Technical Report Documentation Page

1. Report No. TX-96/1965-1F	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle USER'S MANUAL FOR A PAVEMENT VIDEO IMAGE PROCESSING SYSTEM		5. Report Date November 1995 Revised: March 1996 6. Performing Organization Code
7. Author(s) Chun-Lok Lau		8. Performing Organization Report No. Research Report 1965-1F
9. Performing Organization Name and Address Texas Transportation Institute The Texas A&M University System College Station, Texas 77843-3135		10. Work Unit No. (TRAIS) 11. Contract or Grant No. Study No. 7-1965
12. Sponsoring Agency Name and Address Texas Department of Transportation Research and Technology Transfer Office P. O. Box 5080 Austin, Texas 78763-5080		13. Type of Report and Period Covered Final Report: September 1994 - August 1995 14. Sponsoring Agency Code

15. Supplementary Notes
Research performed in cooperation with the Texas Department of Transportation.
Research Study Title: Implementation of Automatic Surface Distress Data Collection Procedures for PMIS

16. Abstract

This report provides a user's manual for the automated video distress identification system developed by TTI for the Texas Department of Transportation.

17. Key Words Pavement Distress, Video, Automated Processing, Image Processing	18. Distribution Statement No Restrictions. This document is available to the public through NTIS: National Technical Information Service 5285 Port Royal Road Springfield, Virginia 22161

19. Security Classif.(of this report) Unclassified	20. Security Classif.(of this page) Unclassified	21. No. of Pages 66	22. Price

Form DOT F 1700.7 (8-72) Reproduction of completed page authorized

USER'S MANUAL FOR A PAVEMENT VIDEO IMAGE PROCESSING SYSTEM

by

Chun-Lok Lau
Engineering Research Associate
Texas Transportation Institute

Research Report 1965-1F
Research Study Number 7-1965
Research Study Title: Implementation of Automatic Surface Distress Data Collection Procedures for PMIS

Sponsored by the
Texas Department of Transportation

November 1995
Revised: March 1996

TEXAS TRANSPORTATION INSTITUTE
The Texas A&M University System
College Station, Texas 77843-3135

IMPLEMENTATION STATEMENT

The system described in this report is currently being evaluated and pilot tested by the TxDOT Pavement Design Section. Once this review is complete and minor corrections are made, full implementation should proceed.

DISCLAIMER

ACKNOWLEDGMENT

Mr. David Fink and Carl Bertrand of TxDOT are acknowledged for their support of this study.

TABLE OF CONTENTS

LIST OF FIGURES

SUMMARY

This report provides a user's manual for the automated video distress identification system developed by TTI for the Texas Department of Transportation.

CHAPTER I
INTRODUCTION

This Pavement Image Processing System is capable of automatically characterizing and counting pavement distresses on surfaces recorded on a video tape. The system has been tested on several Texas concrete highway pavements and has a 75% accuracy with respect to manual inspection. For hot mix pavements, a separate algorithm is used which is not included for discussion in this report.

Operators use a special TxDOT vehicle (called the Multi Functional Vehicle (MFV)) equipped with a computerized distance measuring instrument (DMI) and a video capturing equipment to video-tape the pavement surface. The video recording also records time codes (TC). The time code information is correlated to the distance information. In order to facilitate the analysis of the pavement images, a data file, with a cross reference table of frame numbers and distance mileages, is generated with respect to the video. This data file will be taken as the input for the "crc95" program, for analysis of the corresponding tape. Depending on the user's choice, every frame in the video can be processed to detect cracks, or a certain number of frames can be skipped before the next analysis takes place. The detected distresses are categorized into either longitudinal cracking, transverse cracking, spalled cracks, or intact pavement. The time required to process one frame is about two seconds. The result of the analysis is reported in the form of a file listing a summary of each of the pavement sections, or a distress characteristics summary plus summary of distresses at each video location. The summary of each section includes the total number of cracks of each category found and the average crack spacing within that section.

CHAPTER II
HARDWARE CONFIGURATION

The system consists of a host computer, an image processor unit, and a video cassette recorder/player (VCP) with Time Code capability. Other auxiliary equipment is indicated in the system hardware configuration diagram shown in Figure 1. The corresponding picture of the system setup is shown in Figure 2.

The host computer controls and communicates with the VCP through an RS232 to RS422 protocol converter (RS232 on the host computer side and RS422 on the VCP side). The communication between the image processor and the host computer is through their VMS Bus ports.

The VCP sends a video image to the image processor for digitization. The original image is displayed on one monitor while the digitized image is displayed on another monitor. This provides a method by which the pavement can be manually inspected for cracks.

Figure 3 shows the setup schematic of the interconnections among the equipment.

DESCRIPTION OF THE EQUIPMENT

Host Computer

The host computer is a SUN SPARCstation IPX, the fastest SPARCstation available at the time the system was being integrated. Unix is the operation system of this computer. The distress analysis process is operated in this computer. The computer controls how fast the VCP plays, when to capture and digitize the image as well as when to perform various image analysis calibrations.

Image Processor (Imaging Technology Inc. Series 151)

The image processing unit used is a Series 151 from IMAGING Technology Inc. The image processor digitizes images sent from the VCP, sends the resulting binary image file to

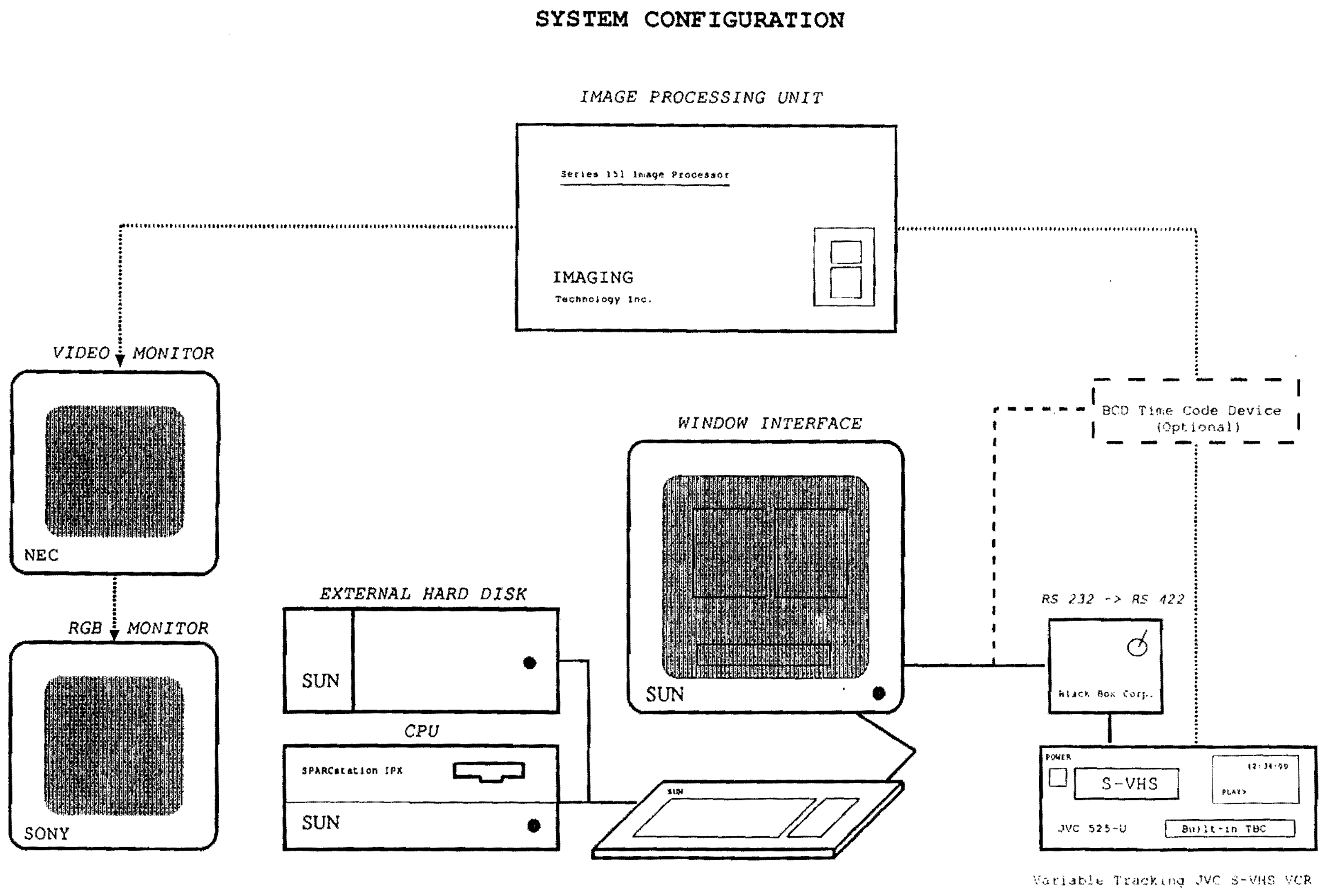


Figure 1. System Configuration.

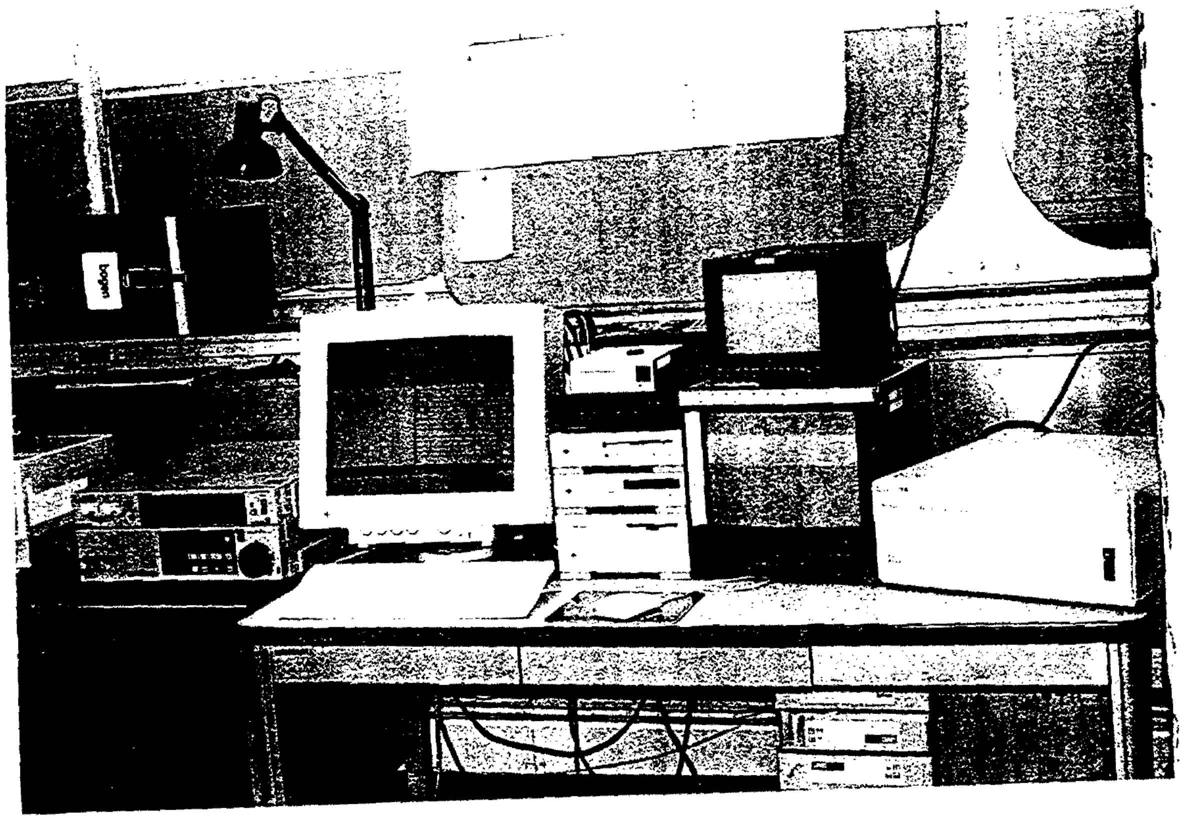


Figure 2. Picture of System Setup.

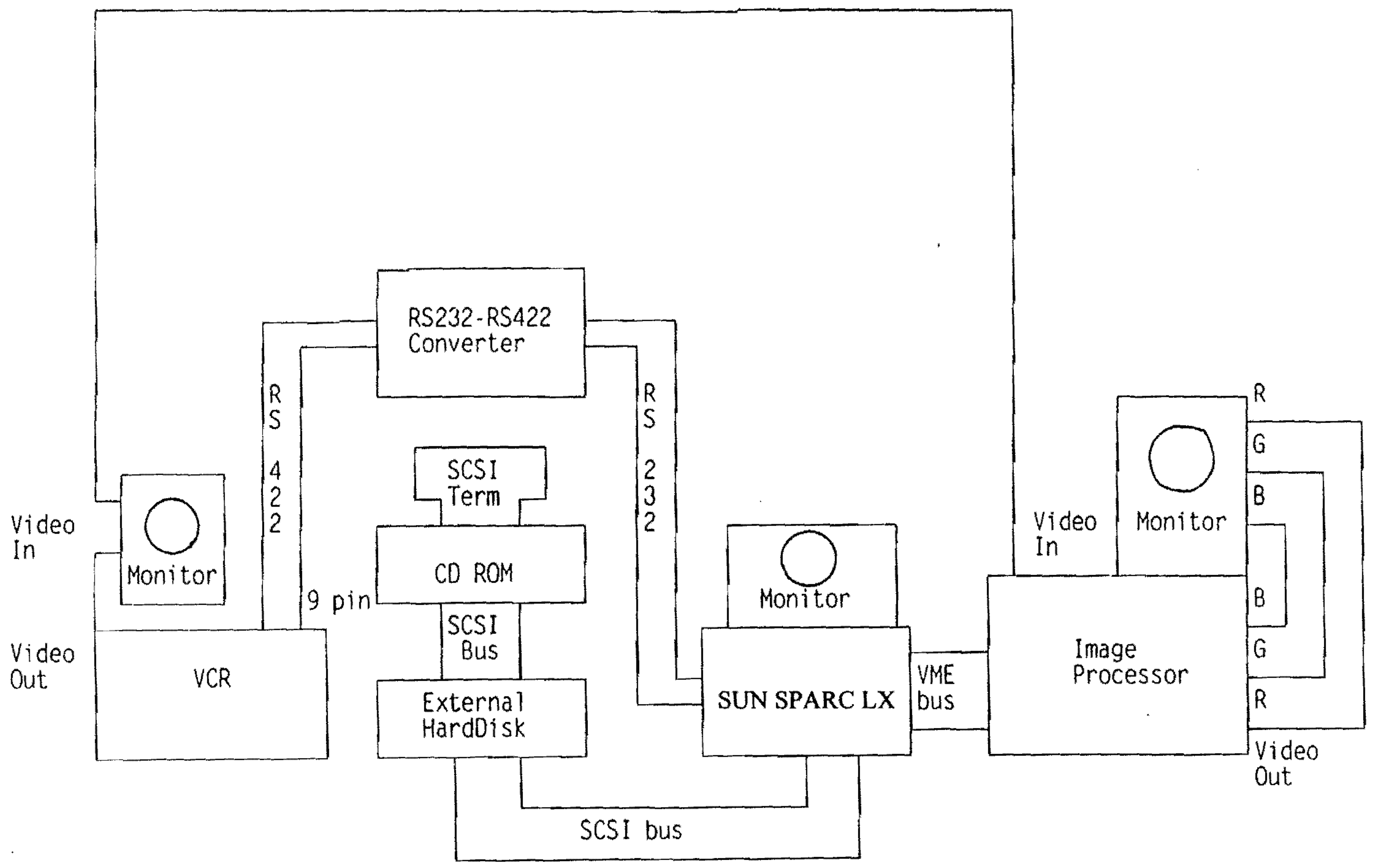


Figure 3. System Components Interconnection.

the host computer for analysis, and displays the digitized image on the monitor. The images are recorded in black & white. The image is sampled to a 512 x 512 pixels (each pixel is a point of standard size on a computer monitor) digital picture. Each pixel can have one of 256 possible grey-levels, corresponding to the level of brightness or darkness of the individual pixel.

Interface to Image Processor (Bit3 VME-S Bs Adaptor, Bit3 S Bus Repeater)

Video Cassette Recorder/Player (JVC Model BR-S525U)

The video cassette recorder/player (VCP) is computer controllable and has variable tracking, capable of reading Time Code information recorded on one of the video tape's audio tracks. The Time Code information is generated by a similar VCP during recording of the tape. When the tape is played back in the system, the "crc 95" algorithm uses the time code to position the tape to the desired video frame. Since the Time Code uses a cross-reference table to the distance/mileage information, each video frame is tied with the distance information. The settings of the VCP during operation should be set as follows:

Set the COUNTER switch on
Select TC (Time Code) on the COUNTER switch
COUNTER switch, select TC(Time Code);
REMOTE switch, select 9PIN (i.e. RS232); and
Set VIDEO LEVEL, CHROMA LEVEL, CHROMA PHASE and BLACK LEVEL at mid-level.

Interface to VCP (National Instrument SCSI-IEEE 488 Adaptor)

RS232/RS422 Converter

Since the host computer uses the RS232 communication protocol while the VCP uses RS422, a converter performs the conversion between the two protocols. This converter is made by Black Box Corporation.

Analog Video Monitor (NEC Model PM-1271A)

This analog composite video monitor is used to display the original image from the VCP.

Analog/Digital Video Monitor(Sony Model PVM-13420)

This video monitor, capable of displaying both analog and video signals, is used to display the digitized image sent by the image processor. Use digital video mode to display images from the image processing unit.

CHAPTER III
SOFTWARE CONFIGURATION

The "crc95" software is a command-driven program which runs on a SUN unix platform. It accepts a Mission Manager File as input. A Mission Manager File is an ASCII data file which consists of cross-related columns of numbers about date, time, mileage, video time code, assigned pavement section number, pavement type code ... etc. This file is generated by a program known as the "Mission Manager" during video-taping of the pavement. As output, "crc95" generates a file summarizing the number of cracks detected for each of the categories of cracks, as well as the average crack spacing for the section.

A complete listing of the program crc95 can be found in Appendix A.
The algorithm and crack-classification rules are discussed in the final report of Study 1189, a TTI research report. They are included in Appendix B.

When the video tape of the pavement and the corresponding Mission Manager File (which is stored on an IBM PC format floppy diskette) are ready, put the tape into the VCP and the diskette into the floppy drive of the SUN SPARC workstation.

Now login the appropriate account where the crc95 and Mission Manager Files are.

```
username   pchan
password   responsible
```

Change directory to where the Mission Manager files will be found/stored.

```
cd crack/section
```

To see what files are in the floppy drive, type

```
dosdir
```

After you locate the Mission Manager file (*filename*) you want to process, read it into the current directory of the workstation by typing,

dosread *filename filename*

Since *filename* is in DOS format, you need to convert it into Unix format by typing the command

dos2unix *filename filename*

The above procedures are shown in Figure 4.

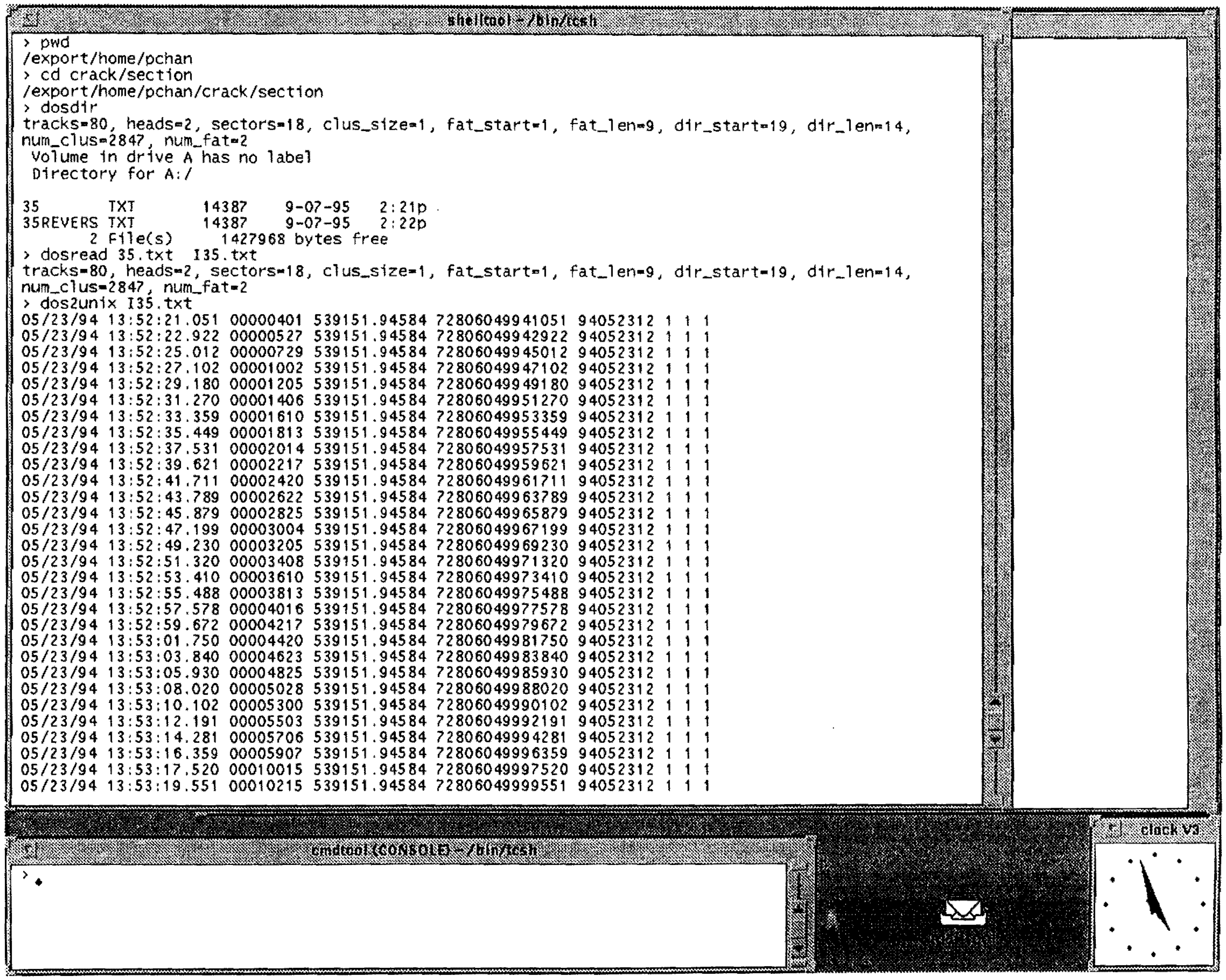
The crc95 program consists of 2 modes: a consecutive mode and an interactive mode. When you first start the program, you will be prompted to enter your request for either mode by typing a "**1**" or "**2**".

RUNNING THE crc95 PROGRAM IN CONFIGURED MODE

In the configured mode, it is assumed that there already existed a file in the "../crack/section" directory called "config.txt". If this file does not exist or is empty, you must request the interactive mode. "config.txt" has only 4 rows. The first row is the name of the Mission Manager File. The second row is the output file name used to store the summary of the analysis. The third row is the VCP speed (1, 2, 5, 10) where the numbers represent the number of frames skipped before processing the next one. The fourth row is the prompt for brief (type "1") or detail (enter anything else) result file. This mode is capable of handling Mission Manager Files with section numbers in either increasing or decreasing value with respect to DMI value (distance mileage reading). The following is a sample run of the "crc95" program in configured mode. Assuming the "config.txt" file exists in the "../crack/section" directory and that the Mission Manager File named "testreverse.txt" is also available, then the steps required to run the program are listing below.

Turn on the image processing unit and the VCP, in that order. Make sure the VCP settings are adjusted as described earlier on page seven.

> crc95



```
> pwd
/export/home/pchan
> cd crack/section
/export/home/pchan/crack/section
> dosdir
tracks=80, heads=2, sectors=18, clus_size=1, fat_start=1, fat_len=9, dir_start=19, dir_len=14,
num_clus=2847, num_fat=2
 Volume in drive A has no label
 Directory for A:/

35        TXT       14387    9-07-95   2:21p
35REVERS TXT        14387    9-07-95   2:22p
       2 File(s)      1427968 bytes free
> dosread 35.txt  I35.txt
tracks=80, heads=2, sectors=18, clus_size=1, fat_start=1, fat_len=9, dir_start=19, dir_len=14,
num_clus=2847, num_fat=2
> dos2unix I35.txt
05/23/94 13:52:21.051 00000401 539151.94584 72806049941051 94052312 1 1 1
05/23/94 13:52:22.922 00000527 539151.94584 72806049942922 94052312 1 1 1
05/23/94 13:52:25.012 00000729 539151.94584 72806049945012 94052312 1 1 1
05/23/94 13:52:27.102 00001002 539151.94584 72806049947102 94052312 1 1 1
05/23/94 13:52:29.180 00001205 539151.94584 72806049949180 94052312 1 1 1
05/23/94 13:52:31.270 00001406 539151.94584 72806049951270 94052312 1 1 1
05/23/94 13:52:33.359 00001610 539151.94584 72806049953359 94052312 1 1 1
05/23/94 13:52:35.449 00001813 539151.94584 72806049955449 94052312 1 1 1
05/23/94 13:52:37.531 00002014 539151.94584 72806049957531 94052312 1 1 1
05/23/94 13:52:39.621 00002217 539151.94584 72806049959621 94052312 1 1 1
05/23/94 13:52:41.711 00002420 539151.94584 72806049961711 94052312 1 1 1
05/23/94 13:52:43.789 00002622 539151.94584 72806049963789 94052312 1 1 1
05/23/94 13:52:45.879 00002825 539151.94584 72806049965879 94052312 1 1 1
05/23/94 13:52:47.199 00003004 539151.94584 72806049967199 94052312 1 1 1
05/23/94 13:52:49.230 00003205 539151.94584 72806049969230 94052312 1 1 1
05/23/94 13:52:51.320 00003408 539151.94584 72806049971320 94052312 1 1 1
05/23/94 13:52:53.410 00003610 539151.94584 72806049973410 94052312 1 1 1
05/23/94 13:52:55.488 00003813 539151.94584 72806049975488 94052312 1 1 1
05/23/94 13:52:57.578 00004016 539151.94584 72806049977578 94052312 1 1 1
05/23/94 13:52:59.672 00004217 539151.94584 72806049979672 94052312 1 1 1
05/23/94 13:53:01.750 00004420 539151.94584 72806049981750 94052312 1 1 1
05/23/94 13:53:03.840 00004623 539151.94584 72806049983840 94052312 1 1 1
05/23/94 13:53:05.930 00004825 539151.94584 72806049985930 94052312 1 1 1
05/23/94 13:53:08.020 00005028 539151.94584 72806049988020 94052312 1 1 1
05/23/94 13:53:10.102 00005300 539151.94584 72806049990102 94052312 1 1 1
05/23/94 13:53:12.191 00005503 539151.94584 72806049992191 94052312 1 1 1
05/23/94 13:53:14.281 00005706 539151.94584 72806049994281 94052312 1 1 1
05/23/94 13:53:16.359 00005907 539151.94584 72806049996359 94052312 1 1 1
05/23/94 13:53:17.520 00010015 539151.94584 72806049997520 94052312 1 1 1
05/23/94 13:53:19.551 00010215 539151.94584 72806049999551 94052312 1 1 1
```

```
>
```

Figure 4. Command Procedures as Seen in the Screen.

```
TO STOP THE PROGRAM AT ANY TIME, PRESS 'CTRL C'

Type '1' to process CONSECUTIVE sections.
Type '2' to process in INTERACTIVE MODE.
1

Mission Manager filename
output result filename
VCR speed
shortReportFlag
The above items must be read in this order as arranged in the 'config.txt' file

/dev/ttya opened

Mission Manager File Name read is 35reverse.txt.

Number of row entries in the Mission Manager File is : 190

The array arg corresponding to the moment the vehicle started moving is 67

Section numbers in DECREASING order detected!

Enter Begin Section Number (From  54 to  44 ):48

 Enter End Section Number (From  48 to  44 ):47
Begin timecode for Sec#48 is: 00041729
 End timecode for Sec#48 is: 00043013    Section length: 955.143738'
Begin timecode for Sec#47 is: 00043013
 End timecode for Sec#47 is: 00050528    Section length: 2663.306152'
00041729 541892.250000  48
00042001 541941.500000  48
00042204 541982.375000  48
00042406 542036.062500  48
00042609 542077.437500  48
```

```
00042811 542132.687500  48
00043013 542181.687500  47
00043216 542230.000000  47
00045919 542843.000000  47
00050125 542893.875000  47
00050325 542940.125000  47
00050528 542988.750000  46

Output File Name is result.txt.

VCR speed is 5.

SECTION  48 :
Begin timecode is 00041729       End timecode is 00043013
Distress type 3 Transverse at dmi 541895.4375 for time 41803
Distress type 6 Spalled Crack at dmi 541904.9375 for time 41815
Distress type 3 Transverse at dmi 541915.3125 for time 41828

Distress type 1 Intact at dmi 542149.3125 for time 42902
Distress type 1 Intact at dmi 542158.7500 for time 42914

Distress type 3 Transverse at dmi 542168.2500 for time 42926
Distress type 1 Intact at dmi 542178.5000 for time 43009

Reached END

SECTION  47 :
Begin timecode is 00043013       End timecode is 00050528
Distress type 1 Intact at dmi 542188.0000 for time 43021
Distress type 3 Transverse at dmi 542197.0000 for time 43103
Distress type 1 Intact at dmi 542207.0000 for time 43116
Distress type 3 Transverse at dmi 542216.1875 for time 43128
Distress type 3 Transverse at dmi 542938.5625 for time 50323
Distress type 3 Transverse at dmi 542947.8125 for time 50405
Distress type 3 Transverse at dmi 542957.1250 for time 50417
Distress type 3 Transverse at dmi 542966.3750 for time 50429
Distress type 3 Transverse at dmi 542975.6250 for time 50511
Distress type 3 Transverse at dmi 542985.6875 for time 50524

Reached END
Reached EOF
```

When the program ends, the results are stored in the output file specified in the second line of the "config.txt" file. It reads like the following

```
> more result.txt

Mission Manager File Name : 35reverse.txt

Output File Name : result.txt

VCR speed : 5.

SECTION  48 :

Begin timecode is 00041729      End timecode is 00043013

SUMMARY for SECTION 48
Transverse  Longitudinal  Spalled Crack  Intact  Avg Crack Spacing  Sec Length
   13          0            5          13        10.6         955.1

SECTION  47 :
Begin timecode is 00043013      End timecode is 00050528

SUMMARY for SECTION 47
Transverse  Longitudinal  Spalled Crack  Intact  Avg Crack Spacing  Sec Length
   38          1           12          36        10.7         2663.3

>
```

RUNNING THE crc95 PROGRAM IN INTERACTIVE MODE

In the interactive mode you will be prompted for the name of the Mission Manager File, the section numbers in the Mission Manager File to be processed, the output file name to store the results, the VCR speed, as well as the type of the result file (detail or brief). The section numbers can be entered in any order. A sample run is shown below.

```
> crc95

T0 STOP THE PROGRAM AT ANY TIME, PRESS 'CTRL C'

Type '1' to process CONSECUTIVE sections.
Type '2' to process in INTERACTIVE MODE.
2
/dev/ttya opened



Enter Mission Manager File Name : 35.txt

Number of row entries in the Mission Manager File is : 190

The array arg corresponding to the moment the vehicle started moving is 65

Enter Section Number (From  44 to  54 ):54
00061704 544622.562500 54
00061907 544678.562500 54
00062110 544728.187500 54
00062211 544750.875000 54
00062413 544800.562500 54
00062617 544849.625000 54
```

```
00062817 544896.625000 54
00063020 544946.375000 54

Begin timecode for Sec#54 is: 00061704
  End timecode for Sec#54 is: 00063020   Section length: 1068.581299'

Type '1' to enter more sections : 1

Enter Section Number (From  44 to  54 ):47
00032805 540739.562500 47
00033005 540789.250000 47
00033210 540838.125000 47
00033409 540883.375000 47
00033613 540929.625000 48

Begin timecode for Sec#47 is: 00032805
  End timecode for Sec#47 is: 00033613   Section length: 627.206238'

Type '1' to enter more sections : 1

Enter Section Number (From  44 to  54 ):50
00041729 541892.250000 50
00042001 541941.500000 50
00042204 541982.375000 50
00042406 542036.062500 50
00042609 542077.437500 50
00042811 542132.687500 50
00043013 542181.687500 51

Begin timecode for Sec#50 is: 00041729
  End timecode for Sec#50 is: 00043013   Section length: 955.143738'

Type '1' to enter more sections : 2
```

```
Enter Output File:35rst.txt

File NOT empty!
 Type '1' to APPEND New Result to it!
 Type '2' to OVERWRITE it!  2

Select VCR PLAY speed 1, 2, 5, 8 or 10 : 5

Type '1' to choose short Output File: 1

SECTION  54 :
Begin timecode is 00061704        End timecode is 00063020
Distress type 1 Intact at dmi 544634.1250 for time 61717
Distress type 6 Spalled Crack at dmi 544644.8125 for time 61729
Distress type 3 Transverse at dmi 544655.4375 for time 61811
Distress type 3 Transverse at dmi 544667.0000 for time 61824
Distress type 3 Transverse at dmi 544677.6875 for time 61906
Distress type 3 Transverse at dmi 544688.3125 for time 61918
Distress type 1 Intact at dmi 544696.6875 for time 62000
Distress type 3 Transverse at dmi 544905.2500 for time 62828
Distress type 1 Intact at dmi 544914.8125 for time 62910
Distress type 3 Transverse at dmi 544925.0625 for time 62923
Distress type 3 Transverse at dmi 544934.5000 for time 63005
Distress type 6 Spalled Crack at dmi 544944.0000 for time 63017

Reached END

SECTION  47 :
Begin timecode is 00032805        End timecode is 00033613
Distress type 1 Intact at dmi 540750.3125 for time 32818
Distress type 1 Intact at dmi 540761.1250 for time 32901
Distress type 2 Longitudinal at dmi 540892.7500 for time 33422
Distress type 3 Transverse at dmi 540901.4375 for time 33504
Distress type 3 Transverse at dmi 540910.1250 for time 33516
Distress type 6 Spalled Crack at dmi 540919.5000 for time 33529
Distress type 3 Transverse at dmi 540928.1875 for time 33611

Reached END

SECTION  50 :
Begin timecode is 00041729       End timecode is 00043013
Distress type 1 Intact at dmi 541902.5625 for time 41812
Distress type 1 Intact at dmi 541912.1250 for time 41824
Distress type 1 Intact at dmi 541921.6250 for time 41906
Distress type 1 Intact at dmi 542146.9375 for time 42829
Distress type 3 Transverse at dmi 542157.1875 for time 42912
Distress type 1 Intact at dmi 542166.6875 for time 42924
Distress type 3 Transverse at dmi 542176.1250 for time 43006

Reached END
Reached EOF
```

The corresponding output file has the following information in it.

```
> more 35rst.txt

Mission Manager File read is : 35.txt
          Output File is : 35rst.txt
VCR PLAY speed is : 5
SECTION  54 :
Begin timecode is 00061704       End timecode is 00063020

SUMMARY for SECTION 54
Transverse  Longitudinal  Spalled Crack  Intact  Avg Crack Spacing  Sec Length
   15          0            3           15        11.9          1068.6
```

```
SECTION  47 :
Begin timecode is 00032805        End timecode is 00033613

SUMMARY for SECTION 47
Transverse  Longitudinal  Spalled Crack  Intact  Avg Crack Spacing  Sec Length
     5          1            2          12        17.9          627.2



SECTION  50 :
Begin timecode is 00041729        End timecode is 00043013

SUMMARY for SECTION 50
Transverse  Longitudinal  Spalled Crack  Intact  Avg Crack Spacing  Sec Length
    13          0            1          16        13.6          955.1



SUMMARY for SECTION 50
Transverse  Longitudinal  Spalled Crack  Intact  Avg Crack Spacing  Sec Length
     0          0            0           0        NaN            0.0


>
```

CHAPTER IV
SUMMARY

A Pavement Video Image Processing System has been developed to detect and classify distresses on pavements. The system takes as input videos of pavements and Mission Manager files recorded using TxDOT's Multi Functional Vehicle. Each video-taped pavement picture is associated with a time code which, in turn, ties in with footage information in the Mission Manager file. The analysis algorithm reads information about time code, mileage (dmi) and section number from the Mission Manager file. This information is then used to locate the section from the video tape, digitize the pavement image and employ embedded image processing techniques to detect and classify cracks on the pavement image.

In this manual, the function of this system is described. System setup, configuration and inter-equipment connection are shown. Detailed procedures are also presented to explain how the crc95 software should be run in both Configured Mode or Interactive Mode. The discrimination rules for distress detection and classification behind the software is also included in the appendix for reference.

SUGGESTION FOR FUTURE WORK

Although the current version of the software is reasonably user friendly, there is still room for improvement. One natural move is to convert this software into a Windows-driven, drag-and-drop type application. Windows-driven applications are easier to learn and faster to operate. They can eliminate the chance of typing mistakes.

As far as the matter of accuracy is concerned, more projection histograms at oblique angles across the image can be employed to more thoroughly check for distress within the image. Currently, only two histograms are performed, one vertical and one horizontal. With the availability of faster SPARC workstations, more histogram calculations can be performed without slowing down the analysis speed.

APPENDIX A

PROGRAM LISTING OF crc95

Program listing of crc95

```
/*************************************************
Program : conf_integ_prc.c (configurated program for processing consecutive sections continously.
                              It checks whether Begin and End section numbers are in order. Therefore,
                              it will NOT process files with dmi values reversed)
               New name of the program: crc95.c
                              The new program has the feature of integrating the INTERACTIVE and CONFIGURED MODE
                              together so that when any of the required files for the CONFIGURED MODE are missing,
                              INTERACTIVE MODE will be invoked.

Author  : Texas Transportation Institute

Created Date  : March 1, 1993

Modified Date : -remove the need of manually preparing the timecode_dmi file
                              and the timecode file. Read in only the Mission Manager file.(10/19/94)
                              -allow choice of different VCR Play speeds. (10/25/94)
                              -bug fixed:
                              array dimensions are increased to handle large Mission Manager files(06/26/95)
                              -combined the interactive mode with the configured mode together so that when a
                              required file is missing in the configured mode, interactive mode will be invoked
                              automatically.(07/21/95)
                              -add the capability of detecting whether section#'s are in decreasing order and can
                              process in such order.(07/22/95)
                              -bug fixed:
                              in the INTERACTIVE MODE, for Mission Manager files in decreasing order section#'s,
                              the program kept asking for "input section#"(08/03/95).


*************************************************/
#include <stdio.h>
#include <stdlib.h>
#include <sys/termio.h>
#include <fcntl.h>
#include <itex150.h>
#include <ipa.h>
#include <sys/types.h>
#include <sys/timeb.h>
#include "/export/lang/SC2.0.1/include/cc_411/time.h"


/*************************************************

GLOBALS

*************************************************/
int hex_time[4];
int dev1;
FILE *configuration;
```

```
char input[15];

int open_port(name)
char *name;
{

        int dev1;
        struct termios dev1_mode;

        if (( dev1=open(name,O_RDWR))==EOF)
        {
                printf("%s not opened\n",name);
                exit(1);
        }
        printf("%s opened \n",name);
        ioctl(dev1,TCGETS,&dev1_mode);
        dev1_mode.c_iflag=IGNBRK|ICRNL|IXANY;
        dev1_mode.c_oflag=OPOST|ONLCR;
        dev1_mode.c_lflag=ISIG;
        dev1_mode.c_cflag = B38400|CS8|CREAD|PARENB|PARODD;
        dev1_mode.c_cc[VINTR]=0;
        dev1_mode.c_cc[VQUIT]=0;
        dev1_mode.c_cc[VERASE]=0;
        dev1_mode.c_cc[VKILL]=0;
        dev1_mode.c_cc[VEOF]=0;
        dev1_mode.c_cc[VEOL]=0;
        dev1_mode.c_cc[VEOL2]=0;
        dev1_mode.c_cc[VSWTCH]=0;
        dev1_mode.c_cc[VSTART]=0;
        dev1_mode.c_cc[VSTOP]=0;
        dev1_mode.c_cc[VSUSP]=0;
        dev1_mode.c_cc[VREPRINT]=0;
        dev1_mode.c_cc[VDISCARD]=0;
        dev1_mode.c_cc[VWERASE]=0;
        dev1_mode.c_cc[VLNEXT]=0;
        dev1_mode.c_cc[VMIN]=0;
        dev1_mode.c_cc[VTIME]=0;

        if ( ioctl(dev1,TCSETS,&dev1_mode) < 0){
                printf("Error Setting hardware flow control \n");
                exit(1);
        }
        return(dev1);
}



int get_ltc()
{
        BYTE out_buf[10];
        BYTE in_buf[7];
        int hr, min, sec, fr;

        out_buf[0]=0x61;
        out_buf[1]=0x0C;
        out_buf[2]=0x01;
        out_buf[3]=out_buf[0]+out_buf[1]+out_buf[2];
        write(dev1,out_buf,4);
        sleep(1);
        read(dev1,in_buf,7);
        while((in_buf[0]!=0x74 && in_buf[1]!=0x14)||(in_buf[0]!=0x74 && in_buf[1]!=0x04)) read(dev1,in_buf,7);
        in_buf[2]= in_buf[2]&0x3F;
        hr=(int) (in_buf[5]/16)*10 + in_buf[5]%16;
        min=(int) (in_buf[4]/16)*10 + in_buf[4]%16;
        sec=(int) (in_buf[3]/16)*10 + in_buf[3]%16;
        fr=(int) (in_buf[2]/16)*10 + in_buf[2]%16;
        return(1000000*hr+10000*min+100*sec+fr);
}


convert_to_hex(time)
int time;
{
        int temp;


        temp=(int) time%100;
        temp=(temp/10)*16+temp%10;
        hex_time[3]=(int) temp;

        temp= (int) time%10000/100;
        temp=(temp/10)*16+temp%10;
        hex_time[2]= temp;

        temp= (int) time%1000000/10000;
        temp=(temp/10)*16+temp%10;
        hex_time[1]= temp;

        temp= (int) time%100000000/1000000;
        temp=(temp/10)*16+temp%10;
        hex_time[0]= temp;

}



void cue_up(time)
int time;
{
        BYTE out_buf[10];
        BYTE in_buf[7];

        convert_to_hex(time);
```

```
            out_buf[0]=0x24;
            out_buf[1]=0x31;
            out_buf[2]=(BYTE) hex_time[3];
            out_buf[3]=(BYTE) hex_time[2];
            out_buf[4]=(BYTE) hex_time[1];
            out_buf[5]=(BYTE) hex_time[0];

            out_buf[6]=out_buf[0]+out_buf[1]+out_buf[2]+out_buf[3]+out_buf[4]+out_buf[5];
            write(dev1,out_buf,7);
            sleep(7);
            read(dev1,in_buf,3);
}


BYTE status()
{
            BYTE out_buf[10];
            BYTE in_buf[7];

            out_buf[0]=0x61;
            out_buf[1]=0x20;
            out_buf[2]=0x21;
            out_buf[3]=out_buf[0]+out_buf[1]+out_buf[2];
            write(dev1,out_buf,4);
            sleep(1);
            read(dev1,in_buf,4);
            return(in_buf[2]);
}


void stop()
{
            BYTE out_buf[10];
            BYTE in_buf[7];

            out_buf[0]=0x20;
            out_buf[1]=0x00;
            out_buf[2]=out_buf[0]+out_buf[1];
            write(dev1,out_buf,3);
            sleep(1);
            read(dev1,in_buf,2);
}

void slow_play1()
{
            BYTE out_buf[10];
            BYTE in_buf[7];

            out_buf[0]=0x21;
            out_buf[1]=0x12;
            out_buf[2]=0x11;
            out_buf[3]=out_buf[0]+out_buf[1]+out_buf[2];
```

```
		write(dev1,out_buf,4);
		sleep(1);
		read(dev1,in_buf,7);
}

void slow_play2()
{
		BYTE out_buf[10];
		BYTE in_buf[7];

		out_buf[0]=0x21;
		out_buf[1]=0x12;
		out_buf[2]=0x1A;
		out_buf[3]=out_buf[0]+out_buf[1]+out_buf[2];
		write(dev1,out_buf,4);
		sleep(1);
		read(dev1,in_buf,7);
}

void slow_play5()
{
		BYTE out_buf[10];
		BYTE in_buf[7];

		out_buf[0]=0x21;
		out_buf[1]=0x12;
		out_buf[2]=0x26;
		out_buf[3]=out_buf[0]+out_buf[1]+out_buf[2];
		write(dev1,out_buf,4);
		sleep(1);
		read(dev1,in_buf,7);
}

void slow_play8()
{
		BYTE out_buf[10];
		BYTE in_buf[7];

		out_buf[0]=0x21;
		out_buf[1]=0x12;
		out_buf[2]=0x2C;
		out_buf[3]=out_buf[0]+out_buf[1]+out_buf[2];
		write(dev1,out_buf,4);
		sleep(1);
		read(dev1,in_buf,7);
}

void slow_play10()
{
		BYTE out_buf[10];
		BYTE in_buf[7];
		out_buf[0]=0x21;
		out_buf[1]=0x12;
		out_buf[2]=0x30;
		out_buf[3]=out_buf[0]+out_buf[1]+out_buf[2];
		write(dev1,out_buf,4);
		sleep(1);
		read(dev1,in_buf,7);
}



int compare_time(file_time, tape_time)
int file_time, tape_time;
{
		int hr, min,sec,fr;
		int filetime, tapetime;

		fr=(int) file_time%100;
		sec=(int) file_time%10000/100;
		min=(int) file_time%1000000/10000;
		hr=(int) file_time%100000000/1000000;
		filetime=hr*60*60*30+min*60*30+sec*30+fr;

		fr=(int) tape_time%100;
		sec=(int) tape_time%10000/100;
		min=(int) tape_time%1000000/10000;
		hr=(int) tape_time%100000000/1000000;
		tapetime=hr*60*60*30+min*60*30+sec*30+fr;

		if (filetime<=tapetime) return 1;
		else return 2;
}


float interpolate(behind_time, ahead_time, tape_time, behind_dmi,ahead_dmi)
int behind_time, ahead_time, tape_time;
float behind_dmi,ahead_dmi;
{
		int aheadfiletime, behindfiletime, tapetime;
		int hr, min, sec, fr;
		float extra, newdmi;

		fr=(int) ahead_time%100;
		sec=(int) ahead_time%10000/100;
		min=(int) ahead_time%1000000/10000;
		hr=(int) ahead_time%100000000/1000000;
		aheadfiletime=hr*60*60*30+min*60*30+sec*30+fr;

		fr=(int) behind_time%100;
		sec=(int) behind_time%10000/100;
		min=(int) behind_time%1000000/10000;
		hr=(int) behind_time%100000000/1000000;
		behindfiletime=hr*60*60*30+min*60*30+sec*30+fr;

		fr=(int) tape_time%100;
		sec=(int) tape_time%10000/100;
		min=(int) tape_time%1000000/10000;
		hr=(int) tape_time%100000000/1000000;
		tapetime=hr*60*60*30+min*60*30+sec*30+fr;


		extra=((float) (tapetime-behindfiletime))/((float) (aheadfiletime-behindfiletime));
		newdmi=behind_dmi+(ahead_dmi-behind_dmi)*extra;
		return((float) newdmi);
}




int i_crc(img_num,LMNX,LMNY,nLMNX,nLMNY, stats)
float LMNX,LMNY,nLMNX,nLMNY;
int img_num;
float *stats;
{
		int fcb_num,retval;

		fcb_num=fcb_make("_i_crc", ASYNC, img_num, LMNX,LMNY,nLMNX,nLMNY, 2, stats);
		/*fcb_num=fcb_create("_i_crc",0); */
		if (fcb_num<0) return fcb_num;
		/*fcb_arg(fcb_num,1,(int) img_num);
    fcb_arg(fcb_num,2,(float) LMNX);
    fcb_arg(fcb_num,3,(float) LMNY);
    fcb_arg(fcb_num,4,(float) nLMNX);
    fcb_arg(fcb_num,5,(float) nLMNY); */
		retval=fcb_exec(fcb_num);
		fcb_delete(fcb_num);
		return(retval);
}

void show ()
{
		alu_setop(UNSIGNED,PASS_A,0);
		alu_selwxyz(VPIH,VPIL,CONSTANT,CONSTANT);
		ipa_show(BLW);
		select_path(B1);
}


void init()
{
		load_cfg("/usr/ITEX/150/lib/std.cfg");
		initsys();
		adi_hbanksel(15);
		adi_hgroupsel(RED|GREEN|BLUE);
		adi_clearlut(255);
		fb_clf(B1,0);
		fb_clf(B2,0);
		adi_lutmode(DYNAMIC);
		contour(ADI,GREEN,1,0,156,255);
		linlut(ADI,RED|GREEN|BLUE,0);
}


int my_exit(input_file, output_file)
FILE *input_file, *output_file;
{
		fclose(output_file);
		fclose(input_file);
		stop();
		exit(0);
}


void write_to_file(output_file, nintact, nlong, ntrans, nalli, ninter, nspall, SecLength, i, speed)
FILE *output_file;
int nintact ,nlong, ntrans, nalli, ninter, nspall, i, speed;
float SecLength;
{
		float AvgCrack=0.0;

		AvgCrack=SecLength/(ntrans+nspall)/speed;

    fprintf(output_file, "\nSUMMARY for SECTION %d\n", i);
    fprintf(output_file, "Transverse  Longitudinal  Spalled Crack  Intact  Avg Crack Spacing  Sec Length\n");
    fprintf(output_file, "   %3d      %2d       %3d       %3d      %5.1f       %7.1f\n\n", ntrans, nlong, nspall, nintact, AvgCrack, SecLength);
}

void main2();

void preprc1()
{
		FILE *input_file, *output_file, *tc_dmi, *section_tc;
		int tc[300], sec_no[300], pav_type, lane, sec_tc[10], section, last_section;
		int i=0, j=0, section_no, bgn_arry_arg=0, no_of_record, bgn_indx=0, end_indx=0;
		int decreasing_sect_no_flag;
		float dmi[300];
		float sectionLength=0.0;
		char date[8], time[12], index[14], tape_no[8], output[15];


		fscanf( configuration, "%s\n", input );
		if( (input_file=fopen( input, "rb"))==NULL )
		{
				printf("\nMission Manager File is empty or does not exist!\n");
				fclose(input_file);
				printf("\nINTERACTIVE MODE invoked\n\n\n");
```

```
                        main2();
                        exit(0);
            }
            printf( " \nMission Manager File Name read is %s.\n", input );

            tc_dmi=fopen( "tc_dmi.dat", "wb");
            section_tc=fopen( "sec_tc.dat", "wb");

            i=1;
            while( fscanf(input_file, "%08s %012s %08u %f %014s %08s %d %d %d\n",
            date, time, &tc[i], &dmi[i], index, tape_no, &sec_no[i], &pav_type, &lane) != EOF )
            {
                        i++;
            }
            fclose( input_file );

            no_of_record = i-4;
            printf( " \nNumber of row entries in the Mission Manager File is : %d\n", no_of_record );

            i=1;
            while( dmi[i]==dmi[1] )
            {
                        i++;
            }
            bgn_arry_arg = i;
            printf( "\nThe array arg corresponding to the moment the vehicle started moving is %d\n", bgn_arry_arg );

            /* Check whether the beginning section # is bigger than the last section # */
            decreasing_sect_no_flag = 1;
            if( (sec_no[bgn_arry_arg+2] - sec_no[no_of_record]) > 0 )
            {
                        decreasing_sect_no_flag = 2;
                        printf("\nSection numbers in DECREASING order detected!\n" );
            }


  switch (decreasing_sect_no_flag) {
  case 1:
                        do
                        {
                                    printf( "\nEnter Begin Section Number (From %3d to %3d ):", sec_no[bgn_arry_arg+2], sec_no[no_of_record] );
                                    scanf( "%d", &section );
            label2:     printf( "\n  Enter End Section Number (From %3d to %3d ):", section, sec_no[no_of_record] );
                                    scanf( "%d", &last_section );
                                    if( last_section<section | last_section>sec_no[no_of_record-1] )
                                    {
                                                printf( " \nEnd Section Number must BIGGER than Begin Section Number\nand SMALLER than %3d !\n",
sec_no[no_of_record] );
                                                goto label2;
                                    }
                        } while( section<sec_no[bgn_arry_arg+2] | section>sec_no[no_of_record-1] );
                        getchar();
```

```
        i=bgn_arry_arg;
        while( sec_no[i]<section )
        {
                i++;
        }
        bgn_arry_arg=i;
        fprintf( section_tc, "%08u\n", tc[bgn_arry_arg]);

        for( j=section; j<=last_section; j++ )
        {
                i=bgn_arry_arg;
                while( sec_no[i]<j )
                {
                        i++;
                }
                bgn_indx=i;

                /* check how many rows in the Mission Manager File with the same section no.
                 Then use this as the end index of the section*/
                i=bgn_indx;
                while( sec_no[i]==j )
                {
                        i++;
                }
                end_indx=i;
                sectionLength=3.3*(dmi[end_indx]-dmi[bgn_indx]);
                if( last_section==sec_no[no_of_record] )
                {
                        end_indx--;
                        sectionLength=3.3*(dmi[end_indx]-dmi[bgn_indx]);
                }
                printf( "Begin timecode for Sec#%d is: %08u\n", j, tc[bgn_indx] );
                printf( "  End timecode for Sec#%d is: %08u\t  Section length: %f\n", j, tc[end_indx], sectionLength );
                fprintf( section_tc, "%08u %f %3d\n", tc[end_indx], sectionLength, sec_no[end_indx-1] );
        }
        fclose( section_tc );

        for( i=bgn_arry_arg; i<=end_indx; i++ )
        {
                printf( "%08u %f %3d\n", tc[i], dmi[i], sec_no[i] );
                fprintf( tc_dmi, "%08u %f\n", tc[i], dmi[i] );
        }
        fclose( tc_dmi );

        break;

case 2:
                do
                {
                        printf( "\nEnter Begin Section Number (From %3d to %3d ):", sec_no[bgn_arry_arg+2], sec_no[no_of_record] );
                        scanf( "%d", &section );
```

```
		label3:		printf( "\n  Enter End Section Number (From %3d to %3d ):", section, sec_no[no_of_record] );
				scanf( "%d", &last_section );
				if( last_section>section | last_section<sec_no[no_of_record-1] )
				{
					printf( " \nEnd Section Number must SMALLER than Begin Section Number\nand BIGGER than %3d !\n",
sec_no[no_of_record] );
					goto label3;
				}
			} while( section>sec_no[bgn_arry_arg+2] | section<sec_no[no_of_record-1] );
			getchar();

		i=bgn_arry_arg;
		while( sec_no[i]>section )
		{
			i++;
		}
		bgn_arry_arg=i;
		fprintf( section_tc, "%08u\n", tc[bgn_arry_arg]);

		for(j=section; j>=last_section;  j-- )
		{
			i=bgn_arry_arg;
			while( sec_no[i]>j )
			{
				i++;
			}
			bgn_indx=i;

			/* check how many rows in the Mission Manager File with the same section no.
			 Then use this as the end index of the section*/
			i=bgn_indx;
			while( sec_no[i]==j )
			{
				i++;
			}
			end_indx=i;
			sectionLength=3.3*(dmi[end_indx]-dmi[bgn_indx]);
			if( last_section==sec_no[no_of_record] )
			{
				end_indx--;
				sectionLength=3.3*(dmi[end_indx]-dmi[bgn_indx]);
			}
			printf( "Begin timecode for Sec#%d is: %08u\n", j, tc[bgn_indx] );
			printf( "  End timecode for Sec#%d is: %08u\t   Section length: %f\n", j, tc[end_indx], sectionLength );
			fprintf( section_tc, "%08u %f %3d\n", tc[end_indx], sectionLength, sec_no[end_indx-1] );
		}
		fclose( section_tc );

		for( i=bgn_arry_arg; i<=end_indx; i++ )
		{
			printf( "%08u %f %3d\n", tc[i], dmi[i], sec_no[i] );
			fprintf( tc_dmi, "%08u %f\n", tc[i], dmi[i] );
```

```
            }
            fclose( tc_dmi );

            break;
  }
}

void preprc2()
{
            FILE *input_file, *tc_dmi, *section_tc;
            int tc[10000], sec_no[10000], pav_type, lane, sec_tc[500], section, last_section;
            int i=0, j=0, section_no, bgn_arry_arg=0, no_of_record, bgn_indx=0, end_indx=0;
            float dmi[10000], sectionLength=0.0;
            char date[8], time[12], index[14], tape_no[8], output[15];
            int moreEntryFlag, decreasing_sect_no_flag;


label1:     printf( " \n\n\nEnter Mission Manager File Name : " );
            scanf( "%s", input );
            getchar();

            if( (input_file=fopen( input, "rb") ) == NULL )
            {
                        printf( " No such a file or file is empty.\n" );
                        goto label1;
            }

            tc_dmi=fopen( "tc_dmi.dat", "wb");
            section_tc=fopen( "sec_tc.dat", "wb");

            i=1;
            while( fscanf(input_file, "%08s %012s %08u %f %014s %08s %d %d %d\n",
            date, time, &tc[i], &dmi[i], index, tape_no, &sec_no[i], &pav_type, &lane) != EOF )
            {
                        i++;
            }
            fclose( input_file );

            no_of_record = i-4;
            printf( " \nNumber of row entries in the Mission Manager File is : %d\n", no_of_record );

            i=1;
            while( dmi[i]==dmi[1] )
            {
                        i++;
            }
            bgn_arry_arg = i;
            printf( "\nThe array arg corresponding to the moment the vehicle started moving is %d\n", bgn_arry_arg );

            /* Check whether the beginning section # is bigger than the last section # */
            decreasing_sect_no_flag = 1;
            if( (sec_no[bgn_arry_arg+2] - sec_no[no_of_record]) > 0 )
```

```
{
            decreasing_sect_no_flag = 2;
            printf("\nSection numbers in DECREASING order detected!\n" );
}


do
{
  switch(decreasing_sect_no_flag){
  case 1:
            do
    {
                        printf( "\nEnter Section Number (From %3d to %3d ):", sec_no[bgn_arry_arg+2], sec_no[no_of_record] );
                        scanf( "%d", &section );
    } while( section<sec_no[bgn_arry_arg+2] || section>sec_no[no_of_record-1] );
            getchar();
            i=bgn_arry_arg;
            while( sec_no[i]<section )
            {
                        i++;
            }
            bgn_indx=i;
            break;
  case 2:
            do
    {
                        printf( "\nEnter Section Number (From %3d to %3d ):", sec_no[bgn_arry_arg+2], sec_no[no_of_record] );
                        scanf( "%d", &section );
    } while( section>sec_no[bgn_arry_arg+2] || section<sec_no[no_of_record-1] );
            getchar();
            i=bgn_arry_arg;
            while( sec_no[i]>section )
            {
                        i++;
            }
            bgn_indx=i;
            break;
 }

            i=bgn_indx;
            while( sec_no[i]==section )
            {
                        i++;
            }
            end_indx=i;
            sectionLength=3.3*(dmi[end_indx]-dmi[bgn_indx]);
            if( section==sec_no[no_of_record] )
            {
                        end_indx--;
                        sectionLength=3.3*(dmi[end_indx]-dmi[bgn_indx]);
            }
```

```
                    for( i=bgn_indx; i<=end_indx; i++ )
                    {
                                printf( "%08u %f %d\n", tc[i], dmi[i], sec_no[i] );
                                fprintf( tc_dmi, "%08u %f\n", tc[i], dmi[i] );
                    }

                    printf( "\nBegin timecode for Sec#%d is: %08u", section, tc[bgn_indx] );
                    printf( "\n  End timecode for Sec#%d is: %08u\t Section length: %f\n", section, tc[end_indx], sectionLength );
                    fprintf( section_tc, "%08u %08u %f %d\n", tc[bgn_indx], tc[end_indx], sectionLength, sec_no[bgn_indx] );

                    printf( "\nType '1' to enter more sections : ");
                    scanf( "%d", &moreEntryFlag );
                    getchar();

          } while( moreEntryFlag == 1 );

          fclose( tc_dmi );
          fclose( section_tc );
}



void main1()
{

          FILE *fopen(), *input_file, *output_file, *shape, *section_tc;
          int hr, min, sec, fr;
          int hr1, min1, sec1, fr1;
          BYTE state;
          float LMNX,LMNY,nLMNX,nLMNY;
          float newdmi;
          int nintact,ntrans,nalli,nlong,ninter,nspall;
          float SecLen;
          int i, img_num, count;
          float STARTDMI, ENDDMI;
          int BEGINTC, ENDTC;
          int tape_time;
          int file_time;
          float file_dmi;
          float tape_dmi;
          int behind_ptr_time;
          int ahead_ptr_time;
          float behind_ptr_dmi;
          float ahead_ptr_dmi;
          char output[20], sectiontc[20];
          float stats[2];
          int time_section;
          int section_no;
          char *class="";
          int shortFileFlag, overWriteFlag;
          int speed;
```

```
        dev1=open_port("/dev/ttya");
        init();
        initsys();
        nintact=ntrans=nalli=nlong=ninter=nspall=0;

        load_umodule("/export/home/pchan/crack/itexm.mcx");
        shape=fopen("/export/home/pchan/crack/lmn_crc","r");
        fscanf(shape,"%f %f %f %f",&LMNX,&LMNY,&nLMNX,&nLMNY);
        fclose(shape);
        count=0;
        img_num=0;


        chdir("/export/home/pchan/crack/section");

        if( (configuration=fopen( "config.txt", "rb")) == NULL)
        {
                printf("\n'config.txt' doesn't exist or is empty\n");
                printf("\nInvoking INTERACTIVE MODE...\n\n\n");
                main2();
        }

        preprc1();

        fscanf( configuration, "%s\n", output );
        output_file=fopen( output, "wb");
        printf( "\nOutput File Name is %s.\n", output );
        fprintf( output_file,"\nMission Manager File Name : %s\n", input );
        fprintf( output_file,"\nOutput File Name : %s\n", output );

        input_file=fopen("tc_dmi.dat", "rb");
        section_tc=fopen("sec_tc.dat", "rb");

        fscanf( configuration, "%d\n", &speed );
        printf( " \nVCR speed is %d.\n", speed );
        fprintf( output_file, "\nVCR speed : %d.\n", speed );

        fscanf( configuration, "%d", &shortFileFlag );

        fclose( configuration );


        if (fscanf(section_tc,"%d\n",&time_section)==EOF){
    printf("Reached EOF\n");
                write_to_file(output_file, nintact ,nlong, ntrans, nalli,ninter, nspall, SecLen, section_no, speed);
                my_exit(input_file, output_file);
}

        BEGINTC=time_section;

        if (fscanf(section_tc,"%d %f %3d\n",&time_section, &SecLen, &section_no)==EOF){
    printf("Reached EOF\n");
```

```
        my_exit(input_file, output_file);
    }

         ENDTC=time_section;
         printf( "\nSECTION %3d :\nBegin timecode is %08u\t End timecode is %08u\n", section_no, BEGINTC, ENDTC );
         fprintf( output_file, "\nSECTION %3d :\n\nBegin timecode is %08u\t End timecode is %08u\n", section_no, BEGINTC, ENDTC );

         if (fscanf(input_file,"%d %f\n",&file_time, &file_dmi)==EOF){
                 printf("Reached EOF\n");
                 my_exit(input_file, output_file);
         }
         behind_ptr_time=file_time;
         behind_ptr_dmi=file_dmi;
         if (fscanf(input_file,"%d %f\n",&file_time, &file_dmi)==EOF){
                 printf("Reached EOF\n");
                 my_exit(input_file, output_file);
         }
         ahead_ptr_time=file_time;
         ahead_ptr_dmi=file_dmi;

         while (compare_time(file_time, BEGINTC)!=2){
                 if (fscanf(input_file,"%d %f\n",&file_time, &file_dmi)==EOF){
                         printf("Reached EOF\n");
                         my_exit(input_file, output_file);
                 }
                 behind_ptr_time=ahead_ptr_time;
                 behind_ptr_dmi=ahead_ptr_dmi;
                 ahead_ptr_time=file_time;
                 ahead_ptr_dmi=file_dmi;
         }



         cue_up(BEGINTC);
         state=status()&0x01;
         while(state==0x0) state=status()&0x01;


         switch(speed)
         {
         case 1:
                 slow_play1();
                 break;
         case 2:
                 slow_play2();
                 break;
         case 5:
                 slow_play5();
                 break;
         case 8:
                 slow_play8();
                 break;
```

```
case 10:
            slow_play10();
            break;
}


stats[0]=120.0;
stats[1]=30.0;

for(;;){

            tape_time=get_ltc();
            snap(B1);

            ipa_snap(0,VDB,BLW);


            i=i_crc(img_num,LMNX,LMNY,nLMNX,nLMNY, stats);
            newdmi=interpolate(behind_ptr_time,ahead_ptr_time,tape_time,behind_ptr_dmi,ahead_ptr_dmi);
            if (compare_time(tape_time, ENDTC)==2){
                        printf("Reached END\n");
                        write_to_file(output_file, nintact ,nlong, ntrans, nalli,ninter, nspall, SecLen, section_no, speed);
                        SecLen=nintact=ntrans=nalli=nlong=ninter=nspall=0;
                        fclose(output_file);
                        BEGINTC=ENDTC;
                        output_file=fopen(output,"a");
                        if (fscanf(section_tc,"%d %f %3d\n",&time_section, &SecLen, &section_no)==EOF){
   printf("Reached EOF\n");
            my_exit(input_file, output_file);
            }
                        ENDTC=time_section;
                        printf( "\nSECTION %3d :\nBegin timecode is %08u\t  End timecode is %08u\n", section_no, BEGINTC, ENDTC );
                        fprintf( output_file, "\nSECTION %3d :\nBegin timecode is %08u\t  End timecode is %08u\n", section_no, BEGINTC, ENDTC );
            }

            if ( shortFileFlag == 1 )
            {
                        switch(i){
                        case 1:
                                    nintact++;
                                    class="Intact";
                                    break;
                        case 2:
                                    nlong++;
                                    class="Longitudinal";
                                    break;
                        case 3:
                                    ntrans++;
                                    class="Transverse";
                                    break;
                        case 4:
```

```
                        nalli++;
                        class="Alligator";
                        break;
                case 5:
                        ninter++;
                        class="Intersect";
                        break;
                case 6:
                        nspall++;
                        class="Spalled Crack";
                        break;
                }
}
else
{
                switch(i){
                case 1:
                        nintact++;
                        class="Intact";
                        break;
                case 2:
                        nlong++;
                        fprintf(output_file, "%.4f %d %s\n", newdmi, tape_time, "1");
                        class="Longitudinal";
                        break;
                case 3:
                        ntrans++;
                        fprintf(output_file, "%.4f %d %s\n", newdmi, tape_time, "2");
                        class="Transverse";
                        break;
                case 4:
                        nalli++;
                        fprintf(output_file, "%.4f %d %s\n", newdmi, tape_time, "3");
                        class="Alligator";
                        break;
                case 5:
                        ninter++;
                        class="Intersect";
                        break;
                case 6:
                        nspall++;
                        fprintf(output_file, "%.4f %d %s\n", newdmi, tape_time, "4");
                        class="Spalled Crack";
                        break;
                }
}

printf("Distress type %d %s at dmi %.4f for time %d \n\n", i, class, newdmi, tape_time);
while (compare_time(ahead_ptr_time, tape_time)!=2){
                if (fscanf(input_file,"%d %f\n",&file_time, &file_dmi)==EOF){
                printf("Reached EOF\n");
                write_to_file(output_file, nintact ,nlong, ntrans, nalli,ninter, nspall, SecLen, section_no, speed);
```

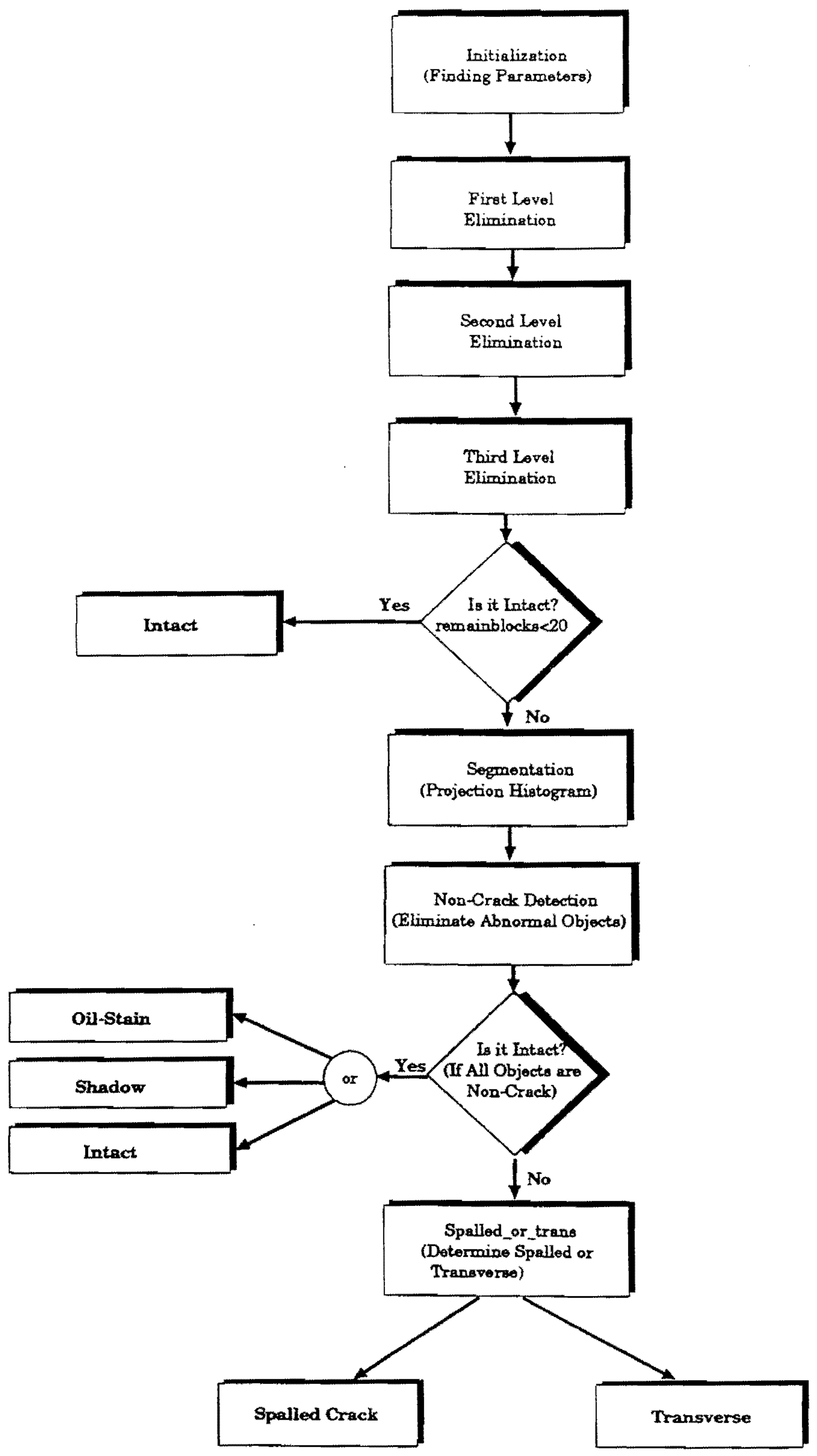
```
                                        my_exit(input_file, output_file);
                                        }
                                        behind_ptr_time=ahead_ptr_time;
                                        behind_ptr_dmi=ahead_ptr_dmi;
                                        ahead_ptr_time=file_time;
                                        ahead_ptr_dmi=file_dmi;
                              }
                    }

}


void main2()
{

          FILE *fopen(), *input_file, *output_file, *shape, *section_tc;
          int hr, min, sec, fr;
          int hr1, min1, sec1, fr1;
          BYTE state;
          float LMNX,LMNY,nLMNX,nLMNY;
          float newdmi;
          int nintact,ntrans,nalli,nlong,ninter,nspall;
          float SecLen;
          int i, img_num, count;
          float STARTDMI, ENDDMI;
          int BEGINTC, ENDTC;
          int tape_time;
          int file_time;
          float file_dmi;
          float tape_dmi;
          int behind_ptr_time;
          int ahead_ptr_time;
          float behind_ptr_dmi;
          float ahead_ptr_dmi;
          char output[20], sectiontc[20];
          float stats[2];
          int bgn_tc, end_tc;
          int section_no=0;
          char *class="";
          int shortFileFlag, overWriteFlag;
          int speed;


          dev1=open_port("/dev/ttya");
          init();
          initsys();
          nintact=ntrans=nalli=nlong=ninter=nspall=0;

          load_umodule("/export/home/pchan/crack/itexm.mcx");
          shape=fopen("/export/home/pchan/crack/lmn_crc","r");
          fscanf(shape,"%f %f %f %f",&LMNX,&LMNY,&nLMNX,&nLMNY);
          fclose(shape);
```

```
        count=0;
        img_num=0;

        chdir("/export/home/pchan/crack/section");

        preprc2();

label3:
        printf("\nEnter Output File:");
        scanf("%s", output);
        getchar();

        if((output_file=fopen(output, "r"))!=NULL)
        {
                printf("\nFile NOT empty!\n Type '1' to APPEND New Result to it!\n Type '2' to OVERWRITE it!  ");
                scanf("%d", &overWriteFlag);
                switch(overWriteFlag)
                {
                case 1:
                        fclose(output_file);
                        output_file=fopen(output, "ab");
                        break;
                case 2:
                        fclose(output_file);
                        output_file=fopen(output, "wb");
                        break;
                }
        }
        else
        {
        fclose(output_file);
        output_file=fopen(output, "wb");
        }

        fprintf( output_file, "\nMission Manager File read is : %s", input );
        fprintf( output_file, "\n           Output File is : %s", output );
        input_file=fopen("tc_dmi.dat", "rb");
        section_tc=fopen("sec_tc.dat", "rb");

label5:
        printf("\nSelect VCR PLAY speed 1, 2, 5, 8 or 10 : ");
        scanf("%d", &speed);
        if(speed!=1 & speed!=2 & speed!=5 & speed!=8 & speed!=10)
        {
                printf("\n This VCR speed has not been implemented. Try again.\n\n");
                goto label5;
        }

        fprintf( output_file, "\nVCR PLAY speed is : %d", speed );
        printf("\nType '1' to choose short Output File: ");
        scanf("%d", &shortFileFlag);
        getchar();
```

```
label4:     if (fscanf(section_tc,"%d %d %f %d\n",&bgn_tc, &end_tc, &SecLen, &section_no)==EOF){
        printf("Reached EOF\n");
                    write_to_file(output_file, nintact ,nlong, ntrans, nalli,ninter, nspall, SecLen, section_no, speed);
        my_exit(input_file, output_file);
    }
            BEGINTC=bgn_tc;
            ENDTC=end_tc;
            printf( "\nSECTION %3d :\nBegin timecode is %08u\t  End timecode is %08u\n", section_no, BEGINTC, ENDTC );
            fprintf( output_file, "\nSECTION %3d :\nBegin timecode is %08u\t  End timecode is %08u\n", section_no, BEGINTC, ENDTC );

            if (fscanf(input_file,"%d %f\n",&file_time, &file_dmi)==EOF){
                    printf("Reached EOF\n");
                    write_to_file(output_file, nintact ,nlong, ntrans, nalli,ninter, nspall, SecLen, section_no, speed);
                    my_exit(input_file, output_file);
            }
            behind_ptr_time=file_time;
            behind_ptr_dmi=file_dmi;
            if (fscanf(input_file,"%d %f\n",&file_time, &file_dmi)==EOF){
                    printf("Reached EOF\n");
                    write_to_file(output_file, nintact ,nlong, ntrans, nalli,ninter, nspall, SecLen, section_no, speed);
                    my_exit(input_file, output_file);
            }
            ahead_ptr_time=file_time;
            ahead_ptr_dmi=file_dmi;

            while (compare_time(file_time, BEGINTC)!=2){
                    if (fscanf(input_file,"%d %f\n",&file_time, &file_dmi)==EOF){
                            printf("Reached EOF\n");
                            write_to_file(output_file, nintact ,nlong, ntrans, nalli,ninter, nspall, SecLen, section_no, speed);
                            my_exit(input_file, output_file);
                    }
                    behind_ptr_time=ahead_ptr_time;
                    behind_ptr_dmi=ahead_ptr_dmi;
                    ahead_ptr_time=file_time;
                    ahead_ptr_dmi=file_dmi;
            }


            cue_up(BEGINTC);
            state=status()&0x01;
            while(state==0x0) state=status()&0x01;

            switch(speed)
            {
            case 1:
                    slow_play1();
                    break;
            case 2:
                    slow_play2();
                    break;
            case 5:
                slow_play5();
                break;
case 8:
                slow_play8();
                break;
case 10:
                slow_play10();
                break;
}

tape_time=get_ltc();
tape_time=get_ltc();

stats[0]=120.0;
stats[1]=30.0;


for(;;)
{
                tape_time=get_ltc();

                snap(B1);

                ipa_snap(0,VDB,BLW);



                i=i_crc(img_num,LMNX,LMNY,nLMNX,nLMNY, stats);
                newdmi=interpolate(behind_ptr_time,ahead_ptr_time,tape_time,behind_ptr_dmi,ahead_ptr_dmi);
                if (compare_time(tape_time, ENDTC)==2)
                {
                                printf("Reached END\n");
                                write_to_file(output_file, nintact ,nlong, ntrans, nalli,ninter, nspall, SecLen, section_no, speed);
                                SecLen=nintact=ntrans=nalli=nlong=ninter=nspall=0;
                                fclose(output_file);
                                /* stop(); */
                                output_file=fopen(output, "ab");

                                goto label4;
                }

                if ( shortFileFlag == 1 )
                {
                                switch(i){
                                case 1:
                                                nintact++;
                                                class="Intact";
                                                break;
                                case 2:
                                                nlong++;
                                                class="Longitudinal";
                                                break;
                case 3:
                                ntrans++;
                                class="Transverse";
                                break;
                case 4:
                                nalli++;
                                class="Alligator";
                                break;
                case 5:
                                ninter++;
                                class="Intersect";
                                break;
                case 6:
                                nspall++;
                                class="Spalled Crack";
                                break;
                }
}
else
                switch(i){
                case 1:
                                nintact++;
                                class="Intact";
                                break;
                case 2:
                                nlong++;
                                fprintf(output_file, "%.4f %d %s\n", newdmi, tape_time, "1");
                                class="Longitudinal";
                                break;
                case 3:
                                ntrans++;
                                fprintf(output_file, "%.4f %d %s\n", newdmi, tape_time, "2");
                                class="Transverse";
                                break;
                case 4:
                                nalli++;
                                fprintf(output_file, "%.4f %d %s\n", newdmi, tape_time, "3");
                                class="Alligator";
                                break;
                case 5:
                                ninter++;
                                class="Intersect";
                                break;
                case 6:
                                nspall++;
                                fprintf(output_file, "%.4f %d %s\n", newdmi, tape_time, "4");
                                class="Spalled Crack";
                                break;
                }

printf("Distress type %d %s at dmi %.4f for time %d \n\n", i, class, newdmi, tape_time);
while (compare_time(ahead_ptr_time, tape_time)!=2)
```

```
                {
                        if (fscanf(input_file,"%d %f\n",&file_time, &file_dmi)==EOF)
                        {
                                printf("Reached EOF\n");
                                write_to_file(output_file, nintact ,nlong, ntrans, nalli,ninter, nspall, SecLen, section_no, speed);
                                my_exit(input_file, output_file);
                        }
                        behind_ptr_time=ahead_ptr_time;
                        behind_ptr_dmi=ahead_ptr_dmi;
                        ahead_ptr_time=file_time;
                        ahead_ptr_dmi=file_dmi;
                }
        }

}

void main()
{
        int ProcessType;

        printf( "\nTO STOP THE PROGRAM AT ANY TIME, PRESS 'CTRL C'\n" );

        printf( "\nType '1' to process CONSECUTIVE sections." );
        printf( "\nType '2' to process in INTERACTIVE MODE.\n" );



        scanf( "%d", &ProcessType );

        switch( ProcessType )
        {
        case 1:
                printf("\nMission Manager filename");
                printf("\noutput result filename");
                printf("\nVCR speed");
                printf("\nshortReportFlag");
                printf("\nThe above items must be read in this order arranged in the 'config.txt' file\n\n");
                main1();
                break;
        case 2:
                main2();
                break;
        }
        exit(0);
}
```

APPENDIX B
CRACK DETECTION ALGORITHMS AND RULES


Initialization (Finding Parameters)
First Level Elimination
Second Level Elimination
Third Level Elimination
Is it Intact? remainblocks<20
Yes
Intact
No
Segmentation (Projection Histogram)
Non-Crack Detection (Eliminate Abnormal Objects)
Is it Intact? (If All Objects are Non-Crack)
Yes
or
Oil-Stain
Shadow
Intact
No
Spalled_or_trans (Determine Spalled or Transverse)
Spalled Crack
Transverse

Overview Flowchart of the CRC Algorithm

ELIMINATION ALGORITHMS

1. First Level --- using 16X16 pixels subimage average, global average, subimage variance, and horizontal projection histogram.

Rules:

a. maxim < finave

b. (rowave < finave) and (((maxim - rowave) < thre)

c. ((maxim - rowave) < thre/2) and ((maxim - finave) < thre)

d. rowvar < thre/2 + 3

Any of the above cases => intact

2. Second Level --- using the maximum fft spectrum to rank subimages.

Rules:

a. fftspec > thre1 -- rating is 3

b. fftspec > thre2 -- rating is 2

c. fftspec > thre3 -- rating is 1

d. fftspec > thre4 -- rating is 0 => eliminated

Cases a. , b. and c. -- thresholds.

Note that the higher the rating, the higher the "spallness".

3. Third Level --- decompose each subimage into four parts and perform further elimination.

Rules:

a. sublocave < finave

b. sublocave < rowave

Any of the two cases => intact

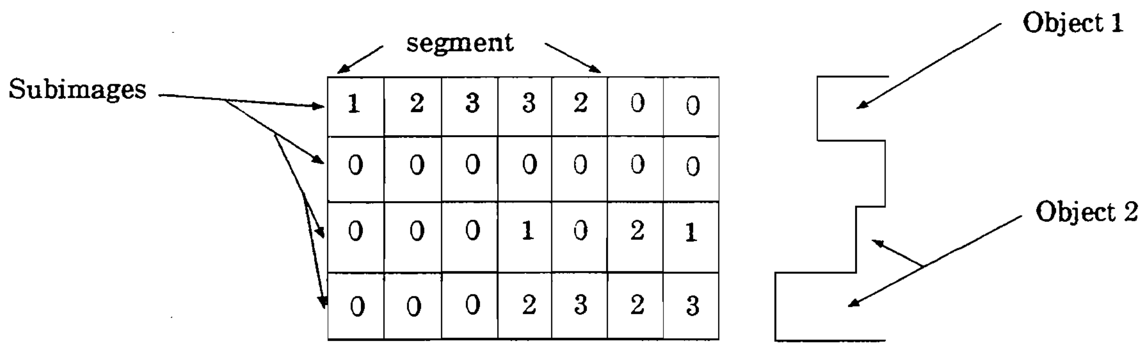
sublocave -- average of each part in a subimage.

Segmentation Algorithm

Based on the "spallness" ratings:

0 -- eliminated subimage
1 -- lightly spalled (or transverse) area
2 -- medium spalled area
3 -- heavy spalled area

We have a spallness map:



1	2	3	3	2	0	0
0	0	0	0	0	0	0
0	0	0	1	0	2	1
0	0	0	2	3	2	3

Non-Crack Detection

Rule:

a. ((asegwidth > thre 1) or (maxseg >= thre2)) => crack

asegwidth -- average segment width in each object

maxseg -- maximum segment width

thre, thre2 -- thresholds

Spalled_or_Transvers

Based on the strength of each object:

High count on each object -- spalled

Low count on each object -- transverse

Technical Report Documentation Page

1. Report No. TX-96/1965-1F	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle USER'S MANUAL FOR A PAVEMENT VIDEO IMAGE PROCESSING SYSTEM		5. Report Date November 1995
		6. Performing Organization Code revised : March 1996
7. Author(s) Chun-Lok Lau		8. Performing Organization Report No.
9. Performing Organization Name and Address Texas Transportation Institute The Texas A&M University System College Station, Texas 77843-3135		10. Work Unit No. (TRAIS) 11. Contract or Grant No.
12. Sponsoring Agency Name and Address Texas Department of Transportation Research and Technology Transfer Office P. O. Box 5080 Austin, Texas 78763-5080		13. Type of Report and Period Covered Final Report: September 1994 - August 1995
		14. Sponsoring Agency Code

15. Supplementary Notes
Research performed in cooperation with the Texas Department of Transportation.
Research Study Title: Implementation of Automatic Surface Distress Data Collection Procedures for PMIS

16. Abstract

This report provides a user's manual for the automated video distress identification system developed by TTI for the Texas Department of Transportation.

17. Key Words Pavement Distress, Video, Automated Processing, Image Processing	18. Distribution Statement No Restrictions. This document is available to the public through NTIS: National Technical Information Service 5285 Port Royal Road Springfield, Virginia 22161

19. Security Classif.(of this report) Unclassified	20. Security Classif.(of this page) Unclassified	21. No. of Pages 66	22. Price

Form DOT F 1700.7 (8-72) Reproduction of completed page authorized