0              ; 1/2 YL
        SUB     R0,R3           ; TOP EDGE
        ADD     R0,R4           ; BOTTOM EDGE
        MOVB    R1,L
        MOVB    R2,R
        MOVB    R3,T
        MOVB    R4,B
;
;
        MOV     #ARRAY,R0       ; ARRAY START
;
        MOV     #1,R4           ; LOOP COUNT
        MOVB    T,@#YADDR       ; SET Y FOR TOP
LOOPA:  MOVB    L,R1            ; START POINT
        BIC     #177400,R1      ; CLEAR HIGH BYTE
```

```
        INC     R1                  ; SKIP CORNER POINT
        MOVB    R,R2                ; END POINT
        BIC     #177400,R2          ; CLEAR HIGH BYTE
TB:     MOVB    R1,@#XADDR          ; SET X ADDRESS
        MOVB    @#PICBUF,(RO)+      ; PUT VALUE IN ARRAY
        MOVB    #255.,@#PICBUF      ; CHANGE TO WHITE
        INC     R1                  ; NEXT X ADDRESS
        CMP     R2,R1               ; AT END OF LINE?
        BGT     TB                  ; NO, GO BACK
        MOVB    B,@#YADDR           ; SET Y FOR BOTTOM
        DEC     R4                  ; CHANGE LOOP COUNT
        BEQ     LOOPA               ; DO AGAIN FOR BOTTOM
;
        MOV     #1,R4               ; LOOP COUNT
        MOVB    L,@#XADDR           ; SET X FOR LEFT
LOOPB:  MOVB    T,R1                ; START POINT
        BIC     #177400,R1          ; CLEAR HIGH BYTE
        MOVB    B,R2                ; END POINT
        BIC     #177400,R2          ; CLEAR HGIH BYTE
RL:     MOVB    R1,@#YADDR          ; SET Y ADDRESS
        MOVB    @#PICBUF,(RO)+      ; PUT VALUE IN ARRAY
        MOVB    #255.,@#PICBUF      ; CHANGE TO WHITE
        INC     R1                  ; NEXT Y ADDRESS
        CMP     R2,R1               ; AT END OF LINE?
        BGT     RL                  ; NO, GO BACK
        MOVB    R,@#XADDR           ; SET X FOR RIGHT
        DEC     R4                  ; CHANGE LOOP COUNT
        BEQ     LOOPB               ; DO AGAIN FOR RIGHT
;
; CHECK FOR KEYBOARD HIT
;    (NOTE: .TTINR USES RO)
        .TTINR                      ; TEST FOR KBD HIT
        BCC     RETURN              ; HIT -- RETURN
;
;
; RESTORE THE VALUES (ERASE BOX)
;
        MOV     #ARRAY,RO           ; ARRAY START
;
        MOV     #1,R4               ; LOOP COUNT
        MOVB    T,@#YADDR           ; SET Y FOR TOP
LOOPA2: MOVB    L,R1                ; START POINT
        BIC     #177400,R1          ; CLEAR HIGH BYTE
        INC     R1                  ; SKIP CORNER POINT
        MOVB    R,R2                ; END POINT
        BIC     #177400,R2          ; CLEAR HIGH BYTE
TB2:    MOVB    R1,@#XADDR          ; SET X ADDRESS
        MOVB    (RO)+,@#PICBUF      ; PUT OLD PIXEL BACK
        INC     R1                  ; NEXT X ADDRESS
        CMP     R2,R1               ; AT END OF LINE?
        BGT     TB2                 ; NO, GO BACK
        MOVB    B,@#YADDR           ; SET Y FOR BOTTOM
        DEC     R4                  ; CHANGE LOOP COUNT
        BEQ     LOOPA2              ; DO AGAIN FOR BOTTOM
;
        MOV     #1,R4               ; LOOP COUNT
        MOVB    L,@#XADDR           ; SET X FOR LEFT
LOOPB2: MOVB    T,R1                ; START POINT
        BIC     #177400,R1          ; CLEAR HIGH BYTE
        MOVB    B,R2                ; END POINT
        BIC     #177400,R2          ; CLEAR HIGH BYTE
RL2:    MOVB    R1,@#YADDR          ; SET Y ADDRESS
        MOVB    (RO)+,@#PICBUF      ; PUT OLD PIXEL BACK
```

```
        INC     R1              ; NEXT Y ADDRESS
        CMP     R2,R1           ; AT END OF LINE?
        BGT     RL2             ; NO, GO BACK
        MOVB    R,@#XADDR       ; SET X FOR RIGHT
        DEC     R4              ; CHANGE LOOP COUNT
        BEQ     LOOPB2          ; DO AGAIN FOR RIGHT
;
;
        JMP     START           ; DO IT ALL AGAIN
;
;
;
;
RETURN: BIC     #10000,@#44     ; TT NORMAL MODE
        RTS     PC
;
;
HORS:   0
VERT:   0
XL:     0
YL:     0
T:      .BYTE
B:      .BYTE
R:      .BYTE
L:      .BYTE
ARRAY:  .BLKB   1000
```

```
  IMPLICITBYTE(B-C),INTEGER*2(I-N),REAL*4(A,E-H,O-Y),REAL*8(D),COMP         !156
*LEX*8(Z)  !156
  COMMON/SZC/INPSZ,IOUPSZ,CIFORM,COFORM,IO0001,IO0002       !157
  CALLVSVFIQ(1)     !160
  CALLVSVCLR('all')         !161
  CALLVPRINT('Please wait a moment')          !163
  CALLPDB('DAT:OCR1.PIC','DAT:OCR1.PDB')      !164
  CALLVPRINT('okashiina')   !165
  CALLPDB('DAT:OCR2.PIC','DAT:OCR2.PDB')      !166
  CALLVPRINT('Please define subpicture A by the space key')        !167
  CALLBOX(IX,IY,64,64)      !168
  X=FLOAT(IX-31)    !170
  Y=FLOAT(IY-31)    !171
  CALLRPRINT('X of subpicture A is ',X)       !172
  CALLRPRINT('Y of subpicture A is ',Y)       !173
  CALLBOPENF('DAT:A.DAT',1,'new')   !174
  CALLRPUT('',X)    !175
  CALLRPUT('',Y)    !176
  CLOSE(UNIT=1)     !177
  CALLVPRINT('Please define subpicture B') !179
  CALLBOX(IX,IY,64,64)      !180
  X=FLOAT(IX-31)    !182
  Y=FLOAT(IY-31)    !183
  CALLRPRINT('X of subpicture B is ',X)       !184
  CALLRPRINT('Y of subpicture B is ',Y)       !185
  CALLBOPENF('DAT:B.DAT',1,'new')   !186
  CALLRPUT('',X)    !187
  CALLRPUT('',Y)    !188
  CLOSE(UNIT=1)     !189
  CALLVPRINT('I am making a mask file')       !191
  CALLMAKSR('DAT:ONE.Z',64,64,1.,32.,32.,10.,10.)   !192
  IOUPSZ=256        !199
  CALLVPRINT('converting OCR1.PDB into OCR1.Z')     !201
  CALLINPSET('DAT:OCR1.PDB')        !202
  CALLOUPSET('DAT:OCR1.Z') !203
  CALLCZCVT         !204
  CALLOUPPDB        !205
  CALLVSVCLR('characters') !206
  CALLVPRINT('converting OCR2.PDB into OCR2.Z')     !208
  CALLINPSET('DAT:OCR2.PDB')        !209
  CALLOUPSET('DAT:OCR2.Z') !210
  CALLCZCVT         !211
  CALLOUPPDB        !212
  CALLVSVCLR('characters') !213
  CALLVPRINT('making subpicture A1.Z')        !215
  CALLSUBPIC('DAT:OCR1.Z','DAT:A1.Z','DAT:A.DAT',64,64)     !216
  CALLVPRINT('making subpicture B1.Z')        !220
  CALLSUBPIC('DAT:OCR1.Z','DAT:B1.Z','DAT:B.DAT',64,64)     !221
  CALLVPRINT('making subpicture A2.Z')        !225
  CALLSUBPIC('DAT:OCR2.Z','DAT:A2.Z','DAT:A.DAT',64,64)     !226
  CALLVPRINT('making subpicture B2.Z')        !230
  CALLSUBPIC('DAT:OCR2.Z','DAT:B2.Z','DAT:B.DAT',64,64)     !231
  CALLVPRINT('making Mexican hat filter')  !242
  CALLMAKSR8('DAT:MH.Z',64,64,1.,32.,32.,0.7071068)        !243
  CALLROT11('DAT:MH.Z','DAT:MHF.Z',-31,-31)         !249
  CALLVPRINT('FFT of the Mexican hat filter')       !252
  CALLXFORM(10,'DAT:MHF.Z')         !253
  CALLVPRINT('making edge mask of filter') !255
  CALLMAKSR('DAT:MSK.Z',64,64,1.,32.5,32.5,28.5,28.5)       !256
  CALLVPRINT('FFT of subpictures') !260
  CALLXFORM(10,'DAT:A1.Z') !261
```

```
CALLXFORM(10,'DAT:B1.Z') !262
CALLXFORM(10,'DAT:A2.Z') !263
CALLXFORM(10,'DAT:B2.Z') !264
CALLVPRINT('filtering subpictures A1.Z B1.Z A2.Z B2.Z with')        !266
CALLVPRINT('Mexican hat filter') !267
CALLTWOFIL('DAT:MHF.Z','DAT:A1.Z',20,0.0001)      !268
CALLTWOFIL('DAT:MHF.Z','DAT:B1.Z',20,0.0001)      !271
CALLTWOFIL('DAT:MHF.Z','DAT:A2.Z',20,0.0001)      !274
CALLTWOFIL('DAT:MHF.Z','DAT:B2.Z',20,0.0001)      !277
CALLVPRINT('FFT[-1] of filtered subpictures')     !280
CALLXFORM(11,'DAT:A1.Z') !281
CALLXFORM(11,'DAT:B1.Z') !282
CALLXFORM(11,'DAT:A2.Z') !283
CALLXFORM(11,'DAT:B2.Z') !284
CALLVPRINT('edge masking of subpictures')         !286
CALLTWOFIL('DAT:MSK.Z','DAT:A1.Z',20,0.0001)      !287
CALLTWOFIL('DAT:MSK.Z','DAT:B1.Z',20,0.0001)      !290
CALLTWOFIL('DAT:MSK.Z','DAT:A2.Z',20,0.0001)      !293
CALLTWOFIL('DAT:MSK.Z','DAT:B2.Z',20,0.0001)      !296
CALLVPRINT('making ROI from subpictures A1.Z & B1.Z')     !305
CALLNORMF('DAT:ONE.Z','DAT:A1.Z')       !306
CALLNORMF('DAT:ONE.Z','DAT:B1.Z')       !307
CALLVPRINT('correlating')      !315
CALLCOPY('DAT:A2.Z','DAT:AA2.Z',65)       !316
CALLCOPY('DAT:B2.Z','DAT:BB2.Z',65)       !317
CALLTWOFIL('DAT:A2.Z','DAT:AA2.Z',20,0.0001)      !319
CALLTWOFIL('DAT:B2.Z','DAT:BB2.Z',20,0.0001)      !320
CALLVPRINT('FFT for correlating')       !322
CALLXFORM(10,'DAT:A2.Z') !323
CALLXFORM(10,'DAT:B2.Z') !324
CALLXFORM(10,'DAT:AA2.Z')      !325
CALLXFORM(10,'DAT:BB2.Z')      !326
CALLVPRINT('I am tired') !328
CALLXFORM(10,'DAT:A1.Z') !330
CALLXFORM(10,'DAT:B1.Z') !331
CALLCOPY('DAT:ONE.Z','DAT:ONE.Z1',65)     !332
CALLXFORM(10,'DAT:ONE.Z1')     !333
CALLCOPY('DAT:A1.Z','DAT:A1.Z1',65)       !335
CALLCOPY('DAT:B1.Z','DAT:B1.Z1',65)       !336
CALLVPRINT('Help me , Yoshi !')  !337
CALLTWOFIL('DAT:A2.Z','DAT:A1.Z1',21,0.0001)      !338
CALLTWOFIL('DAT:B2.Z','DAT:B1.Z1',21,0.0001)      !339
CALLCOPY('DAT:ONE.Z1','DAT:ONE.Z2',65)    !340
CALLTWOFIL('DAT:AA2.Z','DAT:ONE.Z1',21,0.0001)    !341
CALLTWOFIL('DAT:BB2.Z','DAT:ONE.Z2',21,0.0001)    !342
CALLVPRINT('FFT[-1]')    !344
CALLXFORM(11,'DAT:A1.Z1')      !345
CALLXFORM(11,'DAT:B1.Z1')      !346
CALLXFORM(11,'DAT:ONE.Z1')     !347
CALLXFORM(11,'DAT:ONE.Z2')     !348
CALLVPRINT('square root')      !350
CALLONEFL9('DAT:ONE.Z1') !351
CALLONEFL9('DAT:ONE.Z2') !352
CALLVPRINT('almost done')      !354
CALLTWOFIL('DAT:ONE.Z1','DAT:A1.Z1',23,0.0001)    !355
CALLTWOFIL('DAT:ONE.Z2','DAT:B1.Z1',23,0.0001)    !356
CALLVPRINT('finding peak')     !358
CALLPEAK4('DAT:A1.Z1','DAT:OCRBOX.DAT',1)         !360
CALLCTROM('DAT:ONE.Z','DAT:OCRBOX.DAT','DAT:A.DAT',2)     !362
CALLPEAK4('DAT:B1.Z1','DAT:OCRBOX.DAT',3)         !364
CALLCTROM('DAT:ONE.Z','DAT:OCRBOX.DAT','DAT:B.DAT',4)     !366
CALLCALC20('DAT:OCRBOX.DAT',1,3,5)      !368
```

```
      SUBROUTINEPDB(CSTR1,CSTR2)          !149
      IMPLICITBYTE(B-C),INTEGER*2(I-N),REAL*4(A,E-H,O-Y),REAL*8(D),COMP           !150
     *LEX*8(Z)   !150
      BYTECSTR1(81)       !151
      BYTECSTR2(81)       !152
      BYTECPDB(512)       !153
      INTEGER*2IPDB(256)          !153
      REAL*4RPDB(128)   !153
      EQUIVALENCE(CPDB,IPDB,RPDB)         !153
      COMMON/PDBC/IPDB !153
      DATAIPDB/256*0/   !154
C
      CONTINUE !156
23002 CONTINUE !156
      OPEN(UNIT=1,NAME=CSTR1,TYPE='OLD',ACCESS='DIRECT',RECORDSIZE=128,           !158
     *ASSOCIATEVARIABLE=IASV1,ERR=1)      !158
      IF(.NOT.(.FALSE.))GOTO23005         !159
1     CALLVSVCLR('characters') !160
      CALLVPRINT(CSTR1)           !161
      CALLVPRINT('Input file not found')          !162
      GOTO23003           !163
23005 CONTINUE !165
      GOTO23004           !165
23003 GOTO23002           !167
23004 CONTINUE !167
      IASV1=1   !167
      OPEN(UNIT=2,NAME=CSTR2,TYPE='NEW',ACCESS='DIRECT',RECORDSIZE=128,           !169
     *ASSOCIATEVARIABLE=IASV2)   !169
      IPDB(15)=256        !170
      IPDB(16)=256        !171
      IASV2=1   !172
      WRITE(2'IASV2)IPDB          !173
      DO 23007I=1,128   !174
      READ(1'IASV1)IPDB           !175
      WRITE(2'IASV2)IPDB          !176
23007 CONTINUE !177
23008 CONTINUE !177
      CLOSE(UNIT=1)       !178
      CLOSE(UNIT=2)       !179
      RETURN    !181
      END       !182
      SUBROUTINEMAKSR(CSTR,ISZX,ISZY,RMAG,XO,YO,XC,YC) !185
      IMPLICITBYTE(B-C),INTEGER*2(I-N),REAL*4(A,E-H,O-Y),REAL*8(D),COMP           !191
     *LEX*8(Z)   !191
      COMPLEX*8ZLINE(256),ZREC(64)        !192
      COMPLEX*8ZBUFF(64,4)        !192
      REAL*4RLINE(512) !192
      BYTECLINE(256),CREC(512) !192
      EQUIVALENCE(ZLINE,RLINE,CLINE,ZBUFF)        !192
      EQUIVALENCE(ZREC,CREC)      !192
      COMMON/ZC/ZREC,ZLINE        !192
      BYTECPDB(512)       !193
      INTEGER*2IPDB(256)          !193
      REAL*4RPDB(128)   !193
      EQUIVALENCE(CPDB,IPDB,RPDB)         !193
      COMMON/PDBC/IPDB !193
      DO 23009I=1,256   !194
      IPDB(I)=0           !195
23009 CONTINUE !195
23010 CONTINUE !195
      CALLOPNFNW(CSTR,1,IOOOO1,ISZX,ISZY)         !196
```

```
      ZMAG=CMPLX(RMAG,0.)          !197
      XMAX=AMIN1(XO-1.,FLOAT(ISZX)-XO) !198
      YMAX=AMIN1(YO-1.,FLOAT(ISZY)-YO) !199
      RMAX=AMIN1(XMAX,YMAX)     !200
      CONTINUE !201
      IY=1      !201
      RMAX=0.   !201
23011 IF(.NOT.(IY . LE . ISZY ))GOTO23013        !201
      DO 23014IX=1,ISZX         !202
      RSQ=(FLOAT(IX)-XO)**2+(FLOAT(IY)-YO)**2  !203
      R=SQRT(RSQ)         !204
      IF(.NOT.(ABS(FLOAT(IX)-XO).LE.XC.AND.ABS(FLOAT(IY)-YO) LE.YC))GOT          !205
     *O23016    !205
      ZLINE(IX)=ZMAG     !206
      GOTO23017          !207
23016 CONTINUE !207
      ZLINE(IX)=(0.,0.)          !208
23017 CONTINUE !208
      RMAX=AMAX1(RMAX,ABS(REAL(ZLINE(IX))))     !209
23014 CONTINUE !210
23015 CONTINUE !210
      CALLOUPLNZ(1,I00001,ZLINE,IY,ISZX,ZREC)   !211
23012 IY=IY+1   !212
      GOTO23011          !212
23013 CONTINUE !212
      IPDB(15)=ISZX      !213
      IPDB(16)=ISZY      !214
      RPDB(128)=RMAX     !215
      WRITE(1'1)IPDB     !216
      CLOSE(UNIT=1)      !217
      RETURN    !218
      END       !219
      SUBROUTINESUBPIC(CSTR1,CSTR2,CSTR3,ISZX2,ISZY2)   !225
      IMPLICITBYTE(B-C),INTEGER*2(I-N),REAL*4(A,E-H,O-Y),REAL*8(D),COMP          !226
     *LEX*8(Z)  !226
      BYTECPDB(512)      !227
      INTEGER*2IPDB(256)         !227
      REAL*4RPDB(128)  !227
      EQUIVALENCE(CPDB,IPDB,RPDB)        !227
      COMMON/PDBC/IPDB !227
      COMPLEX*8ZLINE(256),ZREC(64)       !228
      COMPLEX*8ZBUFF(64,4)       !228
      REAL*4RLINE(512) !228
      BYTECLINE(256),CREC(512) !228
      EQUIVALENCE(ZLINE,RLINE,CLINE,ZBUFF)       !228
      EQUIVALENCE(ZREC,CREC)     !228
      COMMON/ZC/ZREC,ZLINE       !228
      I00002=IPDB(250) !229
      CALLBOPENF(CSTR3,3,'old')          !230
      CALLRGET(XO)       !231
      CALLRGET(YO)       !232
      CALLIPRINT('output x-dimension is ',ISZX2)        !233
      CALLOPNFOD(CSTR1,1,I00003,ISZX1,ISZY1)    !234
      CALLOPNFNW(CSTR2,2,I00004,ISZX2,ISZY2)    !235
      IXO=IFIX(XO)       !236
      IYO=IFIX(YO)       !237
      CONTINUE !238
      IY1=IYO   !238
      IY2=1     !238
      RMAX=0.   !238
23018 IF(.NOT.(IY2 . LE . ISZY2 ))GOTO23020     !238
      CALLINPLNZ(1,I00003,ZLINE,IY1,ISZX1,ZREC)         !239
      CALLOUPLNZ(2,I00004,ZLINE(IXO),IY2,ISZX2,ZREC)    !240
```

```
      DO 23021I=IXO,IXO+ISZX2-1            !241
      RMAX=AMAX1(RMAX,ABS(REAL(ZLINE(I))))      !242
23021 CONTINUE !242
23022 CONTINUE !242
23019 IY1=IY1+1          !243
      IY2=IY2+1          !243
      GOTO23018          !243
23020 CONTINUE !243
      IPDB(15)=ISZX2     !244
      IPDB(16)=ISZY2     !245
      IF(.NOT.(I00002.EQ.0))GOTO23023   !246
      IPDB(250)=ISZX2    !247
      IPDB(251)=IXO-1    !248
      IPDB(252)=IYO-1    !249
      GOTO23024          !251
23023 CONTINUE !251
      IPDB(251)=IXO-1    !252
      IPDB(252)=IYO-1    !253
23024 CONTINUE !254
      RPDB(128)=RMAX     !255
      WRITE(2'1)IPDB     !256
      CLOSE(UNIT=1)      !257
      CLOSE(UNIT=2)      !258
      CLOSE(UNIT=3)      !259
      RETURN    !260
      END       !261
      SUBROUTINEMAKSRB(CSTR,ISZX,ISZY,RMAG,XO,YO,SIGMA)           !268
      IMPLICITBYTE(B-C),INTEGER*2(I-N),REAL*4(A,E-H,O-Y),REAL*8(D),COMP          !269
     *LEX*8(Z)   !269
      COMPLEX*8ZLINE(256),ZREC(64)       !270
      COMPLEX*8ZBUFF(64,4)       !270
      REAL*4RLINE(512) !270
      BYTECLINE(256),CREC(512) !270
      EQUIVALENCE(ZLINE,RLINE,CLINE,ZBUFF)       !270
      EQUIVALENCE(ZREC,CREC)     !270
      COMMON/ZC/ZREC,ZLINE       !270
      BYTECPDB(512)      !271
      INTEGER*2IPDB(256)         !271
      REAL*4RPDB(128)    !271
      EQUIVALENCE(CPDB,IPDB,RPDB)        !271
      COMMON/PDBC/IPDB !271
      DO 23025I=1,256    !272
      IPDB(I)=0          !273
23025 CONTINUE !273
23026 CONTINUE !273
      CALLOPNFNW(CSTR,1,I00001,ISZX,ISZY)        !274
      ZMAG=CMPLX(RMAG,0.)        !275
      XMAX=AMIN1(XO-1.,FLOAT(ISZX)-XO) !276
      YMAX=AMIN1(YO-1.,FLOAT(ISZY)-YO) !277
      RMAX=AMIN1(XMAX,YMAX)      !278
      V00005=1./SIGMA**2         !279
      T00006=1./(2.*SIGMA**2)    !280
      S00007=1./(SIGMA**4*SQRT(2.*3.14159265358979))      !281
      CONTINUE !282
      IY=1       !282
      RMAX=0.    !282
23027 IF(.NOT (IY . LE . ISZY ))GOTO23029       !282
      DO 23030IX=1,ISZX          !283
      RSQ=(FLOAT(IX)-XO)**2+(FLOAT(IY)-YO)**2   !284
      R=SQRT(RSQ)        !285
      ZLINE(IX)=CMPLX(S00007*(2.-V00005*RSQ)*EXP(-T00006*RSQ),0.)          !287
      RMAX=AMAX1(RMAX,ABS(REAL(ZLINE(IX))))     !288
23030 CONTINUE !289
```

```
23031   CONTINUE !289
        CALLOUPLNZ(1,I00001,ZLINE,IY,ISZX,ZREC)   !290
23028   IY=IY+1  !291
        GOTO23027          !291
23029   CONTINUE !291
        IPDB(15)=ISZX      !292
        IPDB(16)=ISZY      !293
        RPDB(128)=RMAX     !294
        WRITE(1'1)IPDB     !295
        CLOSE(UNIT=1)      !296
        RETURN    !297
        END       !298
        SUBROUTINEROT11(CSTR1,CSTR2,IX,IY)        !307
        IMPLICITBYTE(B-C),INTEGER*2(I-N),REAL*4(A,E-H,O-Y),REAL*8(D),COMP          !314
     *LEX*8(Z)  !314
        COMPLEX*8ZLINE(512)       !315
        COMPLEX*8ZLINE1(256),ZLINE2(256)  !315
        COMPLEX*8ZBUFF(64,4)      !315
        COMPLEX*8ZREC1(64),ZREC2(64)      !315
        REAL*4RLINE1(512),RLINE2(512)     !315
        BYTECLINE1(256),CLINE2(256)       !315
        BYTECREC1(512),CREC2(512)         !315
        EQUIVALENCE(ZLINE,ZLINE1,RLINE1,ZBUFF)    !315
        EQUIVALENCE(ZLINE(257),ZLINE2,RLINE2)     !315
        EQUIVALENCE(ZREC1,CREC1),(ZREC2,CREC2)    !315
        COMMON/ZC/ZREC1,ZLINE,ZREC2       !315
        BYTECPDB(512)     !316
        INTEGER*2IPDB(256)        !316
        REAL*4RPDB(128)  !316
        EQUIVALENCE(CPDB,IPDB,RPDB)       !316
        COMMON/PDBC/IPDB !316
        EQUIVALENCE(ISZX,ISZX1),(ISZY,ISZY1)      !317
        COMMON/PPRAMC/ISZX1,ISZY1,ISZX2,ISZY2,SCALEX,SCALEY,U00008,U00009          !317
        CALLOPNFOD(CSTR1,1,I00003,ISZX1,ISZY1)    !318
        CALLOPNFNW(CSTR2,2,I00004,ISZX1,ISZY1)    !319
        IF(.NOT.(IX.LT.0))GOTO23032       !320
        IX=ISZX1+IX       !321
23032   CONTINUE !322
        IF(.NOT.(IY.LT.0))GOTO23034       !322
        IY=ISZY1+IY       !323
23034   CONTINUE !324
        CONTINUE !324
        J=1      !324
23036   IF(.NOT.(J . LE . ISZY1 ))GOTO23038       !324
        CALLINPLNZ(1,I00003,ZLINE1,J,ISZX1,ZREC1)         !325
        DO 23039I=1,ISZX1         !326
        ZLINE2(MOD(I-1+IX,ISZX1)+1)=ZLINE1(I)     !327
23039   CONTINUE !327
23040   CONTINUE !327
        CALLOUPLNZ(2,I00004,ZLINE2,MOD(J-1+IY,ISZY1)+1,ISZX1,ZREC2)        !329
23037   J=J+1    !330
        GOTO23036         !330
23038   CONTINUE !330
        IPDB(251)=0       !331
        IPDB(252)=0       !332
        WRITE(2'1)IPDB    !333
        CLOSE(UNIT=1)     !334
        CLOSE(UNIT=2)     !335
        RETURN   !336
        END      !337
        SUBROUTINEXFORM(IOP,CSTR)         !340
        IMPLICITBYTE(B-C),INTEGER*2(I-N),REAL*4(A,E-H,O-Y),REAL*8(D),COMP          !344
     *LEX*8(Z)  !344
```

```
      COMPLEX*8ZLINE(512)       !345
      COMPLEX*8ZLINE1(256),ZLINE2(256) !345
      COMPLEX*8ZBUFF(64,4)      !345
      COMPLEX*8ZREC1(64),ZREC2(64)     !345
      REAL*4RLINE1(512),RLINE2(512)    !345
      BYTECLINE1(256),CLINE2(256)      !345
      BYTECREC1(512),CREC2(512)        !345
      EQUIVALENCE(ZLINE,ZLINE1,RLINE1,ZBUFF)    !345
      EQUIVALENCE(ZLINE(257),ZLINE2,RLINE2)     !345
      EQUIVALENCE(ZREC1,CREC1),(ZREC2,CREC2)    !345
      COMMON/ZC/ZREC1,ZLINE,ZREC2      !345
      BYTECPDB(512)     !346
      INTEGER*2IPDB(256)        !346
      REAL*4RPDB(128)   !346
      EQUIVALENCE(CPDB,IPDB,RPDB)      !346
      COMMON/PDBC/IPDB !346
      CALLOPNFOD(CSTR,1,IO0001,ISZX,ISZY)       !347
      CONTINUE !348
      J=1       !348
      RMAX=0.   !348
23041 IF(.NOT.(J . LE . ISZY ))GOTO23043        !348
      CALLINPLNZ(1,IO0001,ZLINE1,J,ISZX,ZREC1) !349
      IF(.NOT.(IOP.EQ.10))GOTO23044    !350
      CALLFFT(ZLINE1,ISZX,'forward')   !351
      GOTO23045         !352
23044 CONTINUE !352
      CALLFFT(ZLINE1,ISZX,'inverse')   !353
23045 CONTINUE !353
      CALLOUPLNZ(1,IO0001,ZLINE1,J,ISZX,ZREC1) !354
      DO 23046I=1,ISZX !355
      RMAX=AMAX1(RMAX,ABS(REAL(ZLINE1(I))))     !356
23046 CONTINUE !356
23047 CONTINUE !356
23042 J=J+1     !357
      GOTO23041         !357
23043 CONTINUE !357
      CALLTRANSP(1,IO0001,ISZX)        !358
      CONTINUE !359
      J=1       !359
      RMAX=0.   !359
23048 IF(.NOT.(J . LE . ISZY ))GOTO23050        !359
      CALLINPLNZ(1,IO0001,ZLINE1,J,ISZX,ZREC1) !360
      IF(.NOT.(IOP.EQ.10))GOTO23051    !361
      CALLFFT(ZLINE1,ISZX,'forward')   !362
      GOTO23052         !363
23051 CONTINUE !363
      CALLFFT(ZLINE1,ISZX,'inverse')   !364
23052 CONTINUE !364
      CALLOUPLNZ(1,IO0001,ZLINE1,J,ISZX,ZREC1) !365
      DO 23053I=1,ISZX !366
      RMAX=AMAX1(RMAX,ABS(REAL(ZLINE1(I))))     !367
23053 CONTINUE !367
23054 CONTINUE !367
23049 J=J+1     !368
      GOTO23048         !368
23050 CONTINUE !368
      RPDB(128)=RMAX    !369
      WRITE(1'1)IPDB    !370
      CLOSE(UNIT=1)     !371
      RETURN    !372
      END       !373
      SUBROUTINETWOFIL(CSTR1,CSTR2,IOP,RMIN)    !376
      IMPLICITBYTE(B-C),INTEGER*2(I-N),REAL*4(A,E-H,O-Y),REAL*8(D),COMP        '384
```

```
*LEX*8(Z)  !384
      COMPLEX*8ZLINE(512)        !385
      COMPLEX*8ZLINE1(256),ZLINE2(256) !385
      COMPLEX*8ZBUFF(64,4)       !385
      COMPLEX*8ZREC1(64),ZREC2(64)       !385
      REAL*4RLINE1(512),RLINE2(512)      !385
      BYTECLINE1(256),CLINE2(256)        !385
      BYTECREC1(512),CREC2(512)          !385
      EQUIVALENCE(ZLINE,ZLINE1,RLINE1,ZBUFF)    !385
      EQUIVALENCE(ZLINE(257),ZLINE2,RLINE2)     !385
      EQUIVALENCE(ZREC1,CREC1),(ZREC2,CREC2)    !385
      COMMON/ZC/ZREC1,ZLINE,ZREC2        !385
      BYTECPDB(512)     !386
      INTEGER*2IPDB(256)         !386
      REAL*4RPDB(128)   !386
      EQUIVALENCE(CPDB,IPDB,RPDB)        !386
      COMMON/PDBC/IPDB !386
      CALLOPNFOD(CSTR1,1,I00003,ISZX1,ISZY1)    !387
      RMAX1=RPDB(128)   !388
      CALLOPNFOD(CSTR2,2,I00004,ISZX2,ISZY2)    !389
      IF(.NOT.(IOP.EQ.23))GOTO23055      !390
      RMIN=ABS(RMAX1*RMIN)       !391
23055 CONTINUE !392
      I00003=2 !392
      I00004=2 !393
      RMAX=0.  !394
      N00010=FLOAT(ISZX2)*FLOAT(ISZY2)/64+0.999         !395
      DO 23057J=1,N00010         !396
      READ(1'I00003)ZREC1        !397
      READ(2'I00004)ZREC2        !398
      IF(.NOT.(IOP.EQ.20))GOTO23059      !399
      DO 23061I=1,64    !400
      ZREC2(I)=ZREC1(I)*ZREC2(I)         !401
23061 CONTINUE !401
23062 CONTINUE !401
23059 CONTINUE !402
      IF(.NOT.(IOP.EQ.21))GOTO23063      !402
      DO 23065I=1,64    !403
      ZREC2(I)=ZREC1(I)*CONJG(ZREC2(I))         !404
23065 CONTINUE !404
23066 CONTINUE !404
23063 CONTINUE !405
      IF(.NOT.(IOP.EQ.23))GOTO23067      !405
      DO 23069I=1,64    !406
      IF(.NOT.(ABS(REAL(ZREC1(I))).GE.RMIN))GOTO23071   !407
      ZREC2(I)=CMPLX(REAL(ZREC2(I))/REAL(ZREC1(I)),0.) !408
      GOTO23072         !409
23071 CONTINUE !409
      ZREC2(I)=(0.,0.) !410
23072 CONTINUE !410
23069 CONTINUE !410
23070 CONTINUE !410
23067 CONTINUE !411
      WRITE(2'I00004-1)ZREC2     !411
      DO 23073I=1,64    !412
      RMAX=AMAX1(RMAX,ABS(REAL(ZREC2(I))))      !413
23073 CONTINUE !413
23074 CONTINUE !413
23057 CONTINUE !414
23058 CONTINUE !414
      RPDB(128)=RMAX    !415
      WRITE(2'1)IPDB    !416
      CLOSE(UNIT=1)     !417
```

```
      CLOSE(UNIT=2)      !418
      RETURN    !419
      END       !420
      SUBROUTINENORMF(CSTR1,CSTR2)       !425
      IMPLICITBYTE(B-C),INTEGER*2(I-N),REAL*4(A,E-H,O-Y),REAL*8(D),COMP          !426
     *LEX*8(Z)  !426
      COMPLEX*8ZLINE(512)        !427
      COMPLEX*8ZLINE1(256),ZLINE2(256)  !427
      COMPLEX*8ZBUFF(64,4)       !427
      COMPLEX*8ZREC1(64),ZREC2(64)      !427
      REAL*4RLINE1(512),RLINE2(512)     !427
      BYTECLINE1(256),CLINE2(256)       !427
      BYTECREC1(512),CREC2(512)         !427
      EQUIVALENCE(ZLINE,ZLINE1,RLINE1,ZBUFF)     !427
      EQUIVALENCE(ZLINE(257),ZLINE2,RLINE2)      !427
      EQUIVALENCE(ZREC1,CREC1),(ZREC2,CREC2)     !427
      COMMON/ZC/ZREC1,ZLINE,ZREC2       !427
      BYTECPDB(512)     !428
      INTEGER*2IPDB(256)        !428
      REAL*4RPDB(128)   !428
      EQUIVALENCE(CPDB,IPDB,RPDB)       !428
      COMMON/PDBC/IPDB !428
      CALLOPNFOD(CSTR1,1,I00003,ISZX1,ISZY1)     !429
      CALLOPNFOD(CSTR2,2,I00004,ISZX2,ISZY2)     !430
      CONTINUE !431
      J=1       !431
      NPEL=0    !431
      ZSUM=(0.,0.)      !431
      SUMSQ=0.  !431
23075 IF(.NOT.(J .LE. ISZY2 ))GOTO23077      !431
      CALLINPLNZ(1,I00003,ZLINE1,J,ISZX1,ZREC1)          !432
      CALLINPLNZ(2,I00004,ZLINE2,J,ISZX2,ZREC2)          !433
      DO 23078I=1,ISZX2         !434
      IF(.NOT.(REAL(ZLINE1(I)).NE.0.))GOTO23080          !435
      NPEL=NPEL+1       !436
      ZSUM=ZSUM+ZLINE2(I)       !437
      SUMSQ=SUMSQ+CABS(ZLINE2(I))       !438
23080 CONTINUE !440
23078 CONTINUE !440
23079 CONTINUE !440
23076 J=J+1     !440
      GOTO23075         !440
23077 CONTINUE !440
      IF(.NOT.(NPEL.EQ.0))GOTO23082     !441
      CALLVPRINT('keisan dekimasen')    !442
      RETURN    !443
23082 CONTINUE !445
      ZMEAN=ZSUM/NPEL   !445
      R00011=1./SQRT(SUMSQ-CABS(ZSUM)/NPEL)      !446
      CONTINUE !447
      J=1       !447
      RMAX=0.   !447
23084 IF(.NOT.(J .LE. ISZY2 ))GOTO23086      !447
      CALLINPLNZ(1,I00003,ZLINE1,J,ISZX1,ZREC1)          !448
      CALLINPLNZ(2,I00004,ZLINE2,J,ISZX2,ZREC2)          !449
      DO 23087I=1,ISZX2         !450
      IF(.NOT.(REAL(ZLINE1(I)).NE.0.))GOTO23089          !451
      ZLINE2(I)=(ZLINE2(I)-ZMEAN)*R00011         !452
      GOTO23090         !453
23089 CONTINUE !453
      ZLINE2(I)=(0.,0.)         !454
23090 CONTINUE !454
23087 CONTINUE !454
```

```
23088  CONTINUE !454
       DO 23091I=1, ISZX2          !455
       RMAX=AMAX1(RMAX, ABS(REAL(ZLINE2(I))))     !456
23091  CONTINUE !456
23092  CONTINUE !456
       CALLOUPLNZ(2, I00004, ZLINE2, J, ISZX2, ZREC2)          !457
23085  J=J+1     !458
       GOTO23084         !458
23086  CONTINUE !458
       RPDB(128)=RMAX    !459
       WRITE(2'1)IPDB    !460
       CLOSE(UNIT=1)     !461
       CLOSE(UNIT=2)     !462
       RETURN    !463
       END       !464
       SUBROUTINEONEFL9(CSTR)    !470
       IMPLICITBYTE(B-C), INTEGER*2(I-N), REAL*4(A, E-H, O-Y), REAL*8(D), COMP        !471
      *LEX*8(Z)  !471
       COMPLEX*8ZLINE(512)         !472
       COMPLEX*8ZLINE1(256), ZLINE2(256)  !472
       COMPLEX*8ZBUFF(64, 4)       !472
       COMPLEX*8ZREC1(64), ZREC2(64)      !472
       REAL*4RLINE1(512), RLINE2(512)     !472
       BYTECLINE1(256), CLINE2(256)       !472
       BYTECREC1(512), CREC2(512)         !472
       EQUIVALENCE(ZLINE, ZLINE1, RLINE1, ZBUFF)     !472
       EQUIVALENCE(ZLINE(257), ZLINE2, RLINE2)       !472
       EQUIVALENCE(ZREC1, CREC1), (ZREC2, CREC2)     !472
       COMMON/ZC/ZREC1, ZLINE, ZREC2      !472
       BYTECPDB(512)     !473
       INTEGER*2IPDB(256)          !473
       REAL*4RPDB(128)   !473
       EQUIVALENCE(CPDB, IPDB, RPDB)      !473
       COMMON/PDBC/IPDB !473
       CALLOPNFOD(CSTR, 1, I00001, ISZX, ISZY)       !474
       N00010=FLOAT(ISZX)*FLOAT(ISZY)/64+.999        !475
       RMAX=RPDB(128)    !476
       I00001=2 !477
       RMAX=0.   !478
       DO 23093J=1, N00010         !479
       READ(1'I00001)ZREC1         !480
       DO 23095I=1, 64   !481
       ZREC1(I)=CMPLX(SQRT(AMAX1(0., REAL(ZREC1(I)))), 0.)         !482
23095  CONTINUE !482
23096  CONTINUE !482
       WRITE(1'I00001-1)ZREC1      !483
       DO 23097I=1, 64   !484
       RMAX=AMAX1(RMAX, ABS(REAL(ZREC1(I))))      !485
23097  CONTINUE !485
23098  CONTINUE !485
23093  CONTINUE !486
23094  CONTINUE !486
       RPDB(128)=RMAX    !487
       WRITE(1'1)IPDB    !488
       CLOSE(UNIT=1)     !489
       RETURN    !490
       END       !491
       SUBROUTINEPEAK4(CSTR1, CSTR2, N)    !496
       IMPLICITBYTE(B-C), INTEGER*2(I-N), REAL*4(A, E-H, O-Y), REAL*8(D), COMP        !497
      *LEX*8(Z)  !497
       REAL*4F(3, 3), TEMP(3)      !498
       BYTECPDB(512)     !499
       INTEGER*2IPDB(256)          !499
```

```
        REAL*4RPDB(128)  !499
        EQUIVALENCE(CPDB,IPDB,RPDB)        !499
        COMMON/PDBC/IPDB !499
        COMPLEX*8ZLINE(256),ZREC(64)       !500
        COMPLEX*8ZBUFF(64,4)        !500
        REAL*4RLINE(512) !500
        BYTECLINE(256),CREC(512) !500
        EQUIVALENCE(ZLINE,RLINE,CLINE,ZBUFF)        !500
        EQUIVALENCE(ZREC,CREC)    !500
        COMMON/ZC/ZREC,ZLINE      !500
        TO(F1,F2,F3)=(F1-F3)/(2.*(F1-2.*F2+F3))   !501
        F00012(TO,F1,F2,F3)=(F1-2.*F2+F3)/2.*TO**2+(F3-F1)/2.*TO+F2         !502
        CALLOPNFOD(CSTR1,1,I00001,ISZX,ISZY)        !503
        CONTINUE !504
        J=1       !504
        RX=0.     !504
        RY=0.     !504
        RN=0.     !504
        RMAX=-1.E38       !504
23099   IF(.NOT.(J .LE. ISZY ))GOTO23101          !504
        CALLINPLNZ(1,I00001,ZLINE,J,ISZX,ZREC)    !505
        DO 23102I=1,ISZX !506
        R=REAL(ZLINE(I)) !507
        IF(.NOT.(R.LT RMAX))GOTO23104      !508
        GOTO23102         !509
23104   CONTINUE !510
        IF(.NOT.(R.EQ.RMAX))GOTO23106      !510
        RN=RN+1. !511
        RX=RX+FLOAT(I)    !512
        RY=RY+FLOAT(J)    !513
        GOTO23102         !514
23106   CONTINUE !516
        RMAX=R    !516
        RN=1.     !517
        RX=I      !518
        RY=J      !519
23102   CONTINUE !520
23103   CONTINUE !520
23100   J=J+1     !521
        GOTO23099         !521
23101   CONTINUE !521
        IF(.NOT.(RN.EQ.0.))GOTO23108       !522
        STOP'No maximum found'    !523
23108   CONTINUE !524
        RX=RX/RN !524
        RY=RY/RN !525
        IX=RX+.5 !526
        IY=RY+.5 !527
        IF(.NOT.(RN.GT.1.))GOTO23110       !528
        CALLVPRINT('More than one peak found')     !529
        CALLRPRINT('rn = ',RN)    !530
        GOTO23111         !532
23110   CONTINUE !532
        CONTINUE !533
        J=1       !533
23112   IF(.NOT.(J .LE. 3 ))GOTO23114  !533
        CALLINPLNZ(1,I00001,ZLINE,MOD(IY+(J-2)-1+ISZY,ISZY)+1,ISZX,ZREC) !535
        DO 23115I=1,3     !536
        II=MOD(IX+(I-2)-1+ISZX,ISZX)+1    !537
        F(I,J)=REAL(ZLINE(II))    !538
23115   CONTINUE !539
23116   CONTINUE !539
23113   J=J+1     !540
```

```
      GOTO23112          !540
23114 CONTINUE !540
      CONTINUE !541
      ITERS=1   !541
      TOX=0.    !541
      TOY=0.    !541
23117 IF(.NOT.(ITERS . LE . 3 ))GOTO23119       !541
      DO 23120I=1,3     !542
      TEMP(I)=F00012(TOY,F(I,1),F(I,2),F(I,3)) !543
23120 CONTINUE !543
23121 CONTINUE !543
      TOX=TO(TEMP(1),TEMP(2),TEMP(3))   !544
      DO 23122I=1,3     !545
      TEMP(I)=F00012(TOX,F(1,I),F(2,I),F(3,I)) !546
23122 CONTINUE !546
23123 CONTINUE !546
      TOY=TO(TEMP(1),TEMP(2),TEMP(3))   !547
23118 ITERS=ITERS+1      !548
      GOTO23117          !548
23119 CONTINUE !548
      RMAX=TEMP(2)       !549
      CALLRPRINT('x interpolation is ',TOX)     !550
      CALLRPRINT('y interpolation is ',TOY)     !551
      RX=RX+TOX          !552
      RY=RY+TOY          !553
23111 CONTINUE !554
      IF(.NOT.(IFIX(RX).GT.ISZX/2))GOTO23124    !555
      RX=RX-FLOAT(ISZX)           !556
23124 CONTINUE !557
      IF(.NOT.(IFIX(RY).GT.ISZY/2))GOTO23126    !557
      RY=RY-FLOAT(ISZY)           !558
23126 CONTINUE !559
      RX=RX+FLOAT(IPDB(251))    !559
      RY=RY+FLOAT(IPDB(252))    !560
      CALLRPRINT('Rmax is ',RMAX)         !561
      CALLRPRINT('The x position is ',RX)       !562
      CALLRPRINT('The y position is ',RY)       !563
      CALLBOPENP(CSTR2,2,3*(N-1))         !564
      CALLRPUT('',RX)   !565
      CALLRPUT('',RY)   !566
      CALLRPUT('',RN)   !567
      CLOSE(UNIT=2)      !568
      CLOSE(UNIT=1)      !569
      RETURN    !570
      END       !571
      SUBROUTINECTROM(CSTR1,CSTR2,CSTR3,N)      !577
      IMPLICITBYTE(B-C),INTEGER*2(I-N),REAL*4(A,E-H,O-Y),REAL*8(D),COMP          !579
     *LEX*8(Z)   !579
      COMPLEX*8ZLINE(256),ZREC(64)        !580
      COMPLEX*8ZBUFF(64,4)       !580
      REAL*4RLINE(512) !580
      BYTECLINE(256),CREC(512) !580
      EQUIVALENCE(ZLINE,RLINE,CLINE,ZBUFF)      !580
      EQUIVALENCE(ZREC,CREC)     !580
      COMMON/ZC/ZREC,ZLINE       !580
      BYTECPDB(512)      !581
      INTEGER*2IPDB(256)         !581
      REAL*4RPDB(128)   !581
      EQUIVALENCE(CPDB,IPDB,RPDB)         !581
      COMMON/PDBC/IPDB !581
      CALLOPNFOD(CSTR1,1,I00001,ISZX,ISZY)      !582
      CONTINUE !583
      J=1       !583
```

```
      RX=0.      !583
      RY=0.      !583
      F=0.       !583
23128 IF(.NOT.(J .LE. ISZY ))GOTO23130          !583
      CALLINPLNZ(1,I00001,ZLINE,J,ISZX,ZREC)    !584
      DO 23131I=1,ISZX !585
      R=REAL(ZLINE(I)) !586
      F=F+R      !587
      RX=RX+R*FLOAT(I) !588
      RY=RY+R*FLOAT(J) !589
23131 CONTINUE !590
23132 CONTINUE !590
23129 J=J+1      !591
      GOTO23128          !591
23130 CONTINUE !591
      RX=RX/F   !592
      RY=RY/F   !593
      RX=RX+FLOAT(IPDB(251))     !594
      RY=RY+FLOAT(IPDB(252))     !595
      CALLRPRINT('The x center is ',RX)         !596
      CALLRPRINT('The y center is ',RY)         !597
      CLOSE(UNIT=1)      !598
      CALLBOPENF(CSTR3,1,'old')           !599
      CALLRGET(X1)       !600
      CALLRGET(Y1)       !601
      CLOSE(UNIT=1)      !602
      CALLBOPENP(CSTR2,2,3*(N-1))         !603
      CALLRPUT('',X1)    !604
      CALLRPUT('',Y1)    !605
      CALLRPUT('',F)     !606
      CLOSE(UNIT=2)      !607
      RETURN    !608
      END       !609
      SUBROUTINECALC20(CSTR,N1,N2,N3)   !614
      IMPLICITBYTE(B-C),INTEGER*2(I-N),REAL*4(A,E-H,O-Y),REAL*8(D),COMP      !615
     *LEX*8(Z)  !615
      BYTECSTR(81)       !616
      CALLBOPENF(CSTR,1,'old') !617
      CONTINUE !618
      I=1       !618
23133 IF(.NOT.(I .LE. 3 * (N1 -1 ) ))GOTO23135        !618
      CALLRGET(R)        !619
23134 I=I+1     !619
      GOTO23133          !619
23135 CONTINUE !619
      CALLRGET(X2A)      !620
      CALLRGET(Y2A)      !621
      CALLRGET(R)        !622
      CALLRGET(X1A)      !623
      CALLRGET(Y1A)      !624
      CALLRGET(R)        !625
      CLOSE(UNIT=1)      !626
      CALLBOPENF(CSTR,1,'old') !627
      CONTINUE !628
      I=1       !628
23136 IF(.NOT.(I .LE. 3 * (N2 -1 ) ))GOTO23138        !628
      CALLRGET(R)        !629
23137 I=I+1     !629
      GOTO23136          !629
23138 CONTINUE !629
      CALLRGET(X2B)      !630
      CALLRGET(Y2B)      !631
      CALLRGET(R)        !632
```

```
CALLRGET(X1B)     !633
CALLRGET(Y1B)     !634
CALLRGET(R)       !635
CLOSE(UNIT=1)     !636
THETA1=ATAN2(Y1A-Y1B,X1A-X1B)     !637
THETA2=ATAN2(Y2A-Y2B,X2A-X2B)     !638
THETA=THETA1-THETA2       !639
CALLRPRINT('The angle (radians) is ',THETA)      !641
CALLRPRINT('The angle (degrees) is ',THETA*180./3.14159265358979)          !642
CALLBOPENP(CSTR,1,3*(N3-1))       !643
CALLRPUT('',THETA)        !644
CALLRPUT('',THETA*180./3.14159265358979) !645
CALLRPUT('',0.)  !646
CLOSE(UNIT=1)     !647
```

```
      SUBROUTINECOPY(CSTR1,CSTR2,N)      !144
      DIMENSIONX(128)   !145
      OPEN(UNIT=1,ACCESS='DIRECT',RECORDSIZE=128,TYPE='OLD',NAME=CSTR1)              !147
      OPEN(UNIT=2,ACCESS='DIRECT',RECORDSIZE=128,TYPE='NEW',NAME=CSTR2)              !149
      DO 23002I=1,N      !150
      READ(1'I)X         !151
      WRITE(2'I)X        !152
23002 CONTINUE !153
23003 CONTINUE !153
      CLOSE(UNIT=1)      !154
      CLOSE(UNIT=2)      !155
      RETURN    !156
      END       !157
      SUBROUTINEOPNFNW(CSTR,IUNIT,I00011,ISZX,ISZY)       !160
      IMPLICITBYTE(B-C),INTEGER*2(I-N),REAL*4(A,E-H,O-Y),REAL*8(D),COMP              !161
     *LEX*8(Z)   !161
      BYTECPDB(512)      !162
      INTEGER*2IPDB(256)           !162
      REAL*4RPDB(128)    !162
      EQUIVALENCE(CPDB,IPDB,RPDB)         !162
      COMMON/PDBC/IPDB  !162
      BYTEC00012(81)     !163
      BYTECSTR(81)       !164
      I=INDEX(CSTR,' ')            !165
      I00013=2-1+FLOAT(ISZX)*FLOAT(ISZY)/64+0.999         !167
23001 OPEN(UNIT=IUNIT,NAME=CSTR,TYPE='NEW',ACCESS='DIRECT',RECORDSIZE=1              !169
     *28,INITIALSIZE=I00013,ASSOCIATEVARIABLE=I00011,BUFFERCOUNT=2)          !170
      RETURN    !171
      END       !172
      SUBROUTINEOPNFOD(CSTR,IUNIT,I00011,ISZX,ISZY)       !180
      IMPLICITBYTE(B-C),INTEGER*2(I-N),REAL*4(A,E-H,O-Y),REAL*8(D),COMP              !182
     *LEX*8(Z)   !182
      BYTECPDB(512)      !183
      INTEGER*2IPDB(256)           !183
      REAL*4RPDB(128)    !183
      EQUIVALENCE(CPDB,IPDB,RPDB)         !183
      COMMON/PDBC/IPDB !183
      BYTEC00012(81)     !184
      BYTECSTR(81)       !185
      I=INDEX(CSTR,' ')            !186
      OPEN(UNIT=IUNIT,NAME=CSTR,TYPE='OLD',ACCESS='DIRECT',RECORDSIZE=1              !188
     *28,ASSOCIATEVARIABLE=I00011,BUFFERCOUNT=2)          !189
      READ(IUNIT'1)IPDB            !190
      ISZX=IPDB(15)      !191
      ISZY=IPDB(16)      !192
      RETURN    !193
      END       !194
      SUBROUTINEVSVFIG(IUNIT)    !199
      IMPLICITBYTE(B-C),INTEGER*2(I-N),REAL*4(A,E-H,O-Y),REAL*8(D),COMP              !200
     *LEX*8(Z)   !200
      COMMON/VSVC/L00014,BHOLD,B00015    !201
      BYTEC00016(81)     !202
      COMMON/TMPSTC/C00016         !202
      BYTE C00012(21)   !203
C
      OPEN(UNIT=IUNIT,NAME='SY:VSVFIG.DAT',TYPE='OLD',CARRIAGECONTROL='              !205
     *LIST',ERR=23000)   !206
      READ(IUNIT,1)B00015          !207
1     FORMAT(L1)         !208
      IF(.NOT.(.FALSE.))GOTO23004         !209
23000 CALLGTLIN(C00016,C00012) !210
```

```
      IF(.NOT.(CUP(C00016(1)).EQ.'Y'))GOTO23006          !211
      B00015=.FALSE.      !212
      GOTO23007           !213
23006 CONTINUE !213
      B00015=.TRUE.       !214
23007 CONTINUE !214
      OPEN(UNIT=IUNIT,NAME='VSVFIG.DAT',TYPE='NEW',CARRIAGECONTROL='LIS      !215
     *T')           !215
      WRITE(IUNIT,1)B00015        !216
23004 CONTINUE !218
      CLOSE(UNIT=IUNIT)           !218
      RETURN     !219
      DATA C00012/73,115,32,116,104,101,114,101,32,97,32,118,115,118,49     !220
     *,49,32,63,32,-128,0/        !220
      END        !220
      LOGICALFUNCTIONCUP*1(CHAR)           !224
      IMPLICITBYTE(B-C),INTEGER*2(I-N),REAL*4(A,E-H,O-Y),REAL*8(D),COMP     !225
     *LEX*8(Z)   !225
C
      IF(.NOT.('a'.LE.CHAR.AND.CHAR.LE.'z'))GOTO23008   !227
      CUP=CHAR-32         !228
      GOTO23009           !229
23008 CONTINUE !229
      CUP=CHAR !230
23009 CONTINUE !230
      RETURN     !231
      END        !232
      LOGICALFUNCTIONBEGFLG*1(CSTR1,CSTR2)       !235
      IMPLICITBYTE(B-C),INTEGER*2(I-N),REAL*4(A,E-H,O-Y),REAL*8(D),COMP     !236
     *LEX*8(Z)   !236
      BYTECSTR1(2),CSTR2(2)       !237
C
      IF(.NOT.(CUP(CSTR1(1)).EQ.CUP(CSTR2(1)).AND.CUP(CSTR1(2)).EQ.CUP(     !239
     *CSTR2(2))))GOTO23010        !239
      BEGFLG=.TRUE.       !240
      GOTO23011           !241
23010 CONTINUE !241
      BEGFLG=.FALSE.      !242
23011 CONTINUE !242
      RETURN     !243
      END        !244
      SUBROUTINEVSVCLR(IWHAT)     !248
      IMPLICITBYTE(B-C),INTEGER*2(I-N),REAL*4(A,E-H,O-Y),REAL*8(D),COMP     !249
     *LEX*8(Z)   !249
      COMMON/VSVC/L00014,BHOLD,B00015    !250
C
      IF(.NOT.(BEGFLG(IWHAT,'vsv')))GOTO23012  !252
      B00015=.FALSE.      !253
23012 CONTINUE !254
      IF(.NOT.(BEGFLG(IWHAT,'no vsv')))GOTO23014         !254
      B00015=.TRUE.       !255
23014 CONTINUE !256
      IF(.NOT.(BEGFLG(IWHAT,'hold')))GOTO23016 !256
      BHOLD=.TRUE.        !257
23016 CONTINUE !258
      IF(.NOT.(BEGFLG(IWHAT,'continue')))GOTO23018       !258
      BHOLD=.FALSE.       !259
23018 CONTINUE !261
      IF(.NOT.(BEGFLG(IWHAT,'all').OR.BEGFLG(IWHAT,'characters')))GOTO2   !261
     *3020       !261
      L00014=1 !262
      IF(.NOT.(B00015))GOTO23022         !263
      CALLPRINT(0)        !264
```

```
      CALLPRINT(0)        !265
      RETURN    !266
23022 CONTINUE !268
      CALLIPOKE("172600+4,0)    !268
      DO 23024I=1,25    !269
      DO 23026J=1,64    !270
      CALLIPOKEB("172600,32)    !271
23026 CONTINUE !271
23027 CONTINUE !271
      CALLIPOKEB("172600,13)    !272
      CALLIPOKEB("172600,10)    !273
23024 CONTINUE !274
23025 CONTINUE !274
      CALLIPOKE("172600,"2000) !275
      CALLIPOKE("172600+4,0)    !276
23020 CONTINUE !278
      IF(.NOT.(B00015))GOTO23028          !278
      RETURN    !279
23028 CONTINUE !281
      IF(.NOT.(BEQFLG(IWHAT,'on')))GOTO23030    !281
      CALLIPOKE("172600,0)       !282
23030 CONTINUE !283
      IF(.NOT.(BEQFLG(IWHAT,'off')))GOTO23032   !283
      CALLIPOKE("172600,"2000) !284
23032 CONTINUE !286
      IF(.NOT.(BEQFLG(IWHAT,'all').OR.BEQFLG(IWHAT,'bit maps')))GOTO230          !286
     *34       !286
      CALLIPOKE("172620,IPEEK("172620).AND..NOT."400)   !287
      CALLIPOKE("172640,IPEEK("172640).AND..NOT."400)   !289
23034 CONTINUE !290
      RETURN    !290
      END       !291
      SUBROUTINEVSVSET(LINE,ICOL)          !296
      IMPLICITBYTE(B-C),INTEGER*2(I-N),REAL*4(A,E-H,O-Y),REAL*8(D),COMP          !297
     *LEX*8(Z)  !297
      COMMON/VSVC/L00014,BHOLD,B00015   !298
C
      L00014=LINE        !300
      IF(.NOT.(B00015))GOTO23036          !301
      RETURN    !302
23036 CONTINUE !303
      CALLIPOKE("172600+4,256*(LINE-1)+ICOL-1)  !303
      RETURN    !304
      END       !305
      SUBROUTINECPRINT(CSTR,CVAL,N)        !308
      IMPLICITBYTE(B-C),INTEGER*2(I-N),REAL*4(A,E-H,O-Y),REAL*8(D),COMP          !309
     *LEX*8(Z)  !309
      BYTECSTR(81),CVAL(81)       !310
      BYTEC00016(81)     !311
      COMMON/TMPSTC/C00016        !311
C
      CONTINUE !313
      K=1       !313
23038 IF(.NOT.(K .LE. 80 .AND. CSTR (K) .NE. 0 .AND. CSTR (K)            !313
     * .NE. 128 ))GOTO23040    !313
      C00016(K)=CSTR(K)           !314
23039 K=K+1     !314
      GOTO23038          !314
23040 CONTINUE !314
      C00016(K)=128      !315
      CALLVPRINT(C00016)          !316
      CONTINUE !317
      I=1       !317
```

```
23041   IF(.NOT.(I . LE . N ))GOTO23043  !317
        IF(.NOT.(CVAL(I).NE.' '))GOTO23044       !318
        GOTO23043          !319
23044   CONTINUE !320
23042   I=I+1      !320
        GOTO23041          !320
23043   CONTINUE !320
        CONTINUE !320
        J=1        !320
23046   IF(.NOT.(I . LE . N ))GOTO23048  !320
        C00016(J)=CVAL(I)        !321
23047   I=I+1      !321
        J=J+1      !321
        GOTO23046          !321
23048   CONTINUE !321
        C00016(J)=CSTR(K)        !322
        CALLVPRINT(C00016)       !323
        RETURN     !324
        END        !325
        SUBROUTINERPRINT(CSTR,RVAL)       !329
        IMPLICITBYTE(B-C),INTEGER*2(I-N),REAL*4(A,E-H,O-Y),REAL*8(D),COMP        !330
     *LEX*8(Z)   !330
        BYTECSTR(81)       !331
        BYTECRVAL(15)      !332
C
        ENCODE(15,1,CRVAL)RVAL   !334
1       FORMAT(G15.7,A00017)     !335
        CONTINUE !336
        I=15       !336
23049   IF(.NOT.(I . GT  1 . AND . (CRVAL (I ). EQ . 32 . OR . CRVAL (I        !336
     *). EQ  '0' ) ))GOTO23051 !336
        I=I-1      !337
23050   CONTINUE !337
        GOTO23049          !337
23051   CONTINUE !337
        CALLCPRINT(CSTR,CRVAL,I) !338
        RETURN     !339
        END        !340
        SUBROUTINEVPRINT(CSTR)   !344
        IMPLICITBYTE(B-C),INTEGER*2(I-N),REAL*4(A,E-H,O-Y),REAL*8(D),COMP        !345
     *LEX*8(Z)   !345
        BYTECSTR(81)       !346
        COMMON/VSVC/L00014,BHOLD,B00015  !347
        DATAB00015/.FALSE./,BHOLD/.FALSE./       !348
C
        CONTINUE !350
        I=1        !350
        N=0        !350
23052   IF(.NOT.(I . LE . 81 ))GOTO23054 !350
        IF(.NOT.(CSTR(I).EQ.128))GOTO23055       !351
        GOTO23054          !352
23055   CONTINUE !353
        IF(.NOT.(CSTR(I).EQ.0))GOTO23057 !353
        N=N+1      !354
        GOTO23054          !355
23057   CONTINUE !357
        IF(.NOT.(CSTR(I).EQ.10))GOTO23059        !357
        N=N+1      !358
23059   CONTINUE !359
23053   I=I+1      !359
        GOTO23052          !359
23054   CONTINUE !359
        IF(.NOT.(L00014.GT.25.OR.L00014+N.GE.25+2))GOTO23061       !360
```

```
      IF(.NOT.(BHOLD))GOTO23063          !361
      I00018=IPEEK("44)         !362
      CALLIPOKE("44,"10100.OR.I00018)   !363
      CALLVSVSET(25,62)         !364
      CALLVSVCLR('on') !365
      CONTINUE !366
23065 CONTINUE !366
23066 IF(.NOT.(ITTINR().EQ.13))GOTO23065          !367
23067 CONTINUE !367
      CALLVSVCLR('off')         !368
      I=ITTINR()        !369
      CALLIPOKE("44,I00018)     !370
      GOTO23064         !372
23063 CONTINUE !372
      CALLVSVCLR('characters') !373
23064 CONTINUE !373
      CALLVSVSET(1,1)   !374
23061 CONTINUE !376
      L00014=L00014+N   !376
      IF(.NOT.(B00015))GOTO23068         !377
      CALLPRINT(CSTR)   !378
      RETURN    !379
23068 CONTINUE !381
      CONTINUE !381
      I=1       !381
23070 IF(.NOT.(I  .  LE  .  80  .  AND  .  CSTR (I )  .  NE   0  .  AND  .  CSTR (I )          !381
     * . NE . 128 ))GOTO23072   !381
      CALLIPOKEB("172600,CSTR(I))       !382
23071 I=I+1     !382
      GOTO23070         !382
23072 CONTINUE !382
      IF(.NOT.(CSTR(I).EQ.0))GOTO23073 !383
      CALLIPOKEB("172600,13)    !384
      CALLIPOKEB("172600,10)    !385
23073 CONTINUE !387
      RETURN    !387
      END       !388
      SUBROUTINEASK(CSTR,C00012)        !393
      IMPLICITBYTE(B-C),INTEGER*2(I-N),REAL*4(A,E-H,O-Y),REAL*8(D),COMP          !394
     *LEX*8(Z)  !394
      BYTECSTR(81),C00012(81)   !395
      COMMON/ASKC/IDBASE,B00019         !396
      BYTEC00016(81)    !397
      COMMON/TMPSTC/C00016      !397
      BYTE CRLF(4)      !398
      BYTE CDEL(5)      !400
      DATAIDBASE/10/,B00019/.TRUE./     !401
C
      CALLRCTRLO        !403
      CALLZ200(C00012,C00016)   !404
      IF(.NOT.(B00019.AND.ISPY("366).LT.0))GOTO23075    !408
      CALLGTLIN(CSTR,C00016)    !409
      RETURN    !410
23075 CONTINUE !412
      CALLVPRINT(C00016)        !412
      CALLVSVCLR('on') !413
      I00018=IPEEK("44)         !414
      CALLIPOKE("44,"10100.OR.I00018)   !416
      CONTINUE !417
      J=1       !417
23077 IF(.NOT.(J . LE . 80 ))GOTO23079 !417
      CONTINUE !418
23080 CONTINUE !418
```

```
      I=ITTINR()          !419
      IF(.NOT.(I.EQ.127.AND.J.GT.1))GOTO23083   !420
      J=J-1      !421
      CALLVPRINT(CDEL) !422
23083 CONTINUE !424
23081 IF(.NOT.(I.GT.0.AND.I.NE.127))GOTO23080   !424
23082 CONTINUE !424
      IF(.NOT.(I.EQ.13))GOTO23085        !425
      GOTO23079           !426
      GOTO23086           !427
23085 CONTINUE !427
      CSTR(J)=I           !428
      CSTR(J+1)=128       !429
      CALLVPRINT(CSTR(J))        !430
23086 CONTINUE !431
23078 J=J+1      !432
      GOTO23077           !432
23079 CONTINUE !432
      CSTR(J)=0           !433
      CALLVPRINT(CRLF) !434
      CALLVSVCLR('off')          !435
      I=ITTINR()          !436
      CALLIPOKE("44,I00018)      !437
      RETURN     !438
      DATA CDEL/8,32,8,-128,0/ !439
      DATA CRLF/13,10,-128,0/  !439
      END        !439
      SUBROUTINEZ200(CSTR1,CSTR2)        !443
      IMPLICITBYTE(B-C),INTEGER*2(I-N),REAL*4(A,E-H,O-Y),REAL*8(D),COMP          !444
     *LEX*8(Z)  !444
      BYTECSTR1(81),CSTR2(81)    !445
C
      CONTINUE !447
      I=1        !447
23087 IF(.NOT.(I . LE . 80 . AND . CSTR1 (I ) . NE . 0 . AND . CSTR1 (I          !447
     * ) . NE . 128 ))GOTO23089 !447
      CSTR2(I)=CSTR1(I)          !448
23088 I=I+1      !448
      GOTO23087           !448
23089 CONTINUE !448
      IF(.NOT.(CSTR1(I).EQ.128))GOTO23090       !449
      CSTR2(I)=0          !450
      GOTO23091           !451
23090 CONTINUE !451
      CSTR2(I)=128        !452
23091 CONTINUE !452
      RETURN     !453
      END        !454
      LOGICALFUNCTIONBOPENF*1(CSTR,IUNIT,CTYPE)          !459
      IMPLICITBYTE(B-C),INTEGER*2(I-N),REAL*4(A,E-H,O-Y),REAL*8(D),COMP          !460
     *LEX*8(Z)  !460
      BYTECSTR(81),CTYPE(81)     !461
      BYTECSTR2(81)       !462
      COMMON/SFC/LUN,CSTR2       !462
      LUN=IUNIT           !464
      IF(.NOT.(CSTR(1).EQ.0))GOTO23092 !465
      CALLASK(CSTR2,'File name ? ')     !466
      GOTO23093           !467
23092 CONTINUE !467
      CALLSCOPY(CSTR,CSTR2,80) !468
23093 CONTINUE !468
      IF(.NOT.(INDEX(CSTR2,'.') EQ.0))GOTO23094          !469
      CALLCONCAT(CSTR2,'.DAT',CSTR2,80)          !470
```

```
23094  CONTINUE !471
       IF(.NOT.(BEQFLG(CTYPE,'new')))GOTO23096  !471
       OPEN(UNIT=LUN,NAME=CSTR2,TYPE='NEW',CARRIAGECONTROL='LIST')       !472
23096  CONTINUE !473
       IF(.NOT.(BEQFLG(CTYPE,'old')))GOTO23098  !473
       OPEN(UNIT=LUN,NAME=CSTR2,TYPE='OLD',CARRIAGECONTROL='LIST',ERR=1)          !474
       IF(.NOT.(.FALSE.))GOTO23100      !475
1      BOPENF=.FALSE.    !476
       RETURN    !477
23100  CONTINUE !479
23098  CONTINUE !480
       IF(.NOT.(BEQFLG(CTYPE,'unknown')))GOTO23102        !480
       OPEN(UNIT=LUN,NAME=CSTR2,TYPE='UNKNOWN',CARRIAGECONTROL='LIST')   !481
23102  CONTINUE !482
       BOPENF=.TRUE.      !482
       RETURN    !483
       END       !484
       LOGICALFUNCTIONBOPENP*1(CSTR,IUNIT,N)      !488
       IMPLICITBYTE(B-C),INTEGER*2(I-N),REAL*4(A,E-H,O-Y),REAL*8(D),COMP          !489
     *LEX*8(Z)  !489
       BYTECSTR(81)       !490
       REAL*4R(30)        !491
       BYTECSTR2(81)      !492
       COMMON/SFC/LUN,CSTR2      !492
       LUN=IUNIT          !494
       IF(.NOT.(CSTR(1).EQ.0))GOTO23104 !495
       CALLASK(CSTR2,'File name ? ')     !496
       GOTO23105          !497
23104  CONTINUE !497
       CALLSCOPY(CSTR,CSTR2,80) !498
23105  CONTINUE !498
       IF(.NOT.(INDEX(CSTR2,'.').EQ.0))GOTO23106         !499
       CALLCONCAT(CSTR2,'.DAT',CSTR2,80)          !500
23106  CONTINUE !501
       IF(.NOT.(N.GT.1))GOTO23108        !501
       OPEN(UNIT=LUN,NAME=CSTR2,TYPE='OLD',CARRIAGECONTROL='LIST',ERR=1)          !502
       IF(.NOT.(.FALSE.))GOTO23110      !503
1      BOPENP=.FALSE.    !504
       RETURN    !505
23110  CONTINUE !507
       CONTINUE !507
       I=1       !507
23112  IF(.NOT.(I . LE . MINO (N , 30 ) ))GOTO23114      !507
       CALLRGET(R(I))    !508
23113  I=I+1    !508
       GOTO23112          !508
23114  CONTINUE !508
       CLOSE(UNIT=LUN)   !509
23108  CONTINUE !511
       OPEN(UNIT=LUN,NAME=CSTR2,TYPE='NEW',CARRIAGECONTROL='LIST')       !511
       CONTINUE !512
       I=1       !512
23115  IF(.NOT.(I . LE . MINO (N , 30 ) ))GOTO23117      !512
       CALLRPUT('',R(I))         !513
23116  I=I+1     !513
       GOTO23115          !513
23117  CONTINUE !513
       BOPENP=.TRUE.      !514
       RETURN    !515
       END       !516
       SUBROUTINEIPRINT(CSTR,IVAL)        !520
       IMPLICITBYTE(B-C),INTEGER*2(I-N),REAL*4(A,E-H,O-Y),REAL*8(D),COMP          !521
     *LEX*8(Z)  !521
```

```
      BYTECSTR(81)        !522
      BYTECIVAL(6)        !523
C
      ENCODE(6,1,CIVAL)IVAL       !525
1     FORMAT(I6,A00017)           !526
      CALLCPRINT(CSTR,CIVAL,6)    !527
      RETURN    !528
      END       !529
      SUBROUTINEOPRINT(CSTR,IVAL)          !532
      IMPLICITBYTE(B-C),INTEGER*2(I-N),REAL*4(A,E-H,O-Y),REAL*8(D),COMP          !533
     *LEX*8(Z)   !533
      BYTECSTR(81)        !534
      BYTECOVAL(6)        !535
C
      ENCODE(6,1,COVAL)IVAL       !537
1     FORMAT(O6,A00017)           !538
      CALLCPRINT(CSTR,COVAL,6)    !539
      RETURN    !540
```

```
      SUBROUTINERPUT(CSTR,R)     !145
      IMPLICITBYTE(B-C),INTEGER*2(I-N),REAL*4(A,E-H,O-Y),REAL*8(D),COMP            !146
     *LEX*8(Z)   !146
      BYTECSTR(81)        !147
      BYTECSTR2(81)       !148
      COMMON/SFC/LUN,CSTR2        !148
      IF(.NOT.(CSTR(1).NE.0))GOTO23002 !150
      CALLRPRINT(CSTR,R)          !151
23002 CONTINUE !152
      WRITE(LUN,1)R       !152
1     FORMAT(G15.7)       !153
      RETURN    !154
      END       !155
      REALFUNCTIONRGET*4(R)       !159
      IMPLICITBYTE(B-C),INTEGER*2(I-N),REAL*4(A,E-H,O-Y),REAL*8(D),COMP            !160
     *LEX*8(Z)   !160
      BYTECSTR2(81)       !161
      COMMON/SFC/LUN,CSTR2        !161
      READ(LUN,1)R        !163
1     FORMAT(G15.7)       !164
      RGET=R    !165
      RETURN    !166
      END       !167
      SUBROUTINECZCVT   !173
      IMPLICITBYTE(B-C),INTEGER*2(I-N),REAL*4(A,E-H,O-Y),REAL*8(D),COMP            !174
     *LEX*8(Z)   !174
      COMMON/SZC/INPSZ,IOUPSZ,CIFORM,COFORM,I00011,I00012        !175
      BYTECPDB(512)       !176
      INTEGER*2IPDB(256)          !176
      REAL*4RPDB(128)   !176
      EQUIVALENCE(CPDB,IPDB,RPDB)          !176
      COMMON/PDBC/IPDB !176
      BYTECLINE1(256)   !177
      INTEGER*2LINE1(256),LINE2(256)    !177
      COMPLEX*8ZLINE1(256)        !177
      EQUIVALENCE(CLINE1,LINE1,ZLINE1) !177
      EQUIVALENCE(LINE2(256),ZLINE1(256))        !177
      COMMON/ARRAYC/ZLINE1        !177
      MAX=0     !179
      DO 23004J=1,IOUPSZ          !180
      CALLGETLNC          !181
      DO 23006I=1,IOUPSZ          !182
      ZLINE1(I)=CMPLX(FLOAT(LINE2(I)),0.)        !183
      MAX=MAX0(MAX,LINE2(I))    !184
23006 CONTINUE !185
23007 CONTINUE !185
      CALLOPLNZ1(ZLINE1)          !186
23004 CONTINUE !187
23005 CONTINUE !187
      RPDB(128)=FLOAT(MAX)        !188
      RETURN    !189
      END       !190
      SUBROUTINEGETLNC !194
      IMPLICITBYTE(B-C),INTEGER*2(I-N),REAL*4(A,E-H,O-Y),REAL*8(D),COMP            !200
     *LEX*8(Z)   !200
      COMMON/SZC/INPSZ,IOUPSZ,CIFORM,COFORM,I00011,I00012        !201
      BYTECLINE1(256)   !202
      INTEGER*2LINE1(256),LINE2(256)    !202
      COMPLEX*8ZLINE1(256)        !202
      EQUIVALENCE(CLINE1,LINE1,ZLINE1) !202
      EQUIVALENCE(LINE2(256),ZLINE1(256))        !202
```

```
      COMMON/ARRAYC/ZLINE1       !202
      DO 23008I=1,INPSZ          !204
      LINE2(I)=0         !205
23008 CONTINUE !205
23009 CONTINUE !205
      DO 23010K=1,I00011         !206
      CALLINPLNC(LINE2)          !207
23010 CONTINUE !207
23011 CONTINUE !207
      IF(.NOT.(I00011.GT.1))GOTO23012   !208
      CONTINUE !209
      I=1        !209
      I2=1       !209
23014 IF(.NOT.(I .LE . IOUPSZ ))GOTO23016      !209
      LINE2(I)=LINE2(I2)         !210
      DO 23017K=2,I00011         !211
      I2=I2+1   !212
      LINE2(I)=LINE2(I)+LINE2(I2)       !213
23017 CONTINUE !214
23018 CONTINUE !214
23015 I=I+1      !215
      I2=I2+1   !215
      GOTO23014          !215
23016 CONTINUE !215
23012 CONTINUE !217
      RETURN     !217
      END        !218
      SUBROUTINEINPSET(CSTR)     !222
      IMPLICITBYTE(B-C),INTEGER*2(I-N),REAL*4(A,E-H,O-Y),REAL*8(D),COMP          !223
     *LEX*8(Z)   !223
      COMMON/INPC/IUNITI,I00013,IPTRI,N00014     !224
      BYTECPDB(512)      !225
      INTEGER*2IPDB(256)         !225
      REAL*4RPDB(128)    !225
      EQUIVALENCE(CPDB,IPDB,RPDB)        !225
      COMMON/PDBC/IPDB !225
      COMMON/SZC/INPSZ,IOUPSZ,CIFORM,COFORM,I00011,I00012        !226
      COMMON/GAMC/IY,IFRAME,IGAMSZ,I00015,B00016         !227
      BYTECSTR(81)       !228
      DATAIUNITI/1/      !229
      CIFORM=CFORM(CSTR)         !231
      OPEN(UNIT=IUNITI,NAME=CSTR,TYPE='OLD',ACCESS='DIRECT',BUFFERCOUNT          !233
     *=2,ASSOCIATEVARIABLE=I00013,RECORDSIZE=128)        !234
      READ(IUNITI'1)IPDB         !235
      INPSZ=IPDB(15)     !236
      IF(.NOT.(CIFORM.EQ.'Z'))GOTO23019          !238
      N00014=2-1+FLOAT(INPSZ)*FLOAT(IPDB(16))/64+0.999 !240
      GOTO23020          !241
23019 CONTINUE !241
      N00014=2-1+FLOAT(INPSZ)*FLOAT(IPDB(16))/512+0.999          !243
23020 CONTINUE !243
      I00013=2 !244
      IPTRI=512+1        !245
      RETURN    !246
      END       !247
      SUBROUTINEINPLNC(LINE2)    !252
      IMPLICITBYTE(B-C),INTEGER*2(I-N),REAL*4(A,E-H,O-Y),REAL*8(D),COMP          !256
     *LEX*8(Z)   !256
      COMMON/INPC/IUNITI,I00013,IPTRI,N00014     !257
      COMMON/SZC/INPSZ,IOUPSZ,CIFORM,COFORM,I00011,I00012        !258
      INTEGER*2LINE2(256)        !259
      BYTECREC(512)      !260
      EQUIVALENCE(C,IC)          !261
```

```
      DATAIC/0/           !262
      DO 23021I=1,INPSZ          !264
      IF(.NOT.(IPTRI.GT.512))GOTO23023 !265
      IF(.NOT.(I00013.GT.N00014))GOTO23025      !266
      RETURN    !267
23025 CONTINUE !268
      READ(IUNITI'I00013)CREC   !268
      IPTRI=1   !269
23023 CONTINUE !271
      C=CREC(IPTRI)       !271
      LINE2(I)=LINE2(I)+IC       !272
      IPTRI=IPTRI+1       !273
23021 CONTINUE !274
23022 CONTINUE !274
      RETURN    !275
      END       !276
      SUBROUTINEOUPSET(CSTR)     !280
      IMPLICITBYTE(B-C),INTEGER*2(I-N),REAL*4(A,E-H,O-Y),REAL*8(D),COMP          !281
     *LEX*8(Z)   !281
      COMMON/OUPC/IUNITO,I00017,IPTRO,B00018     !282
      COMMON/SZC/INPSZ,IOUPSZ,CIFORM,COFORM,I00011,I00012         !283
      BYTECPDB(512)       !284
      INTEGER*2IPDB(256)         !284
      REAL*4RPDB(128)     !284
      EQUIVALENCE(CPDB,IPDB,RPDB)        !284
      COMMON/PDBC/IPDB !284
      COMMON/GAMC/IY,IFRAME,IGAMSZ,I00015,B00016          !285
      BYTECSTR(81)        !286
      DATAIUNITO/2/       !287
      COFORM=CFORM(CSTR)         !289
      I=INDEX(CSTR,'V')          !292
      IS=2-1+FLOAT(IOUPSZ)**2/64+0.999 !293
      OPEN(UNIT=IUNITO,NAME=CSTR,TYPE='NEW',ACCESS='DIRECT',RECORDSIZE=          !295
     *128,ASSOCIATEVARIABLE=I00017,BUFFERCOUNT=2,INITIALSIZE=IS)          !296
      I00017=2 !297
      IPTRO=1   !298
      IPDB(251)=0         !299
      IPDB(252)=0         !300
      IPDB(15)=IOUPSZ     !301
      IPDB(16)=IOUPSZ     !302
      IF(.NOT.(IOUPSZ.EQ.2*INPSZ))GOTO23027     !304
      B00018=.TRUE.       !305
      IOUPSZ=INPSZ        !306
      GOTO23028           !308
23027 CONTINUE !308
      B00018=.FALSE.      !309
23028 CONTINUE !309
      I00011=INPSZ/IOUPSZ        !310
      I00012=I00011**2 !311
      RETURN    !312
      END       !313
      SUBROUTINEOPLNZ1(ZLINE1) !318
      IMPLICITBYTE(B-C),INTEGER*2(I-N),REAL*4(A,E-H,O-Y),REAL*8(D),COMP          !319
     *LEX*8(Z)   !319
      COMMON/OUPC/IUNITO,I00017,IPTRO,B00018     !320
      COMPLEX*8ZLINE1(256)       !321
      CALLOLNZSR(ZLINE1)         !323
      IF(.NOT.(B00018))GOTO23029        !324
      CALLOUPLNO          !325
23029 CONTINUE !326
      RETURN    !326
      END       !327
      SUBROUTINEOLNZSR(ZLINE1) !331
```

```
      IMPLICITBYTE(B-C),INTEGER*2(I-N),REAL*4(A,E-H,O-Y),REAL*8(D),COMP    !332
     *LEX*8(Z)   !332
      COMMON/SZC/INPSZ,IOUPSZ,CIFORM,COFORM,I00011,I00012      !333
      COMMON/OUPC/IUNITO,I00017,IPTRO,B00018    !334
      COMPLEX*8ZRECO(64)          !335
      COMMON/ZRECOC/ZRECO         !335
      COMPLEX*8ZLINE1(256)        !336
      DO 23031I=1,IOUPSZ          !338
      ZRECO(IPTRO)=ZLINE1(I)      !339
      IPTRO=IPTRO+1     !340
      IF(.NOT.(IPTRO.GT.64))GOTO23033    !341
      WRITE(IUNITO'I00017)ZRECO          !342
      IPTRO=1    !343
23033 CONTINUE !345
23031 CONTINUE !345
23032 CONTINUE !345
      RETURN     !346
      END        !347
      SUBROUTINEOUPLNO !351
      IMPLICITBYTE(B-C),INTEGER*2(I-N),REAL*4(A,E-H,O-Y),REAL*8(D),COMP    !352
     *LEX*8(Z)   !352
      COMMON/SZC/INPSZ,IOUPSZ,CIFORM,COFORM,I00011,I00012      !353
      COMMON/OUPC/IUNITO,I00017,IPTRO,B00018    !354
      COMPLEX*8ZRECO(64)          !355
      COMMON/ZRECOC/ZRECO         !355
      DO 23035I=1,IOUPSZ          !357
      ZRECO(IPTRO)=(0.,0.)        !358
      IPTRO=IPTRO+1     !359
      IF(.NOT.(IPTRO.GT.64))GOTO23037    !360
      WRITE(IUNITO'I00017)ZRECO          !361
      IPTRO=1    !362
23037 CONTINUE !364
23035 CONTINUE !364
23036 CONTINUE !364
      RETURN     !365
      END        !366
      LOGICALFUNCTIONCFORM*1(CSTR)       !371
      IMPLICITBYTE(B-C),INTEGER*2(I-N),REAL*4(A,E-H,O-Y),REAL*8(D),COMP    !372
     *LEX*8(Z)   !372
      BYTECSTR(81)      !373
      I=INDEX(CSTR,'.')          !375
      IF(.NOT.(I.EQ.0))GOTO23039         !376
      CFORM=' '         !377
      GOTO23040         !378
23039 CONTINUE !378
      CFORM=CSTR(I+1)   !379
23040 CONTINUE !379
      RETURN     !380
      END        !381
      SUBROUTINEOUPPDB !385
      IMPLICITBYTE(B-C),INTEGER*2(I-N),REAL*4(A,E-H,O-Y),REAL*8(D),COMP    !389
     *LEX*8(Z)   !389
      COMMON/SZC/INPSZ,IOUPSZ,CIFORM,COFORM,I00011,I00012      !390
      COMMON/INPC/IUNITI,I00013,IPTRI,N00014    !391
      COMMON/OUPC/IUNITO,I00017,IPTRO,B00018    !392
      BYTECPDB(512)     !393
      INTEGER*2IPDB(256)         !393
      REAL*4RPDB(128)   !393
      EQUIVALENCE(CPDB,IPDB,RPDB)        !393
      COMMON/PDBC/IPDB !393
      IF(.NOT.(B00018))GOTO23041         !395
      DO 23043I=1,2*IOUPSZ       !396
      CALLOUPLNO        !397
```

```
23043   CONTINUE !397
23044   CONTINUE !397
23041   CONTINUE !398
        CLOSE(UNIT=IUNITI)         !398
        WRITE(IUNITO'1)IPDB        !399
        CLOSE(UNIT=IUNITO)         !400
        RETURN    !401
        END       !402
        SUBROUTINEINPLNZ(IUNIT,I00019,ZLINE,IY,ISZ,ZREC) !405
        IMPLICITBYTE(B-C),INTEGER*2(I-N),REAL*4(A,E-H,O-Y),REAL*8(D),COMP        !406
     *LEX*8(Z)  !406
        COMPLEX*8ZLINE(256),ZREC(64)       !407
        I00019=2+(ISZ/64)*(IY-1) !409
        IPTR=64+1         !410
        DO 23045I=1,ISZ   !411
        IF(.NOT.(IPTR.GT.64))GOTO23047     !412
        READ(IUNIT'I00019)ZREC     !413
        IPTR=1    !414
23047   CONTINUE !416
        ZLINE(I)=ZREC(IPTR)        !416
        IPTR=IPTR+1       !417
23045   CONTINUE !418
23046   CONTINUE !418
        RETURN    !419
        END       !420
        SUBROUTINEOUPLNZ(IUNIT,I00019,ZLINE,IY,ISZ,ZREC) !424
        IMPLICITBYTE(B-C),INTEGER*2(I-N),REAL*4(A,E-H,O-Y),REAL*8(D),COMP        !425
     *LEX*8(Z)  !425
        COMPLEX*8ZLINE(256),ZREC(64)       !426
        I00019=2+(ISZ/64)*(IY-1) !427
        IPTR=1    !428
        DO 23049I=1,ISZ   !429
        ZREC(IPTR)=ZLINE(I)        !430
        IPTR=IPTR+1       !431
        IF(.NOT.(IPTR.GT.64))GOTO23051     !432
        WRITE(IUNIT'I00019)ZREC    !433
        IPTR=1    !434
23051   CONTINUE !436
23049   CONTINUE !436
23050   CONTINUE !436
        RETURN    !437
        END       !438
        SUBROUTINEFFT(Z,N,D00010)          !442
        INTEGER*2N,I,J,K,TWOK      !443
        REAL*4S,D00010    !444
        COMPLEX*8Z(N),U,W,TEMP     !445
C
        IF(.NOT.(D00010.EQ.'inverse'.OR.D00010.EQ.'INVERSE'))GOTO23053    !447
        S=-3.14159265358979        !448
        TEMP=CMPLX(1./FLOAT(N),0.)         !449
        DO 23055I=1,N     !450
        Z(I)=TEMP*Z(I)    !451
23055   CONTINUE !451
23056   CONTINUE !451
        GOTO23054         !453
23053   CONTINUE !453
        S=3.14159265358979         !454
23054   CONTINUE !454
        CONTINUE !455
        I=1       !455
        J=1       !455
23057   IF(.NOT.(I . LT . N ))GOTO23059    !455
        IF(.NOT.(I.LT.J))GOTO23060         !456
```

```
      TEMP=Z(J)           !457
      Z(J)=Z(I)           !458
      Z(I)=TEMP           !459
23060 CONTINUE !463
      CONTINUE !463
      K=N/2    !463
23062 IF(.NOT.(K . LT . J ))GOTO23064  !463
      J=J-K    !464
23063 K=K/2    !464
      GOTO23062           !464
23064 CONTINUE !464
      J=J+K    !465
23058 I=I+1    !466
      GOTO23057           !466
23059 CONTINUE !466
      CONTINUE !468
      K=1      !468
      TWOK=2   !468
23065 IF(.NOT.(TWOK . LE . N ))GOTO23067        !468
      U=(1.,0.)           !469
      W=CEXP(CMPLX(0.,-S/FLOAT(K)))    !470
      DO 23068J=1,K       !472
      DO 23070I=J,N,TWOK          !473
      TEMP=Z(I+K)*U       !474
      Z(I+K)=Z(I)-TEMP    !475
      Z(I)=Z(I)+TEMP      !476
23070 CONTINUE !477
23071 CONTINUE !477
      U=U*W    !478
23068 CONTINUE !479
23069 CONTINUE !479
23066 K=TWOK   !480
      TWOK=2*TWOK         !480
      GOTO23065           !480
23067 CONTINUE !480
      RETURN   !481
      END      !482
      SUBROUTINETRANSP(IUNIT,I00019,ISZ)       !484
      IMPLICITBYTE(B-C),INTEGER*2(I-N),REAL*4(A,E-H,O-Y),REAL*8(D),COMP       !485
     *LEX*8(Z)  !485
      COMPLEX*8ZLINE(512)         !486
      COMPLEX*8ZLINE1(256),ZLINE2(256) !486
      COMPLEX*8ZBUFF(64,4)        !486
      COMPLEX*8ZREC1(64),ZREC2(64)     !486
      REAL*4RLINE1(512),RLINE2(512)    !486
      BYTECLINE1(256),CLINE2(256)      !486
      BYTECREC1(512),CREC2(512)        !486
      EQUIVALENCE(ZLINE,ZLINE1,RLINE1,ZBUFF)    !486
      EQUIVALENCE(ZLINE(257),ZLINE2,RLINE2)     !486
      EQUIVALENCE(ZREC1,CREC1),(ZREC2,CREC2)    !486
      COMMON/ZC/ZREC1,ZLINE,ZREC2      !486
      BYTECPDB(512)       !487
      INTEGER*2IPDB(256)          !487
      REAL*4RPDB(128)     !487
      EQUIVALENCE(CPDB,IPDB,RPDB)      !487
      COMMON/PDBC/IPDB !487
      CONTINUE !489
      KSZ=ISZ/2           !489
23072 IF(.NOT.(KSZ . GE . 1 ))GOTO23074         !489
      CONTINUE !490
      JK=1     !490
23075 IF(.NOT.(JK . LE . ISZ ))GOTO23077        !490
      CONTINUE !491
```

```
      J=JK       !491
23078 IF(.NOT.(J . LE . JK + KSZ -1 ))GOTO23080              !491
      CALLINPLNZ(IUNIT,I00019,ZLINE1,J,ISZ,ZREC1)            !492
      CALLINPLNZ(IUNIT,I00019,ZLINE2,J+KSZ,ISZ,ZREC2)        !493
      DO 23081IK=1,ISZ,2*KSZ    !494
      DO 23083I=IK,IK+KSZ-1     !495
      Z=ZLINE1(I+KSZ)   !496
      ZLINE1(I+KSZ)=ZLINE2(I)   !497
      ZLINE2(I)=Z        !498
23083 CONTINUE !499
23084 CONTINUE !499
23081 CONTINUE !499
23082 CONTINUE !499
      CALLOUPLNZ(IUNIT,I00019,ZLINE1,J,ISZ,ZREC1)            !500
      CALLOUPLNZ(IUNIT,I00019,ZLINE2,J+KSZ,ISZ,ZREC2)        !501
23079 J=J+1      !502
      GOTO23078          !502
23080 CONTINUE !502
23076 JK=JK+2*KSZ        !502
      GOTO23075          !502
23077 CONTINUE !502
23073 KSZ=KSZ/2          !502
      GOTO23072          !502
23074 CONTINUE !502
      CPDB(7)=CPDB(7).XOR.16    !503
```

Appendix 2

Tutorial for using the programs for VAX System

Put the reference picture OCR1.PIC in the RK1: of RT-11 at Man-Vehicle Laboratory. PUt the program disc which includes YTFS1.RAT which is for defining the subpicture location. If edit is necessary for this program, compile this program according the following procedure.

```
R RATFOR
* YTFS1.RAT
* ^C
FORT/NOLINENUMBER YTFS1.FOR
LINK YTFS1,TTT4,BOX
BO RT11BL
```

Get the reference picture on the video screen by the next command.

```
COPY RK1:OCR1.PIC VD:
```

If VD: is not installed, install VD: by this command

```
INSTALL VD:
```

and copy OCR1.PIC on the video screen. DAT: is assaigned to RK1: by BOOT command. Run the program by the next command.

```
RUN YTFS1
```

You are asked to define the subpicture location. Move the box on the video screen by analog input and hit the space key to define the subpicture locatioin. The data is stored DAT:A.DAT and DAT:B.DAT. You must transfer this data into the VAX System at Tufts Medical School Image Analysis Laboratory.

You must transform the picture files OCR1.PIC and OCR2.PIC into transmittable format to Tufts VAX System. Put these picture data file in RK1: of RT-11 and run the program MITFS which is in the program disc in RK2:. Out put files are OCR1.TFS and OCR2.TFS. These files can be transfered to Tufts VAX System.

To use the VAX System from MIT, connect the terminal to the telephone line by the modem. The telephone number of Tufts VAX are

956-7474,7475,7476,7477,7478,7479

User name is SPACEYE. Password is *******. Assign DAT: by the following command.

```
ASS DR1:[SPACEYE] DAT:
```

All prograns of YTFS2.FOR, YTFS3.FOR, YTFS4.FOR and TFSPIC.FOR must be in [SPACEYE]. If edit is necessary, compile the program according to the following procedure.

```
FORT YTFS2.FOR
LINK YTFS2
```

You must reconstruct the picture data file from the transmittable format into the original picture data format. This is done by the program TFSPIC.FOR. This program transform only one data file. So, you must edit the name of the file and again run the program to get the two picture files. (Calling subroutine twice in the main part of this program caused an error. I don't know the reason.)

```
RUN TFSPIC
```

And you will get the original picture data files.

Run YTFS2 first to get necessary data files for using YTFS3 by the command

```
RUN YTFS2
```

and run the main program for the calculation of rotation angle by the command

```
RUN YTFS3
```

The rotation data is stored in the file OCRBOX.DAT.

```
/*********************************************************************************
*       PROGRAM YTFS1.RAT
*       13-Feb-85             Rev A
*       Yoshihiro Nagashima
*       This program defines the x-y coodinate of subpicture location and
*               data are stored in data file DAT:A.DAT and DAT:B.DAT
*       Caution:When you define the location of subpictures, please don't use
*               return key. Return key will make an error. Use space key, etc.
*
*       R RATFOR
*       * YTFS1.RAT
*       * ^C
*       FORT/NOLINENUMBER YTFS1
*       LINK YTFS1,TTT4,BOX
*       BO RT11BL
*       RUN YTFS1
*********************************************************************************

define MAX_ENTRIES          30
define FMT_STAT             format(g15.7)
define UNIT_1               1        # Logical unit for file 1
define PI                   3.14159265358979


define TP_IMPLICIT implicit byte(b-c),integer*2(i-n),real*4(a,e-h,o-y),\
                          real*8(d),complex*8(z)
define CHARACTER            byte
define COM_SAVE_FILE        CHARACTER cstr2(81),\
                            common /sfc/lun,cstr2

##################################################################################
##################################################################################
# main program
##################################################################################
##################################################################################


call PRINT('Please define subpicture A by the space key')
call box(ix,iy,64,64)              # define subpicture A location

x=float(ix)
y=float(iy)
call bopenf('DAT:A.DAT',UNIT_1,'new')
call yrput(x)
call yrput(y)
close(unit=UNIT_1)

call PRINT('Please define subpicture B')
call box(ix,iy,64,64)              # define subpicture B location

x=float(ix)
y=float(iy)
call bopenf('DAT:B.DAT',UNIT_1,'new')
call yrput(x)
call yrput(y)
```

```
/*************************************************************************************
*       PROGRAM YTFS2.RAT
*       13-Feb-85          Rev.A
*       Yoshihiro Nagashima
*       This program makes necessary data files which are used by YTFS3.RAT
*       Inputs: ocr1.pic           Reference raw picture 256*256 byte dimension
*               a.dat              x-y coodinate data of subpicture A
*               b.dat              x-y coodinate data of subpicture B
*       Outputs:ocr1.pic
*               a1.z               subpicture A1 extracted from ocr1.z
*                                  filtered with Mexican Hat
*                                  region of interest is also defined
*                                  ready to use in YTF3.RAT
*               b1.z               subpicture B1 extracted from ocr1.z
*                                  filtered with Mexican Hat
*                                  region of interest is also defined
*                                  ready to use in YTF3.RAT
*               one.z              ROI mask
*               mhf.z              FFT of Mexican Hat Filter
*               msk.z              mask of MHF
*
*       R RATFOR
*       * YTFS2.RAT
*       * ^C
*       FORT/NOLINENUMBER YTFS2
*       LINK YTFS2,YTFS4
*       DO RT11BL
*       RUN YTFS2
*************************************************************************************/

define MAX_ENTRIES        30
define FMT_STAT           format(g15.7)
define UNIT_1             1         # Logical unit for file 1
define PI                 3.14159265358979
define TP_IMPLICIT implicit byte(b-c),integer*2(i-n),real*4(a,e-h,o-y),\
                          real*8(d),complex*8(z)
define CHARACTER          byte
define COM_SAVE_FILE      CHARACTER cstr2(21),\
                          common /sfc/lun,cstr2

#####################################################################################
#####################################################################################
# main program
#####################################################################################
#####################################################################################

complex*8 ONEZ(64,64)
complex*8 A1Z(64,64)
complex*8 B1Z(64,64)
complex*8 MHZ(64,64)
complex*8 MHFZ(64,64)
complex*8 MSKZ(64,64)


call PRINT('I am making a mask file')
call ymaksr(ONEZ,64,64,1.,32,32,10,10)         # make mask
                                               # mask name is ONEZ (array in
                                               # buffer)
                                               # mask dimension is 64*64
                                               # magnitude is 1.
```

```
                                                  # ROI location is (32,32)
                                                  # ROI dimension is 10*10

call PRINT('I am making subpicture A1Z')
call ysbpic(A1Z,'DAT:OCR1.PIC','DAT:A.DAT')

call PRINT('I am making subpicture B1Z')
call ysbpic(B1Z,'DAT:OCR1.PIC','DAT:B.DAT')

#####################################################################
# mexican hat
#####################################################################

call PRINT('making Mexican hat filter')
call ymksr8(MHZ,64,64,1.,32.,32.,0.7071068)
                                                  # make mexican hat filter
                                                  # MHZ is array name of filter
                                                  # 64*64 dimension
                                                  # magnitude is 1
                                                  # center is (32.,32.)
                                                  # sigma is 0.7071068

call yrotil(MHZ,MHFZ,-32,-32)                     # rotation mask
                                                  # new data array is MHFZ


call PRINT('FFT of the Mexican hat filter')
call yxform(10,MHFZ)                              # option 10 is FFT
                                                  # output is MHFZ ( FFT of
                                                  # Mexican Hat Filter )

call ykeep(MHFZ,'DAT:MHF.Z')
call PRINT('making edge mask of filter')
call ymaksr(MSKZ,64,64,1.,32.32.28,28)
                                                         # make edge mask of
                                                         # mexican hat filter
                                                         # mask dimension is
                                                         # 64*64
call ykeep(MSKZ,'DAT.MSK.Z')



#####################################################################
# mexican hat
#####################################################################

call PRINT('FFT of subpictures')
call yxform(10,A1Z)                          # FFT of A1Z
call yxform(10,B1Z)                          # FFT of B1Z

call PRINT('filtering subpictures A1Z B1Z with Mexican Hat Filter')
call y2fil(MHFZ,A1Z,20,0.0001)                    # option 20 is z2=z1*z2
                                                  # z1 is MHFZ
                                                  # z2 is A1Z
call y2fil(MHFZ,B1Z,20,0.0001)                    # option 20 is z2=z1*z2
                                                  # z1 is MHFZ
                                                  # z2 is B1Z
call PRINT('FFT[-1] of filtered subpictures')
call yxform(11,A1Z)                               # FFT[-1] of A1Z
call yxform(11,B1Z)                               # FFT[-1] of B1Z

call PRINT('edge masking of subpictures')
```

```
call y2fil(MSKZ,A1Z,20,0.0001)                  # option 20 is z2=z1*z2
                                                # z1 is MSKZ
                                                # z2 is A1Z
call y2fil(MSKZ,B1Z,20,0.0001)                  # option 20 is z2=z1*z2
                                                # z1 is MSKZ
                                                # z2 is B1Z

####################################################################################
# end of mexican hat filter
####################################################################################


call PRINT('making ROI from subpictures A1Z & B1Z')
call ynormf(ONEZ,A1Z)              # normalize A1Z with ONEZ
call ynormf(ONEZ,B1Z)              # normalize B1Z with ONEZ


###################################################################################
# ocrpc
####################################################################################

call PRINT('I am tired')

call yxform(10,A1Z)                             # FFT
call yxform(10,B1Z)                             # FFT
call yxform(10,ONEZ)                            # FFT

call ykeep(A1Z,'DAT:A1.Z')
call ykeep(B1Z,'DAT:B1.Z')
call ykeep(ONEZ,'DAT:ONE.Z')

end
####################################################################################
####################################################################################
# end of main program
```

```
/******************************************************************************
*       PROGRAM YTFS3.RAT
*       13-Feb-85          Rev. A
*       Yoshihiro Nagashima
*       This program calculates the rotation angle of a data picture using
*                some data file
*       Inputs: ocr2.pic            raw data picture 256*256 byte dimension
*               a1.z                subpicture A1 extracted from OCR1.PIC (
*                                   reference picture )
*               b1.z                subpicture B1 extracted from OCR1.PIC (
*                                   reference picture )
*               a.dat               x-y coodinate of location of subpicture A1
*               b.dat               x-y coodinate of location of subpicture B1
*               mhf.z               Mexican Hat Filter ( already fft'ed )
*               msk.z               edge mask
*       Outputs:ocr2.pic
*               a1.z                subpicture A1 extracted from OCR1.PIC
*               b1.z                subpicture B1 extracted from OCR1.PIC
*               a.dat               x-y center data of subpicture A
*               b.dat               x-y center data of subpicture B
*               mhf.z               FFT of Mexican Hat Filter
*               msk.z               mask of MHF
*               one.z               ROI mask
*               ocrbox.dat          the two data are theta ( radian ) and
*                                   theta ( degree )
*
*       R RATFOR
*       * YTFS3.RAT
*       * ^C
*       FORT/NOLINENUMBER YTFS3
*       LINK YTFS3,YTFS4
*       BO RT11BL
*       RUN YTFS3
******************************************************************************/


define MAX_ENTRIES          30
define FMT_STAT             format(g15.7)
define UNIT_1               1         # Logical unit for file 1
define PI                   3.14159265358979
define TP_IMPLICIT implicit byte(b-c),integer*2(i-n),real*4(a,e-h,o-y),\
                                 real*8(d),complex*8(z)
define CHARACTER            byte
define COM_SAVE_FILE        CHARACTER cstr2(81);\
                            common /sfc/lun,cstr2

##############################################################################
##############################################################################
# main program
##############################################################################
##############################################################################

complex*8 A2Z(64,64)
complex*8 B2Z(64,64)
complex*8 MHFZ(64,64)
complex*8 MSKZ(64,64)
complex*8 AA2Z(64,64)
complex*8 BB2Z(64,64)
complex*8 ONEZ1(64,64)
complex*8 ONEZ2(64,64)
complex*8 A1Z1(64,64)
```

```
complex*8 B1Z1(64,64)

call PRINT('I am making subpicture A2Z')
call ysbpic(A2Z, 'DAT:OCR2.PIC', 'DAT:A.DAT')
call PRINT('I am making subpicture B2Z')
call ysbpic(B2Z, 'DAT:OCR2.PIC', 'DAT:B.DAT')

##############################################################################
# mexican hat
##############################################################################

call PRINT('FFT of subpictures')
call yxform(10,A2Z)                      # FFT of A2Z
call yxform(10,B2Z)                      # FFT of B2Z

call PRINT('filtering subpictures A2Z B2Z with Mexican Hat Filter')
call filbuf('DAT:MHF.Z',MHFZ)                   # get data from file into buf
call y2fil(MHFZ,A2Z,20,0.0001)                  # option 20 is z2=z1*z2
                                                # z1 is MHFZ
                                                # z2 is A2Z
call y2fil(MHFZ,B2Z,20,0.0001)                  # option 20 is z2=z1*z2
                                                # z1 is MHFZ
                                                # z2 is B2Z

call PRINT('FFT[-1] of filtered subpictures')
call yxform(11,A2Z)                             # FFT[-1] of A2Z
call yxform(11,B2Z)                             # FFT[-1] of B2Z

call PRINT('edge masking of subpictures')
call filbuf('DAT:MSK.Z',MSKZ)                   # get data from file into buf
call y2fil(MSKZ,A2Z,20,0.0001)                  # option 20 is z2=z1*z2
                                                # z1 is MSKZ
                                                # z2 is A2Z
call y2fil(MSKZ,B2Z,20,0.0001)                  # option 20 is z2=z1*z2
                                                # z1 is MSKZ
                                                # z2 is B2Z

##############################################################################
# end of mexican hat filter
##############################################################################


##############################################################################
# ocrpc
##############################################################################

call PRINT('correlating')
call copy(A2Z,AA2Z,64)             # copy recordsize is 64
call copy(B2Z,BB2Z,64)             # copy recordsize is 64

call y2fil(A2Z,AA2Z,20,0.0001)            # op20 is z2=z1*z2
call y2fil(B2Z,BB2Z,20,0.0001)            # op20 is z2=z1*z2

call PRINT('FFT for correlating')
call yxform(10,A2Z)                             # FFT
call yxform(10,B2Z)                             # FFT
call yxform(10,AA2Z)                            # FFT
call yxform(10,BB2Z)                            # FFT

call PRINT('I am tired')

call filbuf('DAT:A1.Z',A1Z1)              # get data from file into buffer
call filbuf('DAT:B1.Z',B1Z1)              # get data from file into buffer
```

```
call PRINT('Help me , Yoshi !')
call y2fil(A2Z,A1Z1,21,0.0001)              # op21 is z2=z1*conjg(z2)
call y2fil(B2Z,B1Z1,21,0.0001)              # op21 is z2=z1*conjg(z2)

call filbuf('DAT:ONE.Z',ONEZ1)              # get data from file into buffer
call copy(ONEZ1,ONEZ2,64)                   # copy

call y2fil(AA2Z,ONEZ1,21,0.0001)            # z2=z1*conjg(z2)
call y2fil(BB2Z,ONEZ2,21,0.0001)            # z2=z1*conjg(z2)

call PRINT('FFT[-1]')
call yxform(11,A1Z1)                                  # op 11 is FFT[-1]
call yxform(11,B1Z1)                                  # op 11 is FFT[-1]
call yxform(11,ONEZ1)                                 # op 11 is FFT[-1]
call yxform(11,ONEZ2)                                 # op 11 is FFT[-1]

call PRINT('square root')
call y1fil(ONEZ1)                           # square root
call y1fil(ONEZ2)                           # square root

call PRINT('almost done')
call y2fil(ONEZ1,A1Z1,23,0.0001)            # z2=z2/z1
call y2fil(ONEZ2,B1Z1,23,0.0001)            # z2=z2/z1

call PRINT('finding peak')

call ypeak4(A1Z1,x2a,y2a,rn1)               # find peak

call datatr('DAT:A.DAT',x1a,y1a)            # get x-y coodinate of first location
                                            # of subpic A

call ypeak4(B1Z1,x2b,y2b,rn2)               # find peak


call datatr('DAT:B.DAT',x1b,y1b)            # get x-y coodinate of first location
                                            # of subpic B

call yclc20(x2a,y2a,x1a,y1a,x2b,y2b,x1b,y1b,'DAT:OCRBOX.DAT',1)
                                            # calculation of angle
end
##########################################################################################
##########################################################################################
# end of main program
```

```
/*****************************************************************************
*       PROGRAM YTFS4.RAT
*       13-Feb-85         Rev.A
*       Yoshihiro Nagashima
*       This program contains the subroutines which are to be linked with
*       YTFS1,YTFS2 and YTFS3.
*       Included subroutines
*               ymaksr
*               ysbpic
*               ymksr8
*               yrot11
*               yxform
*               y2fil
*               ynormf
*               y1fil
*               ypeak4
*               datatr
*               yclc20
*               copy
*               bopenf
*               bopenp
*               ykeep
*               filbuf
*               cup
*               beqflg
*               yrput
*               rget
*               fft
*               ytrnsp
*
*       R RATFOR
*       * YTFS4.RAT
*       * ^C
*       FORT/NOLINENUMBER YTFS4
*****************************************************************************/

define MAX_ENTRIES      30
define FMT_STAT         format(g15.7)
define UNIT_1           1       # Logical unit for file 1
define PI               3.14159265358979


define TP_IMPLICIT implicit byte(b-c),integer*2(i-n),real*4(a,e-h,o-y),\
                        real*8(d),complex*8(z)
define CHARACTER        byte
define COM_SAVE_FILE    CHARACTER cstr2(81);\
                        common /sfc/lun,cstr2

##############################################################################
# main subroutines
##############################################################################


subroutine ymaksr(zsub,iszx,iszy,rmag,ix0,iy0,ixc,iyc)  # make rectangle mask

# zsub is dummy array in buffer  actual dimension must be defined in main
#       program.
complex*8 zsub(64,64),zmag
```

```
zmag=cmplx(rmag,0.)
do i=1,iszx [
        do j=1,iszy [
                zsub(i,j)=(0.,0.)
                if(((j.ge.iy0-iyc).and.(j.le.iy0+iyc)).and. \
                        ((i.ge.ix0-ixc).and.(i.le.ix0+ixc)))
                        zsub(i,j)=zmag
                ]
        ]
return
end
```

```
subroutine ysbpic(zsub,cstr1,cstr2)
#       this subroutine makes subpictures of 64*64 byte dimension and
#               64*64 complex*8 dimension from 256*256 byte dimension
#               raw picture.
#       cstr1 is 256*256 dimension original raw picture. and its data
#               type is bytes
#       cstr2 is x-y coodinate data of subpicture
#       cstr3 is 64*64 subpicture, and its data type is bytes.
#       cstr4 is 64*64 subpicture, but its data type is comblex*8.
#       zsub is a dummy array stored in buffer  dimension of actual array
#               must be defined in main program

character *(*) cstr1,cstr2
byte crec(512),c
byte cpic(256,256)
byte csub(64,64)
integer*2 isub(64,64),ic
complex*8 zsub(64,64)
complex*8 zrec(64)
equivalence (c,ic)
data ic/0/

call bopenf(cstr2,2,'old')        # 2 is unit number
call rget(x0)             # ( x0,y0 ) is the center of subpic
call rget(y0)
close(unit=2)
x0=x0-32.                 # ( x0,y0 ) is the upper left of subpic
y0=y0-32.
ix0=ifix(x0)
iy0=ifix(y0)
if(ix0.le.0)
        ix0=ix0+256
if(iy0.le.0)
        iy0=iy0+256
open(unit=1,access='DIRECT',recordsize=128,type='OLD', \
        name=cstr1)
ii=1
jj=1
do i=1,128 [
        read(1%'i) crec
        do j=1,512 [
                cpic(ii,jj)=crec(j)
                jj=jj+1
                if(jj.gt.256) [
```

```
                        ii=ii+1
                        jj=1
                        ]
                ]
        ]

k=1
l=1
if((iy0+63).le.256) [
        do i=iy0,iy0+63 [
                do j=ix0,ix0+63 [
                        j2=j
                        if(j2.gt.256)
                                j2=j2-256
                        csub(k,l)=cpic(i,j2)
                        l=l+1
                        ]
                l=1
                k=k+1
                ]
        ]

k=1
l=1
if((iy0+63).gt.256) [
        do i=1,256 [
                if(((i.ge.1).and.(i.le.iy0+63-256)).or. \
                ((i.ge.iy0).and.(i.le.256))) [
                        l=1
                        do j=ix0,ix0+63 [
                                j2=j
                                if(j2.gt.256)
                                        j2=j2-256
                                csub(k,l)=cpic(i,j2)
                                l=l+1
                                ]
                        l=1
                        k=k+1
                        ]
                ]
        ]
close(unit=1)
# byte-complex conbert
do i=1,64 [
        do j=1,64 [
                c=csub(i,j)
                isub(i,j)=ic
                zsub(i,j)=cmplx(float(isub(i,j)),0.)     # c-z convert
                ]
        ]
return
end




subroutine ymksr8(zsub,iszx,iszy,rmag,x0,y0,sigma)
# make mexican hat filter
FP_IMPLICIT
complex*8 zsub(64,64)
```

```
# zsub is dummy array in buffer. actual dimension must be defined in main
#       program.
zmag=cmplx(rmag,0.)
xmax=amin1(x0-1.,float(iszx)-x0)
ymax=amin1(y0-1.,float(iszy)-y0)
var_inverse=1./sigma**2
twovar_inverse=1./(2.*sigma**2)
sigma_4_2_pi_inverse=1./(sigma**4*sqrt(2.*pi))
for(i=1 ; i.le.iszy ; i=i+1) [
        do j=1,iszx [
                rsq=(float(j)-x0)**2+(float(i)-y0)**2
                r=sqrt(rsq)
                zsub(i,j)=cmplx(sigma_4_2_pi_inverse* \
                        (2.-var_inverse*rsq)*exp(-twovar_inverse*rsq),0.)
                ]
        ]
return
end




subroutine yrot11(zsub1,zsub2,ix1,iy1)
# rotate or translate a picture
# z2=circular_integer_translation(z1)
# z1 is zsub1
# z2 is zsub2
# x translation is ix
# y translation is iy
complex*8 zsub1(64,64),zsub2(64,64)
ix=ix1
iy=iy1
if(ix.lt.0)
        ix=64+ix
if(iy.lt.0)
        iy=64+iy
for(i=1 ; i.le.64 ; i=i+1) [
        do j=1,64
                zsub2(i,(mod(j-1+ix,64)+1))=zsub1(i,j)
        ]
return
end


subroutine yxform(iop,zsub)
# FFT and FFT[-1]
# iop 10 is FFT
# iop 11 is FFT[-1]
complex*8 zsub(64,64),zline(64)
for(i=1 ; i.le.64 ; i=i+1) [
        do j=1,64
                zline(j)=zsub(i,j)
        if(iop.eq.10)
                call fft(zline,64,'forw')
        else
                call fft(zline,64,'inve')
        do j=1,64
                zsub(i,j)=zline(j)
        ]
```

```
call ytrnsp(zsub)
for(i=1 ; i.le.64 ; i=i+1) [
        do j=1,64
                zline(j)=zsub(i,j)
        if(iop.eq.10)
                call fft(zline,64,'forw')
        else
                call fft(zline,64,'inve')
        do j=1,64
                zsub(i,j)=zline(j)
        ]
return
end


subroutine y2fil(zsub1,zsub2,iop,rmin)
# It performs and element by element operation on the files.
# To avoid divide by 0, when real(z1) is less than rmin, the result is set to
# 0. rmin is defined as a fraction of the maximum value, rmax, in z1
# iop 20 is z2=z2*z1
# iop 21 is z2=z1*conjg(z2)
# iop 23 is z2=z2/z1
# zsub1,zsub2 are dummy arrays in buffer. actual dimension must be defined in
# main program.
complex*8 zsub1(64,64),zsub2(64,64)
if(iop.eq.23) [
        rmax=0.
        do i=1,64 [
                do j=1,64 [
                        rmax=amax1(rmax,abs(real(zsub2(i,j))))
                        ]
                ]
        rmin1=abs(rmax*rmin)
        ]
do i=1,64 [
        if(iop.eq.20)
                do j=1,64
                        zsub2(i,j)=zsub1(i,j)*zsub2(i,j)
        if(iop.eq.21)
                do j=1,64 [
                        zsub2(i,j)=zsub1(i,j)*conjg(zsub2(i,j))
                        ]
        if(iop.eq.23)
                do j=1,64
                        if(abs(real(zsub1(i,j))).ge.rmin1)
                                zsub2(i,j)=cmplx(real(zsub2(i,j))/real(zsub1(i,j)),0.)
                        else
                                zsub2(i,j)=(0.,0.)
        ]
return
end



subroutine ynormf(zsub1,zsub2)
TP_IMPLICIT
complex*8 zsub1(64,64),zsub2(64,64),zsum,zmean
for([i=1;npel=0;zsum=(0.,0.);sumsq=0 ] ; i.le.64 ; i=i+1) [
        do j=1,64
                if(real(zsub1(i,j)).ne.0.) [
                        npel=npel+1
                        zsum=zsum+zsub2(i,j)
```

```
                                sumsq=sumsq+cabs(zsub2(i,j))
                                ]
        ]
if(npel.eq.0) [
        call PRINT('Unable to execute')
        return
        ]
zmean=zsum/npel
rootsq_inverse=1./sqrt(sumsq-cabs(zsum)/npel)
for(i=1 ; i.le.64 ; i=i+1) [
        do j=1,64
                if(real(zsub1(i,j)).ne.0.)
                        zsub2(i,j)=(zsub2(i,j)-zmean)*rootsq_inverse
                else
                        zsub2(i,j)=(0.,0.)
        ]
return
end




subroutine yifil(zsub)
complex*8 zsub(64,64)
do i=1,64 [
        do j=1,64
                zsub(i,j)=cmplx(sqrt(amax1(0.,real(zsub(i,j)))),0.)
        ]
return
end



subroutine ypeak4(zsub,x,y,rn)
complex*8 zsub(64,64),zsub2(64,64)
real*4 f(3,3),temp(3)
t0(f1,f2,f3)=(f1-f3)/(2.*(f1-2.*f2+f3))
f_of_t(t0,f1,f2,f3)=(f1-2.*f2+f3)/2.*t0**2+(f3-f1)/2.*t0+f2
for([i=1;x=0.;y=0.;rn=0.;rmax=-1.E38] ; i.le.64 ; i=i+1) [
        do j=1,64 [
                r=real(zsub(i,j))
                if(r.lt.rmax)
                        next
                if(r.eq.rmax) [
                        rn=rn+1.
                        x=x+float(j)
                        y=y+float(i)
                        next
                        ]
                rmax=r
                rn=1.
                x=float(j)
                y=float(i)
                ]
        ]
if(rn.eq.0.)
        stop 'No maximum found'
x=x/rn
y=y/rn
ix=x+.5
iy=y+.5
if(rn.gt.1.) [
```

```
        call PRINT('More than one peak found')
        ]
else[
        for(i=1 ; i.le.3 ; i=i+1) [
                do j=1,64
                        zsub2(i,j)=zsub(mod(iy+(i-2)-1+64,64)+1,j)
                do j=1,3 [
                        jj=mod(ix+(j-2)-1+64,64)+1
                        f(j,i)=real(zsub2(i,jj))
                        ]
                ]
        for([iters=1;t0x=0.;t0y=0.] ; iters.le.3 ; iters=iters+1) [
                do j=1,3
                        temp(j)=f_of_t(t0y,f(j,1),f(j,2),f(j,3))
                t0x=t0(temp(1),temp(2),temp(3))
                do j=1,3
                        temp(j)=f_of_t(t0x,f(1,j),f(2,j),f(3,j))
                t0y=t0(temp(1),temp(2),temp(3))
                ]
        rmax=temp(2)
        x=x+t0x
        y=y+t0y
    ]
if(ifix(x).gt.64/2)
        x=x-64.
if(ifix(y).gt.64/2)
        y=y-64.
return
end




subroutine datatr(cstr,x,y)
# data tramsfer
# get x-y coodinate data of a subpicture from a data file into buffer
character *(*) cstr
call bopenf(cstr,UNIT_1,'old')
call rget(x)
call rget(y)
close(unit=UNIT_1)
return
end




subroutine yclc20(x2a,y2a,x1a,y1a,x2b,y2b,x1b,y1b,cstr,m)
# calculate the angle
# (x2a,y2a) is peak location of subpic A
# (x1a,y1a) is original subpic A location
# (x2b,y2b) is peak location of subpic B
# (x1b,y1b) is original subpic B location
character *(*) cstr

y2a1=y2a+y1a                ########
y2b1=y2b+y1b                ########
x2a1=x2a+x1a                ########
x2b1=x2b+x1b                ########

theta1=atan2(y1a-y1b,x1a-x1b)
theta2=atan2(y2a1-y2b1,x2a1-x2b1)
```

```
TYPE *, THETA1, THETA2
thetar=theta1-theta2
thetad=thetar*180./PI
call bopenp(cstr,UNIT_1,2*(m-1))
call yrput(thetar)
call yrput(thetad)
close(unit=UNIT_1)
return
end


##########################################################################################
# other small subroutines
##########################################################################################


subroutine copy(zsub1,zsub2,n)
complex*8 zsub1(64,64)
complex*8 zsub2(64,64)
do i=1,n [
        do j=1,n
                zsub2(i,j)=zsub1(i,j)
        ]
return
end


logical function bopenf*1(cstr,iunit,ctype)
TP_IMPLICIT
character *(*) cstr
byte ctype(81)
COM_SAVE_FILE

lun=iunit
if(beqflg(ctype,'new'))
        open(unit=lun,name=cstr,type='NEW',carriagecontrol='LIST')
if(beqflg(ctype,'old')) [
        open(unit=lun,name=cstr,type='OLD',carriagecontrol='LIST',err=1)
        if(.false.) [
                1 bopenf=.false.
                return
                ]
        ]
if(beqflg(ctype,'unknown'))
        open(unit=lun,name=cstr,type='UNKNOWN',carriagecontrol='LIST')
bopenf=.true.
return
end


logical function bopenp*1(cstr,iunit,n)
TP_IMPLICIT
character *(*) cstr
real*4 r(MAX_ENTRIES)
COM_SAVE_FILE

lun=iunit
if(n.gt.1) [
```

```
        open(unit=lun,name=cstr,type='OLD',carriagecontrol='LIST',err=1)
        if(.false.) [
                1 bopenp=.false.
                return
                ]
        for(i=1 ; i.le.min0(n,MAX_ENTRIES) ; i=i+1)
                call rget(r(i))
        close(unit=lun)
        ]
open(unit=lun,name=cstr,type='NEW',carriagecontrol='LIST')
for(i=1 ; i.le.min0(n,MAX_ENTRIES) ; i=i+1)
        call yrput(r(i))
bopenp=.true.
return
end


subroutine ykeep(zsub,cstr)
character *(*) cstr
complex*8 zsub(64,64)
complex*8 zline(64)
open(unit=1,access='DIRECT',name=cstr,type='NEW',recordsize=128)
do i=1,64 [
        do j=1,64
                zline(j)=zsub(i,j)
        write(1%'i) zline
        ]
close(unit=1)
return
end


subroutine filbuf(cstr,zsub)
character *(*) cstr
complex*8 zsub(64,64)
complex*8 zline(64)
open(unit=1,access='DIRECT',name=cstr,type='OLD',recordsize=128)
do i=1,64 [
        read(1%'i) zline
        do j=1,64
                zsub(i,j)=zline(j)
        ]
close(unit=1)
return
end



logical function cup*1(char)
TP_IMPLICIT
%C
if('a'.le.char.and.char.le.'z')
        cup=char-32
else
        cup=char
return
end


logical function beqflg*1(cstr1,cstr2)
```

```
TP_IMPLICIT
byte cstr1(2),cstr2(2)
%C
if(cup(cstr1(1)).eq.cup(cstr2(1)).and.cup(cstr1(2)).eq.cup(cstr2(2)))
        beqflg=.true.
else
        beqflg=.false.
return
end



subroutine yrput(r)
TP_IMPLICIT
COM_SAVE_FILE

write(lun,1) r
1 FMT_STAT
return
end



real function rget*4(r)
TP_IMPLICIT
COM_SAVE_FILE

read(lun,1) r
1 FMT_STAT
rget=r
return
end




subroutine fft(z,n,direction)
integer*2 n,i,j,k,twok
real*4 s,direction
complex*8 z(n),u,w,temp
%C
if(direction.eq.'inve'.or.direction.eq.'INVE') [
        s=-PI
        temp=cmplx(1./float(n),0.)
        do i=1,n
                z(i)=temp*z(i)
        ]
else
        s=PI
for([i=1;j=1] ; i.lt.n ; i=i+1) [                   # bit reversal
        if(i.lt.j) [               # switch only once
                temp=z(j)
                z(j)=z(i)
                z(i)=temp
                ]
        # Test bsts form high to low order  If set, clear it and go on to next
        #   bit.  If clear, set it and stop   I.e  bit reversed counting
        for(k=n/2 ; k lt. j ; k=k/2)
                j=j-k
```

```
        j=j+k
        ]
# Number of stages equals log[2] of n
for([k=1;twok=2] ; twok.le.n ; [k=twok;twok=2*twok]) [
        u=(1.,0.)
        w=cexp(cmplx(0.,-s/float(k)))
        # Numper of bufferfiles equals k*n/twok = n/2
        do j=1,k [
                do i=j,n,twok [
                        temp=z(i+k)*u           # butterfly
                        z(i+k)=z(i)-temp        #     :
                        z(i)=z(i)+temp          #     :
                        ]
                u=u*w
                ]
        ]
return
end



subroutine ytrnsp(zsub)

complex*8 zsub(64,64)
complex*8 ztemp(64,64)

do i=1,64 [
        do j=1,64
                ztemp(i,j)=zsub(i,j)
        ]
do i=1,64 [
        do j=1,64
                zsub(i,j)=ztemp(j,i)
        ]
return
```

```
CALLPRINT('Please define subpicture A by the space key') !40
CALLBOX(IX,IY,64,64)       !41
X=FLOAT(IX)        !43
Y=FLOAT(IY)        !44
CALLBOPENF('DAT:A.DAT',1,'new')   !45
CALLYRPUT(X)       !46
CALLYRPUT(Y)       !47
CLOSE(UNIT=1)      !48
CALLPRINT('Please define subpicture B')   !50
CALLBOX(IX,IY,64,64)       !51
X=FLOAT(IX)        !53
Y=FLOAT(IY)        !54
CALLBOPENF('DAT:B.DAT',1,'new')   !55
CALLYRPUT(X)       !56
CALLYRPUT(Y)       !57
```

```
COMPLEX*8ONEZ(64,64)        !48
COMPLEX*8A1Z(64,64)         !49
COMPLEX*8B1Z(64,64)         !50
COMPLEX*8MHZ(64,64)         !51
COMPLEX*8MHFZ(64,64)        !52
COMPLEX*8MSKZ(64,64)        !53
CALLPRINT('I am making a mask file')        !57
CALLYMAKSR(ONEZ,64,64,1.,32,32,10,10)       !58
CALLPRINT('I am making subpicture A1Z')   !66
CALLYSBPIC(A1Z,'DAT:OCR1.PIC','DAT:A.DAT')            !67
CALLPRINT('I am making subpicture B1Z')   !69
CALLYSBPIC(B1Z,'DAT:OCR1.PIC','DAT:B.DAT')            !70
CALLPRINT('making Mexican hat filter')     !77
CALLYMKSR8(MHZ,64,64,1.,32.,32.,0.7071068)            !78
CALLYROT11(MHZ,MHFZ,-32,-32)        !86
CALLPRINT('FFT of the Mexican hat filter')            !90
CALLYXFORM(10,MHFZ)         !91
CALLYKEEP(MHFZ,'DAT:MHF.Z')         !93
CALLPRINT('making edge mask of filter')     !96
CALLYMAKSR(MSKZ,64,64,1.,32,32,28,28)       !97
CALLYKEEP(MSKZ,'DAT:MSK.Z')         !101
CALLPRINT('FFT of subpictures')   !110
CALLYXFORM(10,A1Z)          !111
CALLYXFORM(10,B1Z)          !112
CALLPRINT('filtering subpictures A1Z B1Z with Mexican Hat Filter'            !114
*) !114
CALLY2FIL(MHFZ,A1Z,20,0.0001)       !115
CALLY2FIL(MHFZ,B1Z,20,0.0001)       !118
CALLPRINT('FFT[-1] of filtered subpictures')        !121
CALLYXFORM(11,A1Z)          !122
CALLYXFORM(11,B1Z)          !123
CALLPRINT('edge masking of subpictures') !125
CALLY2FIL(MSKZ,A1Z,20,0.0001)       !126
CALLY2FIL(MSKZ,B1Z,20,0.0001)       !129
CALLPRINT('making ROI from subpictures A1Z & B1Z')          !138
CALLYNORMF(ONEZ,A1Z)        !139
CALLYNORMF(ONEZ,B1Z)        !140
CALLPRINT('I am tired')   !147
CALLYXFORM(10,A1Z)          !149
CALLYXFORM(10,B1Z)          !150
CALLYXFORM(10,ONEZ)         !151
CALLYKEEP(A1Z,'DAT:A1.Z')           !153
CALLYKEEP(B1Z,'DAT:B1.Z')           !154
CALLYKEEP(ONEZ,'DAT:ONE.Z')         !155
```

```
      COMPLEX*8A2Z(64,64)         !54
      COMPLEX*8B2Z(64,64)         !55
      COMPLEX*8MHFZ(64,64)        !56
      COMPLEX*8MSKZ(64,64)        !57
      COMPLEX*8AA2Z(64,64)        !58
      COMPLEX*8BB2Z(64,64)        !59
      COMPLEX*8ONEZ1(64,64)       !60
      COMPLEX*8ONEZ2(64,64)       !61
      COMPLEX*8A1Z1(64,64)        !62
      COMPLEX*8B1Z1(64,64)        !63
      CALLPRINT('I am making subpicture A2Z')  !65
      CALLYSBPIC(A2Z,'DAT:OCR2.PIC','DAT:A.DAT')          !66
      CALLPRINT('I am making subpicture B2Z')  !67
      CALLYSBPIC(B2Z,'DAT:OCR2.PIC','DAT:B.DAT')          !68
      CALLPRINT('FFT of subpictures')  !75
      CALLYXFORM(10,A2Z)          !76
      CALLYXFORM(10,B2Z)          !77
      CALLPRINT('filtering subpictures A2Z B2Z with Mexican Hat Filter'               !79
     *) !79
      CALLFILBUF('DAT:MHF.Z',MHFZ)        !80
      CALLY2FIL(MHFZ,A2Z,20,0.0001)       !81
      CALLY2FIL(MHFZ,B2Z,20,0.0001)       !84
      CALLPRINT('FFT[-1] of filtered subpictures')        !87
      CALLYXFORM(11,A2Z)          !88
      CALLYXFORM(11,B2Z)          !89
      CALLPRINT('edge masking of subpictures') !91
      CALLFILBUF('DAT:MSK.Z',MSKZ)        !92
      CALLY2FIL(MSKZ,A2Z,20,0.0001)       !93
      CALLY2FIL(MSKZ,B2Z,20,0.0001)       !96
      CALLPRINT('correlating') !109
      CALLCOPY(A2Z,AA2Z,64)       !110
      CALLCOPY(B2Z,BB2Z,64)       !111
      CALLY2FIL(A2Z,AA2Z,20,0.0001)       !113
      CALLY2FIL(B2Z,BB2Z,20,0.0001)       !114
      CALLPRINT('FFT for correlating') !116
      CALLYXFORM(10,A2Z)          !117
      CALLYXFORM(10,B2Z)          !118
      CALLYXFORM(10,AA2Z)         !119
      CALLYXFORM(10,BB2Z)         !120
      CALLPRINT('I am tired')  !122
      CALLFILBUF('DAT:A1.Z',A1Z1)         !124
      CALLFILBUF('DAT:B1.Z',B1Z1)         !125
      CALLPRINT('Help me, Yoshi !')       !127
      CALLY2FIL(A2Z,A1Z1,21,0.0001)       !128
      CALLY2FIL(B2Z,B1Z1,21,0.0001)       !129
      CALLFILBUF('DAT:ONE.Z',ONEZ1)       !131
      CALLCOPY(ONEZ1,ONEZ2,64) !132
      CALLY2FIL(AA2Z,ONEZ1,21,0.0001)     !134
      CALLY2FIL(BB2Z,ONEZ2,21,0.0001)     !135
      CALLPRINT('FFT[-1]')        !137
      CALLYXFORM(11,A1Z1)         !138
      CALLYXFORM(11,B1Z1)         !139
      CALLYXFORM(11,ONEZ1)        !140
      CALLYXFORM(11,ONEZ2)        !141
      CALLPRINT('square root') !143
      CALLY1FIL(ONEZ1) !144
      CALLY1FIL(ONEZ2) !145
      CALLPRINT('almost done') !147
      CALLY2FIL(ONEZ1,A1Z1,23,0.0001)     !148
      CALLY2FIL(ONEZ2,B1Z1,23,0.0001)     !149
      CALLPRINT('finding peak')           !151
```

```
CALLYPEAK4(A1Z1, X2A, Y2A, RN1)        !153
CALLDATATR('DAT:A.DAT', X1A, Y1A)      !155
CALLYPEAK4(B1Z1, X2B, Y2B, RN2)        !158
CALLDATATR('DAT:B.DAT', X1B, Y1B)      !161
CALLYCLC20(X2A, Y2A, X1A, Y1A, X2B, Y2B, X1B, Y1B, 'DAT:OCRBOX.DAT', 1)     !164
```

```
      SUBROUTINEYMAKSR(ZSUB, ISZX, ISZY, RMAG, IXO, IYO, IXC, IYC)      !57
      COMPLEX*8ZSUB(64,64), ZMAG         !61
      ZMAG=CMPLX(RMAG,0.)         !63
      DO 23000I=1,ISZX !64
      DO 23002J=1,ISZY !65
      ZSUB(I,J)=(0.,0.)          !66
      IF(.NOT.(((J.GE.IYO-IYC).AND.(J.LE.IYO+IYC)).AND.((I.GE.IXO-IXC)          !68
     *AND (I.LE.IXO+IXC))))GOTO23004      !68
      ZSUB(I,J)=ZMAG     !69
23004 CONTINUE !70
23002 CONTINUE !70
23003 CONTINUE !70
23000 CONTINUE !71
23001 CONTINUE !71
      RETURN     !72
      END        !73
      SUBROUTINEYSBPIC(ZSUB,CSTR1,CSTR2)          !82
      CHARACTER*(*)CSTR1,CSTR2 !94
      BYTECREC(512),C    !95
      BYTECPIC(256,256)           !96
      BYTECSUB(64,64)    !97
      INTEGER*2ISUB(64,64),IC     !98
      COMPLEX*8ZSUB(64,64)        !99
      COMPLEX*8ZREC(64)           !100
      EQUIVALENCE(C,IC)           !101
      DATAIC/0/             !102
      CALLBOPENF(CSTR2,2,'old')            !104
      CALLRGET(XO)          !105
      CALLRGET(YO)          !106
      CLOSE(UNIT=2)         !107
      XO=XO-32              !108
      YO=YO-32              !109
      IXO=IFIX(XO)          !110
      IYO=IFIX(YO)          !111
      IF(.NOT.(IXO.LE.0))GOTO23006         !112
      IXO=IXO+256           !113
23006 CONTINUE !114
      IF(.NOT.(IYO.LE.0))GOTO23008         !114
      IYO=IYO+256           !115
23008 CONTINUE !116
      OPEN(UNIT=1,ACCESS='DIRECT',RECORDSIZE=128,TYPE='OLD',NAME=CSTR1)          !117
      II=1       !118
      JJ=1       !119
      DO 23010I=1,128  !120
      READ(1'I)CREC         !121
      DO 23012J=1,512  !122
      CPIC(II,JJ)=CREC(J)         !123
      JJ=JJ+1    !124
      IF(.NOT.(JJ.GT.256))GOTO23014        !125
      II=II+1    !126
      JJ=1       !127
23014 CONTINUE !129
23012 CONTINUE !129
23013 CONTINUE !129
23010 CONTINUE !130
23011 CONTINUE !130
      K=1        !132
      L=1        !133
      IF(.NOT.((IYO+63).LE.256))GOTO23016          !134
      DO 23018I=IYO,IYO+63        !135
      DO 23020J=IXO,IXO+63        !136
```

```
      J2=J      !137
      IF(.NOT.(J2.GT.256))GOTO23022      !138
      J2=J2-256        !139
23022 CONTINUE !140
      CSUB(K,L)=CPIC(I,J2)      !140
      L=L+1     !141
23020 CONTINUE !142
23021 CONTINUE !142
      L=1       !143
      K=K+1     !144
23018 CONTINUE !145
23019 CONTINUE !145
23016 CONTINUE !148
      K=1       !148
      L=1       !149
      IF(.NOT.((IYO+63).GT.256))GOTO23024        !150
      DO 23026I=1,256   !151
      IF(.NOT.(((I.GE.1).AND.(I.LE.IYO+63-256)).OR.((I.GE.IYO).AND.(I.L        !152
     *E.256))))GOTO23028        !153
      L=1       !154
      DO 23030J=IXO,IXO+63      !155
      J2=J      !156
      IF(.NOT.(J2.GT.256))GOTO23032      !157
      J2=J2-256        !158
23032 CONTINUE !159
      CSUB(K,L)=CPIC(I,J2)      !159
      L=L+1     !160
23030 CONTINUE !161
23031 CONTINUE !161
      L=1       !162
      K=K+1     !163
23028 CONTINUE !165
23026 CONTINUE !165
23027 CONTINUE !165
23024 CONTINUE !167
      CLOSE(UNIT=1)     !167
      DO 23034I=1,64    !169
      DO 23036J=1,64    !170
      C=CSUB(I,J)       !171
      ISUB(I,J)=IC      !172
      ZSUB(I,J)=CMPLX(FLOAT(ISUB(I,J)),0.)       !173
23036 CONTINUE !174
23037 CONTINUE !174
23034 CONTINUE !175
23035 CONTINUE !175
      RETURN    !176
      END       !177
      SUBROUTINEYMKSRS(ZSUB,ISZX,ISZY,RMAG,XO,YO,SIGMA)        !185
      IMPLICITBYTE(B-C),INTEGER*2(I-N),REAL*4(A,E-H,O-Y),REAL*8(D),COMP        !187
     *LEX*8(Z)  !187
      COMPLEX*8ZSUB(64,64)      !189
      ZMAG=CMPLX(RMAG,0.)       !191
      XMAX=AMIN1(XO-1.,FLOAT(ISZX)-XO) !192
      YMAX=AMIN1(YO-1.,FLOAT(ISZY)-YO) !193
      V00001=1./SIGMA**2        !194
      T00002=1./(2.*SIGMA**2)   !195
      S00003=1./(SIGMA**4*SQRT(2.*3.14159265358979))     !196
      CONTINUE !197
      I=1       !197
23038 IF(.NOT.(I.LE.ISZY))GOTO23040        !197
      DO 23041J=1,ISZX !198
      RSQ=(FLOAT(J)-XO)**2+(FLOAT(I)-YO)**2      !199
      R=SQRT(RSQ)       !200
```

```
      ZSUB(I,J)=CMPLX(S00003*(2.-V00001*RSQ)*EXP(-T00002*RSQ),0.)         !202
23041 CONTINUE !203
23042 CONTINUE !203
23039 I=I+1    !204
      GOTO23038         !204
23040 CONTINUE !204
      RETURN   !205
      END      !206
      SUBROUTINEYROT11(ZSUB1,ZSUB2,IX1,IY1)     !215
      COMPLEX*8ZSUB1(64,64),ZSUB2(64,64)        !222
       IX=IX1
       IY=IY1
      IF(.NOT.(IX.LT.0))GOTO23043       !223
      IX=64+IX !224
23043 CONTINUE !225
      IF(.NOT.(IY.LT.0))GOTO23045       !225
      IY=64+IY !226
23045 CONTINUE !227
      CONTINUE !227
      I=1      !227
23047 IF(.NOT.(I . LE . 64 ))GOTO23049 !227
      DO 23050J=1,64    !228
      ZSUB2(I,(MOD(J-1+IX,64)+1))=ZSUB1(I,J)    !229
23050 CONTINUE !229
23051 CONTINUE !229
23048 I=I+1    !230
      GOTO23047         !230
23049 CONTINUE !230
      RETURN   !231
      END      !232
      SUBROUTINEYXFORM(IOP,ZSUB)        !235
      COMPLEX*8ZSUB(64,64),ZLINE(64)    !239
      CONTINUE !240
      I=1      !240
23052 IF(.NOT.(I . LE . 64 ))GOTO23054 !240
      DO 23055J=1,64    !241
      ZLINE(J)=ZSUB(I,J)        !242
23055 CONTINUE !242
23056 CONTINUE !242
      IF(.NOT.(IOP.EQ.10))GOTO23057     !243
      CALLFFT(ZLINE,64,'forw') !244
      GOTO23058         !245
23057 CONTINUE !245
      CALLFFT(ZLINE,64,'inve') !246
23058 CONTINUE !246
      DO 23059J=1,64    !247
      ZSUB(I,J)=ZLINE(J)        !248
23059 CONTINUE !248
23060 CONTINUE !248
23053 I=I+1    !249
      GOTO23052         !249
23054 CONTINUE !249
      CALLYTRNSP(ZSUB) !250
      CONTINUE !251
      I=1      !251
23061 IF(.NOT.(I . LE . 64 ))GOTO23063 !251
      DO 23064J=1,64    !252
      ZLINE(J)=ZSUB(I,J)        !253
23064 CONTINUE !253
23065 CONTINUE !253
      IF(.NOT.(IOP.EQ.10))GOTO23066     !254
      CALLFFT(ZLINE,64,'forw') !255
      GOTO23067         !256
```

```
23066   CONTINUE !256
        CALLFFT(ZLINE,64,'inve') !257
23067   CONTINUE !257
        DO 23068J=1,64    !258
        ZSUB(I,J)=ZLINE(J)        !259
23068   CONTINUE !259
23069   CONTINUE !259
23062   I=I+1     !260
        GOTO23061          !260
23063   CONTINUE !260
        RETURN    !261
        END       !262
        SUBROUTINEY2FIL(ZSUB1,ZSUB2,IOP,RMIN)       !265
        COMPLEX*8ZSUB1(64,64),ZSUB2(64,64)          !274
        IF(.NOT.(IOP.EQ.23))GOTO23070     !275
        RMAX=0.   !276
        DO 23072I=1,64    !277
        DO 23074J=1,64    !278
        RMAX=AMAX1(RMAX,ABS(REAL(ZSUB2(I,J))))     !279
23074   CONTINUE !280
23075   CONTINUE !280
23072   CONTINUE !281
23073   CONTINUE !281
        RMIN1=ABS(RMAX*RMIN)       !282
23070   CONTINUE !284
        DO 23076I=1,64    !284
        IF(.NOT.(IOP.EQ.20))GOTO23078     !285
        DO 23080J=1,64    !286
        ZSUB2(I,J)=ZSUB1(I,J)*ZSUB2(I,J)  !287
23080   CONTINUE !287
23081   CONTINUE !287
23078   CONTINUE !288
        IF(.NOT.(IOP.EQ.21))GOTO23082     !288
        DO 23084J=1,64    !289
        ZSUB2(I,J)=ZSUB1(I,J)*CONJG(ZSUB2(I,J))    !290
        23084  CONTINUE  !292
23085   CONTINUE !292
23082   CONTINUE !293
        IF(.NOT.(IOP.EQ.23))GOTO23086     !293
        DO 23088J=1,64    !294
        IF(.NOT.(ABS(REAL(ZSUB1(I,J))).GE.RMIN1))GOTO23090        !295
        ZSUB2(I,J)=CMPLX(REAL(ZSUB2(I,J))/REAL(ZSUB1(I,J)),0.)    !296
        GOTO23091          !297
23090   CONTINUE !297
        ZSUB2(I,J)=(0.,0.)         !298
23091   CONTINUE !298
23088   CONTINUE !298
23089   CONTINUE !298
23086   CONTINUE !299
23076   CONTINUE !299
23077   CONTINUE !299
        RETURN    !300
        END       !301
        SUBROUTINEYNORMF(ZSUB1,ZSUB2)      !306
        IMPLICITBYTE(B-C),INTEGER*2(I-N),REAL*4(A,E-H,O-Y),REAL*8(D),COMP          !307
      *LEX*8(Z)   !307
        COMPLEX*8ZSUB1(64,64),ZSUB2(64,64),ZSUM,ZMEAN      !308
        CONTINUE !309
        I=1       !309
        NPEL=0    !309
        ZSUM=(0.,0.)       !309
        SUMSQ=0.  !309
23092   IF(.NOT.(I.LE.64))GOTO23094 !309
```

```
        DO 23095J=1,64    !310
        IF(.NOT.(REAL(ZSUB1(I,J)).NE.0.))GOTO23097        !311
        NPEL=NPEL+1        !312
        ZSUM=ZSUM+ZSUB2(I,J)        !313
        SUMSQ=SUMSQ+CABS(ZSUB2(I,J))        !314
23097   CONTINUE !316
23095   CONTINUE !316
23096   CONTINUE !316
23093   I=I+1      !316
        GOTO23092          !316
23094   CONTINUE !316
        IF(.NOT.(NPEL.EQ.0))GOTO23099      !317
        CALLPRINT('Unable to execute')     !318
        RETURN     !319
23099   CONTINUE !321
        ZMEAN=ZSUM/NPEL    !321
        R00004=1./SQRT(SUMSQ-CABS(ZSUM)/NPEL)      !322
        CONTINUE !323
        I=1        !323
23101   IF(.NOT.(I .LE. 64 ))GOTO23103 !323
        DO 23104J=1,64     !324
        IF(.NOT.(REAL(ZSUB1(I,J)).NE.0.))GOTO23106        !325
        ZSUB2(I,J)=(ZSUB2(I,J)-ZMEAN)*R00004      !326
        GOTO23107          !327
23106   CONTINUE !327
        ZSUB2(I,J)=(0.,0.)         !328
23107   CONTINUE !328
23104   CONTINUE !328
23105   CONTINUE !328
23102   I=I+1      !329
        GOTO23101          !329
23103   CONTINUE !329
        RETURN     !330
        END        !331
        SUBROUTINEY1FIL(ZSUB)      !337
        COMPLEX*8ZSUB(64,64)       !338
        DO 23108I=1,64     !339
        DO 23110J=1,64     !340
        ZSUB(I,J)=CMPLX(SQRT(AMAX1(0.,REAL(ZSUB(I,J)))),0.)        !341
23110   CONTINUE !341
23111   CONTINUE !341
23108   CONTINUE !342
23109   CONTINUE !342
        RETURN     !343
        END        !344
        SUBROUTINEYPEAK4(ZSUB,X,Y,RN)      !348
        COMPLEX*8ZSUB(64,64),ZSUB2(64,64)          !349
        REAL*4F(3,3),TEMP(3)       !350
        TO(F1,F2,F3)=(F1-F3)/(2.*(F1-2.*F2+F3))   !351
        F00005(TO,F1,F2,F3)=(F1-2.*F2+F3)/2.*TO**2+(F3-F1)/2.*TO+F2        !352
        CONTINUE !353
        I=1        !353
        X=0.       !353
        Y=0.       !353
        RN=0.      !353
        RMAX=-1.E38        !353
23112   IF(.NOT.(I .LE. 64 ))GOTO23114 !353
        DO 23115J=1,64     !354
        R=REAL(ZSUB(I,J))          !355
        IF(.NOT.(R.LT.RMAX))GOTO23117      !356
        GOTO23115          !357
23117   CONTINUE !358
        IF(.NOT.(R.EQ.RMAX))GOTO23119      !359
```

```
      RN=RN+1.  !359
      X=X+FLOAT(J)        !360
      Y=Y+FLOAT(I)        !361
      GOTO23115           !362
23119 CONTINUE !364
      RMAX=R    !364
      RN=1.     !365
      X=FLOAT(J)          !366
      Y=FLOAT(I)          !367
23115 CONTINUE !368
23116 CONTINUE !368
23113 I=I+1     !369
      GOTO23112           !369
23114 CONTINUE !369
      IF(.NOT.(RN.EQ.0.))GOTO23121       !370
      STOP'No maximum found'    !371
23121 CONTINUE !372
      X=X/RN    !372
      Y=Y/RN    !373
      IX=X+.5   !374
      IY=Y+.5   !375
      IF(.NOT.(RN.GT.1.))GOTO23123       !376
      CALLPRINT('More than one peak found')      !377
      GOTO23124           !379
23123 CONTINUE !379
      CONTINUE !380
      I=1       !380
23125 IF(.NOT.(I .LE. 3 ))GOTO23127   !380
      DO 23128J=1,64    !381
      ZSUB2(I,J)=ZSUB(MOD(IY+(I-2)-1+64,64)+1,J)          !382
23128 CONTINUE !382
23129 CONTINUE !382
      DO 23130J=1,3     !383
      JJ=MOD(IX+(J-2)-1+64,64)+1         !384
      F(J,I)=REAL(ZSUB2(I,JJ)) !385
23130 CONTINUE !386
23131 CONTINUE !386
23126 I=I+1     !387
      GOTO23125           !387
23127 CONTINUE !387
      CONTINUE !388
      ITERS=1   !388
      TOX=0.    !388
      TOY=0.    !388
23132 IF(.NOT.(ITERS .LE. 3 ))GOTO23134        !388
      DO 23135J=1,3     !389
      TEMP(J)=F00005(TOY,F(J,1),F(J,2),F(J,3)) !390
23135 CONTINUE !390
23136 CONTINUE !390
      TOX=TO(TEMP(1),TEMP(2),TEMP(3))   !391
      DO 23137J=1,3     !392
      TEMP(J)=F00005(TOX,F(1,J),F(2,J),F(3,J)) !393
23137 CONTINUE !393
23138 CONTINUE !393
      TOY=TO(TEMP(1),TEMP(2),TEMP(3))   !394
23133 ITERS=ITERS+1     !395
      GOTO23132         !395
23134 CONTINUE !395
      RMAX=TEMP(2)      !396
      X=X+TOX   !397
      Y=Y+TOY   !398
23124 CONTINUE !399
      IF(.NOT.(IFIX(X).GT.64.2))GOTO23139        !400
```

```
      X=X-64.   !401
23139 CONTINUE !402
      IF(.NOT.(IFIX(Y).GT.64/2))GOTO23141       !402
      Y=Y-64.   !403
23141 CONTINUE !404
      RETURN    !404
      END       !405
      SUBROUTINEDATATR(CSTR,X,Y)       !411
      CHARACTER*(*)CSTR          !414
      CALLBOPENF(CSTR,1,'old')  !415
      CALLRGET(X)        !416
      CALLRGET(Y)        !417
      CLOSE(UNIT=1)      !418
      RETURN    !419
      END       !420
      SUBROUTINEYCLC20(X2A,Y2A,X1A,Y1A,X2B,Y2B,X1B,Y1B,CSTR,M)  !425
      CHARACTER*(*)CSTR          !431
      Y2A1=Y2A+Y1A       !433
      Y2B1=Y2B+Y1B       !434
      X2A1=X2A+X1A       !435
      X2B1=X2B+X1B       !436
      THETA1=ATAN2(Y1A-Y1B,X1A-X1B)      !438
      THETA2=ATAN2(Y2A1-Y2B1,X2A1-X2B1)          !439
      TYPE*,THETA1,THETA2        !440
      THETAR=THETA1-THETA2       !441
      THETAD=THETAR*180./3.14159265358979        !442
      CALLBOPENP(CSTR,1,2*(M-1))       !443
      CALLYRPUT(THETAR)          !444
      CALLYRPUT(THETAD)          !445
      CLOSE(UNIT=1)      !446
      RETURN    !447
      END       !448
      SUBROUTINECOPY(ZSUB1,ZSUB2,N)      !458
      COMPLEX*8ZSUB1(64,64)      !459
      COMPLEX*8ZSUB2(64,64)      !460
      DO 23143I=1,N      !461
      DO 23145J=1,N      !462
      ZSUB2(I,J)=ZSUB1(I,J)      !463
23145 CONTINUE !463
23146 CONTINUE !463
23143 CONTINUE !464
23144 CONTINUE !464
      RETURN    !465
      END       !466
      LOGICALFUNCTIONBOPENF*1(CSTR,IUNIT,CTYPE)        !471
      IMPLICITBYTE(B-C),INTEGER*2(I-N),REAL*4(A,E-H,O-Y),REAL*8(D),COMP      !472
     *LEX*8(Z)  !472
      CHARACTER*(*)CSTR          !473
      BYTECTYPE(81)      !474
      BYTECSTR2(81)      !475
      COMMON/SFC/LUN,CSTR2       !475
      LUN=IUNIT          !477
      IF(.NOT.(BEQFLG(CTYPE,'new')))GOTO23147  !478
      OPEN(UNIT=LUN,NAME=CSTR,TYPE='NEW',CARRIAGECONTROL='LIST')       !479
23147 CONTINUE !480
      IF(.NOT.(BEQFLG(CTYPE,'old')))GOTO23149  !480
      OPEN(UNIT=LUN,NAME=CSTR,TYPE='OLD',CARRIAGECONTROL='LIST',ERR=1) !481
      IF(.NOT.(.FALSE.))GOTO23151      !482
1     BOPENF=.FALSE.     !483
      RETURN    !484
23151 CONTINUE !486
23149 CONTINUE !487
      IF(.NOT.(BEQFLG(CTYPE,'unknown')))GOTO23153      !487
```

```
      OPEN(UNIT=LUN,NAME=CSTR,TYPE='UNKNOWN',CARRIAGECONTROL='LIST')    !488
23153 CONTINUE !489
      BOPENF=.TRUE.     !489
      RETURN   !490
      END      !491
      LOGICALFUNCTIONBOPENP*1(CSTR,IUNIT,N)     !495
      IMPLICITBYTE(B-C),INTEGER*2(I-N),REAL*4(A,E-H,O-Y),REAL*8(D),COMP          !496
     *LEX*8(Z)  !496
      CHARACTER*(*)CSTR          !497
      REAL*4R(30)        !498
      BYTECSTR2(81)      !499
      COMMON/SFC/LUN,CSTR2       !499
      LUN=IUNIT          !501
      IF(.NOT.(N.GT.1))GOTO23155         !502
      OPEN(UNIT=LUN,NAME=CSTR,TYPE='OLD',CARRIAGECONTROL='LIST',ERR=1)  !503
      IF(.NOT.(.FALSE.))GOTO23157        !504
1     BOPENP=.FALSE.     !505
      RETURN   !506
23157 CONTINUE !508
      CONTINUE !508
      I=1      !508
23159 IF(.NOT.(I .LE. MINO (N , 30 ) ))GOTO23161      !508
      CALLRGET(R(I))     !509
23160 I=I+1    !509
      GOTO23159          !509
23161 CONTINUE !509
      CLOSE(UNIT=LUN)    !510
23155 CONTINUE !512
      OPEN(UNIT=LUN,NAME=CSTR,TYPE='NEW',CARRIAGECONTROL='LIST')        !512
      CONTINUE !513
      I=1      !513
23162 IF(.NOT.(I .LE. MINO (N , 30 ) ))GOTO23164      !513
      CALLYRPUT(R(I))    !514
23163 I=I+1    !514
      GOTO23162          !514
23164 CONTINUE !514
      BOPENP=.TRUE.      !515
      RETURN   !516
      END      !517
      SUBROUTINEYKEEP(ZSUB,CSTR)         !521
      CHARACTER*(*)CSTR          !522
      COMPLEX*8ZSUB(64,64)       !523
      COMPLEX*8ZLINE(64)         !524
      OPEN(UNIT=1,ACCESS='DIRECT',NAME=CSTR,TYPE='NEW',RECORDSIZE=128)  !525
      DO 23165I=1,64     !526
      DO 23167J=1,64     !527
      ZLINE(J)=ZSUB(I,J)         !528
23167 CONTINUE !528
23168 CONTINUE !528
      WRITE(1'I)ZLINE    !529
23165 CONTINUE !530
23166 CONTINUE !530
      CLOSE(UNIT=1)      !531
      RETURN   !532
      END      !533
      SUBROUTINEFILBUF(CSTR,ZSUB)        !536
      CHARACTER*(*)CSTR          !537
      COMPLEX*8ZSUB(64,64)       !538
      COMPLEX*8ZLINE(64)         !539
      OPEN(UNIT=1,ACCESS='DIRECT',NAME=CSTR,TYPE='OLD',RECORDSIZE=128)  !540
      DO 23169I=1,64     !541
      READ(1'I)ZLINE     !542
      DO 23171J=1,64     !543
```

```
      ZSUB(I,J)=ZLINE(J)        !544
23171 CONTINUE !544
23172 CONTINUE !544
23169 CONTINUE !545
23170 CONTINUE !545
      CLOSE(UNIT=1)     !546
      RETURN    !547
      END       !548
      LOGICALFUNCTIONCUP*1(CHAR)        !554
      IMPLICITBYTE(B-C),INTEGER*2(I-N),REAL*4(A,E-H,O-Y),REAL*8(D),COMP             !555
     *LEX*8(Z)  !555
C
      IF(.NOT.('a'.LE.CHAR.AND.CHAR.LE.'z'))GOTO23173   !557
      CUP=CHAR-32       !558
      GOTO23174         !559
23173 CONTINUE !559
      CUP=CHAR !560
23174 CONTINUE !560
      RETURN    !561
      END       !562
      LOGICALFUNCTIONBEQFLG*1(CSTR1,CSTR2)      !565
      IMPLICITBYTE(B-C),INTEGER*2(I-N),REAL*4(A,E-H,O-Y),REAL*8(D),COMP             !566
     *LEX*8(Z)  !566
      BYTECSTR1(2),CSTR2(2)     !567
C
      IF(.NOT.(CUP(CSTR1(1)).EQ.CUP(CSTR2(1)).AND.CUP(CSTR1(2)).EQ.CUP(             !569
     *CSTR2(2))))GOTO23175      !569
      BEQFLG=.TRUE.     !570
      GOTO23176         !571
23175 CONTINUE !571
      BEQFLG=.FALSE.    !572
23176 CONTINUE !572
      RETURN    !573
      END       !574
      SUBROUTINEYRPUT(R)        !579
      IMPLICITBYTE(B-C),INTEGER*2(I-N),REAL*4(A,E-H,O-Y),REAL*8(D),COMP             !580
     *LEX*8(Z)  !580
      BYTECSTR2(81)     !581
      COMMON/SFC/LUN,CSTR2      !581
      WRITE(LUN,1)R     !583
1     FORMAT(G15.7)     !584
      RETURN    !585
      END       !586
      REALFUNCTIONRGET*4(R)     !590
      IMPLICITBYTE(B-C),INTEGER*2(I-N),REAL*4(A,E-H,O-Y),REAL*8(D),COMP             !591
     *LEX*8(Z)  !591
      BYTECSTR2(81)     !592
      COMMON/SFC/LUN,CSTR2      !592
      READ(LUN,1)R      !594
1     FORMAT(G15.7)     !595
      RGET=R    !596
      RETURN    !597
      END       !598
      SUBROUTINEFFT(Z,N,D00006)         !606
      INTEGER*2N,I,J,K,TWOK     !607
      REAL*4S,D00006    !608
      COMPLEX*8Z(N),U,W,TEMP    !609
C
      IF(.NOT.(D00006.EQ.'inve'.OR.D00006.EQ.'INVE'))GOTO23177 !611
      S=-3.14159265358979       !612
      TEMP=CMPLX(1./FLOAT(N),0.)        !613
      DO 23179I=1,N     !614
      Z(I)=TEMP*Z(I)    !615
```

```
23179  CONTINUE !615
23180  CONTINUE !615
       GOTO23178          !617
23177  CONTINUE !617
       S=3.14159265358979          !618
23178  CONTINUE !618
       CONTINUE !619
       I=1        !619
       J=1        !619
23181  IF(.NOT.(I .LT. N ))GOTO23183  !619
       IF(.NOT.(I.LT.J))GOTO23184        !620
       TEMP=Z(J)          !621
       Z(J)=Z(I)          !622
       Z(I)=TEMP          !623
23184  CONTINUE !627
       CONTINUE !627
       K=N/2      !627
23186  IF(.NOT.(K .LT. J ))GOTO23188  !627
       J=J-K      !628
23187  K=K/2      !628
       GOTO23186          !628
23188  CONTINUE !628
       J=J+K      !629
23182  I=I+1      !630
       GOTO23181          !630
23183  CONTINUE !630
       CONTINUE !632
       K=1        !632
       TWOK=2     !632
23189  IF(.NOT.(TWOK .LE. N ))GOTO23191          !632
       U=(1.,0.)          !633
       W=CEXP(CMPLX(0.,-S/FLOAT(K)))     !634
       DO 23192J=1,K      !636
       DO 23194I=J,N,TWOK          !637
       TEMP=Z(I+K)*U      !638
       Z(I+K)=Z(I)-TEMP   !639
       Z(I)=Z(I)+TEMP     !640
23194  CONTINUE !641
23195  CONTINUE !641
       U=U*W      !642
23192  CONTINUE !643
23193  CONTINUE !643
23190  K=TWOK     !644
       TWOK=2*TWOK        !644
       GOTO23189          !644
23191  CONTINUE !644
       RETURN     !645
       END        !646
       SUBROUTINEYTRNSP(ZSUB)      !651
       COMPLEX*8ZSUB(64,64)        !653
       COMPLEX*8ZTEMP(64,64)       !654
       DO 23196I=1,64     !656
       DO 23198J=1,64     !657
       ZTEMP(I,J)=ZSUB(I,J)        !658
23198  CONTINUE !658
23199  CONTINUE !658
23196  CONTINUE !659
23197  CONTINUE !659
       DO 23200I=1,64     !660
       DO 23202J=1,64     !661
       ZSUB(I,J)=ZTEMP(J,I)        !662
23202  CONTINUE !662
23203  CONTINUE !662
```

```
23200   CONTINUE !663
23201   CONTINUE !663
        RETURN   !664
```

Appendix 3

```
##############################################################################
#
#       PROGRAM MITFS.RAT
#
#       This program transform picture data files into transferable format
#       to Tafts University Image Analisys Laboratory VAX System
#
#       Input   OCR1.PIC and OCR2.PIC
#       Output  OCR1.TFS and OCR2.PIC
#
#       R RATFOR
#       MITFS.RAT
#       ^C
#       FORT/NOLINENUMBER MITFS
#       LINK MITFS
#       RUN MITFS
##############################################################################

call tsfr('DAT OCR1.PIC','DAT:OCR1 TFS')
call tsfr('DAT OCR2.PIC','DAT.OCR2.TFS')
end

subroutine tsfr(cstr1,cstr2)
byte cstr1(81),cstr2(81),creci(512),creco(512),cr,lf
open(unit=1,name=cstr1,access='DIRECT',recordsize=128,type='OLD')
open(unit=2,name=cstr2,access='DIRECT',recordsize=128,type='NEW')
cr=13   # carrige return
lf=10   # line feed
iptro=1
jj=1
do i=1,128 [
        read(1%'i) creci
        do j=1,512 [
                creco(jj)=creci(j).and.240                  # 240 is 11110000
                if(creci(j).ge.0)
                        creco(jj)=creco(jj)/16              # 16 is 00010000
                                                            # shift 4 digits
                else [
                        creco(jj)=creco(jj)/16
                        creco(jj)=creco(jj).or.8            # 8 is 00001000
                        creco(jj)=creco(jj).and 15          # 15 is 00001111
                        ]
                creco(jj)=creco(jj)+64          # 64 is 01000000
                jj=jj+1
                if((jj.eq.63).or (jj.eq 127).or.(jj.eq.191).or. \
                        (jj eq.255).or.(jj.eq.319).or.(jj eq.383).or. \
                        (jj.eq.447)) [
                        creco(jj)=cr
                        jj=jj+1
                        creco(jj)=lf
                        ]
                else [
                        creco(jj)=creci(j).and.15           # 15 is 00001111
                        creco(jj)=creco(jj)+64              # 64 is 01000000
                        ]
                jj=jj+1
                if((jj.eq.63).or.(jj.eq.127).or.(jj.eq 191).or. \
                        (jj.eq 255) or.(jj.eq.319).or (jj.eq.383).or. \
                        (jj.eq 447).or.(jj.eq.511)) [
                        creco(jj)=cr
                        jj=jj+1
                        creco(jj)=lf
```

```
                        jj=jj+1
                        ]
                if(jj.gt.512) [
                        write(2%'iptro) creco
                        type *,iptro
                        iptro=iptro+1
                        jj=1
                        ]
                ]
        ]
do jj=133,512 [
        if((jj.eq.63).or.(jj.eq.127).or.(jj.eq.191).or.(jj.eq.255).or. \
        (jj.eq.319).or.(jj.eq.383).or.(jj.eq.447).or.(jj.eq.511))
                creco(jj)=cr
        else if((jj.eq.64).or.(jj.eq.128).or.(jj.eq.192).or.(jj.eq.256).or. \
        (jj.eq.320).or.(jj.eq.384).or.(jj.eq.448).or.(jj.eq.512))
                creco(jj)=lf
        else
                creco(jj)=64     # 64 is 01000000
        ]
write(2%'iptro) creco
type *,iptro
close(unit=1)
close(unit=2)
return
```

```
      CALLTSFR('DAT:OCR1.PIC','DAT:OCR1.TFS')   !19
      CALLTSFR('DAT:OCR2.PIC','DAT:OCR2.TFS')   !20
      END      !21
      SUBROUTINETSFR(CSTR1,CSTR2)        !23
      BYTECSTR1(81),CSTR2(81),CRECI(512),CRECO(512),CR,LF       !24
      OPEN(UNIT=1,NAME=CSTR1,ACCESS='DIRECT',RECORDSIZE=128,TYPE='OLD')        !25
      OPEN(UNIT=2,NAME=CSTR2,ACCESS='DIRECT',RECORDSIZE=128,TYPE='NEW')        !26
      CR=13     !27
      LF=10     !28
      IPTRO=1   !29
      JJ=1      !30
      DO 23000I=1,128  !31
      READ(1'I)CRECI   !32
      DO 23002J=1,512  !33
      CRECO(JJ)=CRECI(J) AND 240       !34
      IF(.NOT.(CRECI(J) GE.0))GOTO23004         !35
      CRECO(JJ)=CRECO(JJ)/16   !36
      GOTO23005        !38
23004 CONTINUE !38
      CRECO(JJ)=CRECO(JJ)/16   !39
      CRECO(JJ)=CRECO(JJ).OR 8 !40
      CRECO(JJ)=CRECO(JJ).AND.15       !41
23005 CONTINUE !42
      CRECO(JJ)=CRECO(JJ)+64   !43
      JJ=JJ+1  !44
      IF(.NOT.((JJ EQ.63) OR.(JJ.EQ 127) OR (JJ.EQ.191) OR (JJ EQ 255)         !46
     *OR.(JJ EQ.319) OR (JJ EQ 383) OR (JJ EQ 447)))GOTO23006   !47
      CRECO(JJ)=CR     !48
      JJ=JJ+1  !49
      CRECO(JJ)=LF     !50
      GOTO23007        !52
23006 CONTINUE !52
      CRECO(JJ)=CRECI(J) AND 15        !53
      CRECO(JJ)=CRECO(JJ)+64   !54
23007 CONTINUE !55
      JJ=JJ+1  !56
      IF( NOT ((JJ EQ 63) OR (JJ EQ 127) OR.(JJ EQ 191) OR (JJ.EQ 255)         !58
     *OR (JJ.EQ 319) OR.(JJ EQ 383).OR.(JJ EQ.447).OR (JJ EQ 511)))GOTO2       !59
     *3008     !59
      CRECO(JJ)=CR     !60
      JJ=JJ+1  !61
      CRECO(JJ)=LF     !62
      JJ=JJ+1  !63
23008 CONTINUE !65
      IF(.NOT.(JJ.GT.512))GOTO23010    !65
      WRITE(2'IPTRO)CRECO      !66
      TYPE*,IPTRO      !67
      IPTRO=IPTRO+1    !68
      JJ=1     !69
23010 CONTINUE !71
23002 CONTINUE !71
23003 CONTINUE !71
23000 CONTINUE !72
23001 CONTINUE !72
      DO 23012JJ=133,512       !73
      IF(.NOT.((JJ.EQ.63).OR.(JJ.EQ.127).OR.(JJ.EQ.191).OR.(JJ.EQ 255).        !74
     *OR.(JJ.EQ.319).OR.(JJ.EQ.383).OR.(JJ.EQ.447).OR.(JJ.EQ.511)))GOTO2       !75
     *3014     !75
      CRECO(JJ)=CR     !76
      GOTO23015        !77
23014 CONTINUE !77
```

```
      IF(.NOT.((JJ.EQ.64).OR.(JJ.EQ.128).OR.(JJ.EQ.192).OR.(JJ.EQ.256)               '77
     *OR.(JJ.EQ.320).OR.(JJ.EQ.384).OR.(JJ.EQ.448).OR.(JJ.EQ.512)))GOTO2             '78
     *3016         '78
      CRECO(JJ)=LF          !79
      GOTO23017             !80
23016 CONTINUE !80
      CRECO(JJ)=64          !81
23017 CONTINUE '81
23015 CONTINUE !81
23012 CONTINUE !82
23013 CONTINUE !82
      WRITE(2'IPTRO)CRECO          '83
      TYPE*,IPTRO           !84
      CLOSE(UNIT=1)         !85
      CLOSE(UNIT=2)         !86
      RETURN      '87
```

Appendix 4

```
     Welcome to the Image Analysis Lab VAX-11/780 VMS V4.0

    Last interactive login on Monday,  1-APR-1985 11:20
rror opening DRA1:[SPACEYE]LOGIN.COM; as input
ile not found
 type tfspic.for
c      PROGRAM TFSPIC.FOR
c      This program transforms the data file which transfered
c      from MVL RT-11 System to Tufts University Image Analisys
c      Laboratory VAX System into the original picture format.
c      Mar-21-85
c      Yoshihiro Nagashima    Massachusetts Institute of Technology
c                             Man Vehicle Laboratory

       call tfspc('DAT:OCR1.TFS','DAT:TEST.PIC')
C      call tfspc('DAT:OCR2.TFS','DAT:OCR2.PIC')
       end

       subroutine tfspc(cstr1,cstr2)

       character *(*) cstr1,cstr2
       byte cmtxi(256,512),clineo(512),creci(62)      ! 62 comes from the
```

```
      character *(*) cstr1,cstr2
      byte cmtxi(256,512),clineo(512),creci(62)      ! 62 comes from the
                                                     ! max length of records
      byte cmtxh(256,256),cmtxl(256,256),cmtxo(256,256),ctmph(256,256)
      open(unit=1,name=cstr1,access='SEQUENTIAL',recordtype='VARIABLE',
     1 type='OLD',form='FORMATTED')
      open(unit=2,name=cstr2,access='DIRECT',recordsize=128,type='NEW')

cc    get rid of garbage data and define a 256*512 picture matrix
      ii=1
      jj=1
      do 10 i=1,512*300
            read(1,1000) creci
            do 20 j=1,512
                  if(creci(j).ge.64) go to 30
                        go to 40
30                      continue
                        cmtxi(ii,jj)=creci(j)
                        jj=jj+1
                        if(jj.gt.512) go to 50
                              go to 60
50                            continue
                              ii=ii+1
                              if(ii.gt.256) go to 10
```

```
                                                    go to 60
 50                                                 continue
                                                    ii=ii+1
                                                    if(ii.gt.256) go to 100
                                                    jj=1
 60                                                 continue
 40                                         continue
 20                                 continue
 10                         continue
 100        continue

c           every data minus 01000000       ( 01000000=64 )
            do 110 i=1,256
                    do 120 j=1,512
                            cmtxi(i,j)=cmtxi(i,j)-64
 120                        continue
 110                continue

c           define a matrix ( 256,256 ) of the higher 4 digits
            do 130 i=1,256
                    do 140 j=1,256
                            cmtxh(i,j)=cmtxi(i,2*j-1)
                            ctmph(i,j)=cmtxh(i,j)    ! for later use
 140        _
```

```
                do 140 j=1,256
                        cmtxh(i,j)=cmtxi(i,2*j-1)
                        ctmph(i,j)=cmtxh(i,j)    ! for later use
140                     continue
130             continue

c       define a matrix ( 256,256 ) of the lower 4 digits
        do 150 i=1,256
                do 160 j=1,256
                        cmtxl(i,j)=cmtxi(i,2*j)
160                     continue
150             continue

c       shift the higher 4 bits
        do 170 i=1,256
                do 180 j=1,256
                        if(cmtxh(i,j).ge.8) go to 190   ! 8 is 00001000
                                go to 200
190                             continue
                                cmtxh(i,j)=cmtxh(i,j).and.7     ! 7 is 00000111
                                cmtxh(i,j)=cmtxh(i,j)*16        ! 16 is 00010000
                                                                ! shift 4 digits
c                               cmtxh(i,j)=-1*cmtxh(i,j)       ! add 1 in 8th bit
c                               Above statement is not ope_
```

```
:c                              cmtxh(i,j)=-1*cmtxh(i,j)       ! add 1 in 8th bit
:c                              Above statement is not operated here. May be
:c                              later.
                                go to 210
 200                            continue
:c                      else
                                cmtxh(i,j)=cmtxh(i,j)*16       ! shift 4 digits
 210                            continue
 180                    continue
 170            continue

:c      combine two matrices
        do 220 i=1,256
                do 230 j=1,256
:c                      cmtxo(i,j)=cmtxh(i,j).or.cmtxl(i,j)
:c                      Above statement doesn't work for negative (1***0000)
:c                      numbers of cmtxh(i,j) becase ".or." works as addition.
:c                      So, ....
                        if(ctmph(i,j).lt.8) go to 300   ! 8 is 00001000
                                                        ! This is for positive
                                                        ! number.
                                go to 310
 300                            continue
```

```
                                                 ! This is for positive
                                                 ! number.
                              go to 310
300                           continue
                              cmtxo(i,j)=cmtxh(i,j)+cmtxl(i,j)
                              go to 320
310                           continue
:c                    else    (ctmph(i,j)=00001***)   ! This is for negative
                                                 ! number.
                              cmtxo(i,j)=cmtxh(i,j)+cmtxl(i,j)
                              cmtxo(i,j)=-1*cmtxo(i,j)! add 1 in 8th bit
                                                 ! This doesn't work by
                                                 ! itself. So next state-
                                                 ! ment is necessary.
                              cmtxo(i,j)=-128-cmtxo(i,j)
320                           continue
230                   continue
220           continue

:c     write the matrix data every 512 bytes ( 128 recordsize )
       jj=1
       iptro=1
       do 240 i=1,256
              do 250 j=1,2_
```

```
c      write the matrix data every 512 bytes ( 128 recordsize )
       jj=1
       iptro=1
       do 240 i=1,256
              do 250 j=1,256
                     clineo(jj)=cmtxo(i,j)
                     jj=jj+1
                     if(jj.gt.512) go to 260
                            go to 270
260                         continue
                            write(2'iptro) clineo
                            iptro=iptro+1
                            jj=1
270                         continue
250                  continue
240           continue


1000   format (62A)
       close(unit=1)
       close(unit=2)
       return
       end
```

Appendix 5

Appendix 5. ASCII Character Chart

DECIMAL	OCTAL	BINARY	DECIMAL	OCTAL	BINARY
0	000	0000 0000	16	020	0001 0000
1	001	0000 0001	17	021	0001 0001
2	002	0000 0010	18	022	0001 0010
3	003	0000 0011	19	023	0001 0011
4	004	0000 0100	20	024	0001 0100
5	005	0000 0101	21	025	0001 0101
6	006	0000 0110	22	026	0001 0110
7	007	0000 0111	23	027	0001 0111
8	010	0000 1000	24	030	0001 1000
9	011	0000 1001	25	031	0001 1001
10	012	0000 1010	26	032	0001 1010
11	013	0000 1011	27	033	0001 1011
12	014	0000 1100	28	034	0001 1100
13	015	0000 1101	29	035	0001 1101
14	016	0000 1110	30	036	0001 1110
15	017	0000 1111			

DECIMAL	OCTAL	BINARY	DECIMAL	OCTAL	BINARY
31	037	0001 1111	51	063	0011 0011
32	040	0010 0000	52	064	0011 0100
33	041	0010 0001	53	065	0011 0101
34	042	0010 0010	54	066	0011 0110
35	043	0010 0011	55	067	0011 0111
36	044	0010 0100	56	070	0011 1000
37	045	0010 0101	57	071	0011 1001
38	046	0010 0110	58	072	0011 1010
39	047	0010 0111	59	073	0011 1011
40	050	0010 1000	60	074	0011 1100
41	051	0010 1001	61	075	0011 1101
42	052	0010 1010	62	076	0011 1110
43	053	0010 1011	63	077	0011 1111
44	054	0010 1100	64	100	0100 0000
45	055	0010 1101	65	101	0100 0001
46	056	0010 1110	66	102	0100 0010
47	057	0010 1111	67	103	0100 0011
48	060	0011 0000	68	104	0100 0100
49	061	0011 0001	69	105	0100 0101
50	062	0011 0010	70	106	0100 0110

DECIMAL	OCTAL	BINARY	DECIMAL	OCTAL	BINARY
71	107	0100 0111	91	133	0101 1011
72	110	0100 1000	92	134	0101 1100
73	111	0100 1001	93	135	0101 1101
74	112	0100 1010	94	136	0101 1110
75	113	0100 1011	95	137	0101 1111
76	114	0100 1100	96	140	0110 0000
77	115	0100 1101	97	141	0110 0001
78	116	0100 1110	98	142	0110 0010
79	117	0100 1111	99	143	0110 0011
80	120	0101 0000	100	144	0110 0100
81	121	0101 0001	101	145	0110 0101
82	122	0101 0010	102	146	0110 0110
83	123	0101 0011	103	147	0110 0111
84	124	0101 0100	104	150	0110 1000
85	125	0101 0101	105	151	0110 1001
86	126	0101 0110	106	152	0110 1010
87	127	0101 0111	107	153	0110 1011
88	130	0101 1000	108	154	0110 1100
89	131	0101 1001	109	155	0110 1101
90	132	0101 1010	110	156	0110 1110

DECIMAL	OCTAL	BINARY	DECIMAL	OCTAL	BINARY
111	157	0110 1111			
112	160	0111 0000			
113	161	0111 0001			
114	162	0111 0010			
115	163	0111 0011			
116	164	0111 0100			
117	165	0111 0101			
118	166	0111 0110			
119	167	0111 0111			
120	170	0111 1000			
121	171	0111 1001			
122	172	0111 1010			
123	173	0111 1011			
124	174	0111 1100			
125	175	0111 1101			
126	176	0111 1110			
127	177	0111 1111			

DECIMAL	OCTAL	BINARY	DECIMAL	OCTAL	BINARY
-1	377	1111 1111	-21	353	1110 1011
-2	376	1111 1110	-22	352	1110 1010
-3	375	1111 1101	-23	351	1110 1001
-4	374	1111 1100	-24	350	1110 1000
-5	373	1111 1011	-25	347	1110 0111
-6	372	1111 1010	-26	346	1110 0110
-7	371	1111 1001	-27	345	1110 0101
-8	370	1111 1000	-28	344	1110 0100
-9	367	1111 0111	-29	343	1110 0011
-10	366	1111 0110	-30	342	1110 0010
-11	365	1111 0101	-31	341	1110 0001
-12	364	1111 0100	-32	340	1110 0000
-13	363	1111 0011	-33	337	1101 1111
-14	362	1111 0010	-34	336	1101 1110
-15	361	1111 0001	-35	335	1101 1101
-16	360	1111 0000	-36	334	1101 1100
-17	357	1110 1111	-37	333	1101 1011
-18	356	1110 1110	-38	332	1101 1010
-19	355	1110 1101	-39	331	1101 1001
-20	354	1110 1100	-40	330	1101 1000

DECIMAL	OCTAL	BINARY	DECIMAL	OCTAL	BINARY
-41	327	1101 0111	-61	303	1100 0011
-42	326	1101 0110	-62	302	1100 0010
-43	325	1101 0101	-63	301	1100 0001
-44	324	1101 0100	-64	300	1100 0000
-45	323	1101 0011	-65	277	1011 1111
-46	322	1101 0010	-66	276	1011 1110
-47	321	1101 0001	-67	275	1011 1101
-48	320	1101 0000	-68	274	1011 1100
-49	317	1100 1111	-69	273	1011 1011
-50	316	1100 1110	-70	272	1011 1010
-51	315	1100 1101	-71	271	1011 1001
-52	314	1100 1100	-72	270	1011 1000
-53	313	1100 1011	-73	267	1011 0111
-54	312	1100 1010	-74	266	1011 0110
-55	311	1100 1001	-75	265	1011 0101
-56	310	1100 1000	-76	264	1011 0100
-57	307	1100 0111	-77	263	1011 0011
-58	306	1100 0110	-78	262	1011 0010
-59	305	1100 0101	-79	261	1011 0001
-60	304	1100 0100	-80	260	1011 0000

DECIMAL	OCTAL	BINARY	DECIMAL	OCTAL	BINARY
-81	257	1010 1111	-101	233	1001 1011
-82	256	1010 1110	-102	232	1001 1010
-83	255	1010 1101	-103	231	1001 1001
-84	254	1010 1100	-104	230	1001 1000
-85	253	1010 1011	-105	227	1001 0111
-86	252	1010 1010	-106	226	1001 0110
-87	251	1010 1001	-107	225	1001 0101
-88	250	1010 1000	-108	224	1001 0100
-89	247	1010 0111	-109	223	1001 0011
-90	246	1010 0110	-110	222	1001 0010
-91	245	1010 0101	-111	221	1001 0001
-92	244	1010 0100	-112	220	1001 0000
-93	243	1010 0011	-113	217	1000 1111
-94	242	1010 0010	-114	216	1000 1110
-95	241	1010 0001	-115	215	1000 1101
-96	240	1010 0000	-116	214	1000 1100
-97	237	1001 1111	-117	213	1000 1011
-98	236	1001 1110	-118	212	1000 1010
-99	235	1001 1101	-119	211	1000 1001
-100	234	1001 1100	-120	210	1000 1000

DECIMAL	OCTAL	BINARY	DECIMAL	OCTAL	BINARY
-121	207	1000 0111			
-122	206	1000 0110			
-123	205	1000 0101			
-124	204	1000 0100			
-125	203	1000 0011			
-126	202	1000 0010			
-127	201	1000 0001			

BIBLIOGRAPHY

[1] John Anthony Parker, "Measurement of Torsion from Multi-Temporal Images of the Eye Using Digital Signal Processing Techniques" Doctor's Thesis, Massachusetts Institute of Technology, 1983.

[2] J.Anthony Parker, Robert V.Kenyon, Lawrence R.Young, "Measurement of Torsion from Multitemporal Images of the Eye Using Digital Signal Processing Techniques" IEEE Transactions on Biomedical Engineering, January, 1985.

[3] Mehdi Hatamian, "An Image Analysis Algorith for Real Time Measurement of X,Y and Torsional Eye Movements - Theory and Implementaion" Doctor's Thesis, The University of Michigan, 1982.

[4] Mehdi Hatamian, David J.Anderson, "Design Considerations for a Real - Time Ocular Counterroll Instrument" IEEE Transactions on Biomedical Engineering, May, 1983.

[5] James E.Anderson, "Grant's Atlas of Anatomy" Williams and Wilkins.

[6] Robert G.Petersdorf, "Principles of Internal Medicine" McGraw-Hill.

BIOGRAPHY

I was born September 30, 1954 in Japan. I received a bachelor's degree in mechanical engineering in 1977 and a master's degree in engineering in 1979 from Tokyo Institute of Technology.

I worked for Mitsubishi Heavy Industries as an aircraft flight control equipment design engineer. A scholarship from this company allowed me to do full time class work during the years 1983-1985 at MIT. In 1984, I joined the Man-Vehicle Laboratory.

I plan to continue working for Mitsubishi after receiving a master's degree in Aeronautics and Astronautics.