DEVELOPMENT OF A PC VERSION FOR
AXISYMMETRIC DROP SHAPE ANALYSIS (ADSA)

By

Mina Hoorfar

A thesis submitted in conformity with the requirements
For the Degree of Master of Applied Science
Graduate Department of Mechanical and Industrial Engineering
University of Toronto

© Copyright by Mina Hoorfar 2001
The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author’s permission.

0-612-58839-4
To my husband, Homayoun, for his continuous love and support
DEVELOPMENT OF A PC VERSION FOR

AXISYMMETRIC DROP SHAPE ANALYSIS (ADSA)

Mina Hoorfar

M.A.Sc. Thesis, Department of Mechanical and Industrial Engineering.
University of Toronto, 2001

ABSTRACT

Many areas of research in surface phenomena require accurate measurements of interfacial properties. Axisymmetric drop shape analysis (ADSA) has been applied to measure contact angles and liquid-fluid interfacial tensions by fitting the Laplace equation of capillarity to the shape of the sessile and pendant drops. However, ADSA originally developed for a UNIX platform requires special training of personnel for working with the UNIX system, a fact which has limited wider use of ADSA. In addition, lacking peripheral software for the UNIX systems limits the effectiveness and usefulness of the ADSA program. This research develops a PC version of ADSA, a user-friendly program employing standard software controls. Additional features include: Microsoft Foundation Class (MFC), Object Linking and Embedding (OLE), and Multi-Document Interface (MDI), which are implemented using Visual C++, a highly enhanced programming environment. Finally, several experiments were conducted to verify the performance and accuracy of the ADSA PC-version.
ACKNOWLEDGEMENTS

I would like to thank my supervisor Professor A. W. Neumann for his supervision, support, guidance, and precious advice.

I would like to thank my husband for his support and patience.

I would like to thank my family, especially my parents, for their encouragement.

I would like to thank Professor M. L. Hair and Professor D. Li for their helpful suggestions and guidance.

I would also like to thank my laboratory colleagues, especially Zdenka Policova and Catherine Lam, for their invaluable help and friendship.
# TABLE OF CONTENTS

ABSTRACT .................................................................................................................. i

ACKNOWLEDGEMENTS ............................................................................................... ii

TABLE OF CONTENTS ................................................................................................... iii

LIST OF SYMBOLS ......................................................................................................... vi

LIST OF ACRONYMS ..................................................................................................... vii

LIST OF TABLES .............................................................................................................. viii

LIST OF FIGURES .......................................................................................................... ix

CHAPTER 1 ...................................................................................................................... 1

1.1 Introduction ............................................................................................................. 1

1.2 Motivation ............................................................................................................... 6

CHAPTER 2 PC-VERSION OF ADSA PROGRAM ............................................................ 8

2.1 Requirements .......................................................................................................... 8

  2.1.1 Experimental Setup ......................................................................................... 10

    2.1.1.1 Pendant Drop Configuration ................................................................. 11

    2.1.1.2 Sessile Drop Configuration .................................................................... 11

    2.1.1.3 CCD Camera, Microscope, and Vibration-Free Table.......................... 12

  2.1.2 Frame Grabber Board ....................................................................................... 12

  2.1.3 Host Computer ................................................................................................. 13

2.2 Feature of ADSA PC-Version Software .................................................................. 14

  2.2.1 Graphical User Interface (GUI) ...................................................................... 15

  2.2.2 Microsoft Foundation Classes Based on Object Oriented Database .......... 17

  2.2.3 Multiple Document Interface (MDI) ............................................................. 18

  2.2.4 Object Linking and Embedding (OLE) ........................................................ 19
5.2 Pendant Drop Experiment ........................................................................................................... 66
  5.1.1 Materials ............................................................................................................................ 66
  5.1.2 Procedures ........................................................................................................................ 66
  5.2.3 Results and Discussion ....................................................................................................... 67
  5.3 Summary ............................................................................................................................... 68

CHAPTER 6 SUMMARY .................................................................................................................... 77
  6.1 Advantages of the ADSA PC-Version .................................................................................... 77
  6.2 Future Work ......................................................................................................................... 78

APPENDIX A AXISYMMETRIC DROP SHAPE ANALYSIS (ADSA) ............................................ 80
  A.1 Experimental Setup for Digital Image Processing ............................................................... 80
  A.2 Drop Profile Analysis to a Resolution of ±1 pixel .............................................................. 81
  A.3 Correction of Optical Distortion ......................................................................................... 82
  A.4 Drop Profile Analysis to Subpixel Resolution .................................................................... 82

APPENDIX B VIDEO SIGNALS AND INPUT CONTROLS OF THE FRAME GRABBER........... 86

APPENDIX C CODES OF IMPORTANT FUNCTIONS IN THE ADSA PC-VERSION 89
  C.1 Serialize() Function ............................................................................................................ 89
  C.2 OnImageDeviceid() Function ............................................................................................ 90
  C.3 OnImageAcquire() Function ............................................................................................. 91
  C.4 OnProperties() Function ................................................................................................... 94
  C.5 OnImageOpen() Function .................................................................................................. 96
  C.6 OnImageCloseDevice() Function ...................................................................................... 97
  C.7 OnRun() Function .............................................................................................................. 98

REFERENCES .............................................................................................................................. 99
### LIST OF SYMBOLS

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\theta$</td>
<td>Contact Angle [degree]</td>
</tr>
<tr>
<td>$\theta_a$</td>
<td>Advancing Contact Angle [degree]</td>
</tr>
<tr>
<td>$\theta_r$</td>
<td>Receding Contact Angle [degree]</td>
</tr>
<tr>
<td>$\gamma$</td>
<td>Surface Tension [mJ/m$^2$]</td>
</tr>
<tr>
<td>$\gamma_{lv}$</td>
<td>Liquid-Vapor Surface Tension [mJ/m$^2$]</td>
</tr>
<tr>
<td>$\gamma_{sl}$</td>
<td>Solid-Liquid Surface Tension [mJ/m$^2$]</td>
</tr>
<tr>
<td>$\gamma_{sv}$</td>
<td>Solid-Vapor Surface Tension [mJ/m$^2$]</td>
</tr>
<tr>
<td>$A$</td>
<td>Drop surface Area</td>
</tr>
<tr>
<td>$R$</td>
<td>Radius of Curvature [cm]</td>
</tr>
<tr>
<td>$V$</td>
<td>Drop Volume [cm$^3$]</td>
</tr>
<tr>
<td>$\Delta P$</td>
<td>Pressure Difference across the Air-Liquid Interface [mJ/m$^2$]</td>
</tr>
<tr>
<td>$R_1, R_2$</td>
<td>Principal Radii of Curvature of the Drop [m]</td>
</tr>
<tr>
<td>$\Omega$</td>
<td>Free Energy</td>
</tr>
<tr>
<td>$T$</td>
<td>Temperature</td>
</tr>
<tr>
<td>$\mu$</td>
<td>Chemical Potential</td>
</tr>
<tr>
<td>$U_i, i=1, 2, \ldots, N$</td>
<td>Points Selected from the Meridian of an Experimental Drop Profile</td>
</tr>
<tr>
<td>$U(s)$</td>
<td>Calculated Profile</td>
</tr>
<tr>
<td>$N$</td>
<td>Number of Points</td>
</tr>
<tr>
<td>$u_i$</td>
<td>Measured Profile Point</td>
</tr>
<tr>
<td>$u = u(s)$</td>
<td>Calculated Laplacian Curve</td>
</tr>
<tr>
<td>$d_i$</td>
<td>Minimum Distance between $u_i$ and $U_i$</td>
</tr>
<tr>
<td>$x, z$</td>
<td>Coordinate System of a Theoretical Drop Profile</td>
</tr>
<tr>
<td>$X, Z$</td>
<td>Coordinate System of an Experimental Drop Profile</td>
</tr>
<tr>
<td>$x_0, z_0$</td>
<td>Coordinate of the Origin of a Drop</td>
</tr>
<tr>
<td>$\alpha$</td>
<td>Angle of Vertical Misalignment</td>
</tr>
<tr>
<td>Acronym</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>ADSA</td>
<td>Axisymmetric Drop Shape Analysis</td>
</tr>
<tr>
<td>ADSA-P</td>
<td>Axisymmetric Drop Shape Analysis-Profile</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical User Interface</td>
</tr>
<tr>
<td>MFC</td>
<td>Microsoft Foundation Class</td>
</tr>
<tr>
<td>MDI</td>
<td>Multiple Document Interface</td>
</tr>
<tr>
<td>OLE</td>
<td>Object Linking and Embedding</td>
</tr>
<tr>
<td>OS</td>
<td>Operating Systems</td>
</tr>
<tr>
<td>PMMA</td>
<td>Poly (methyl methacrylate)</td>
</tr>
<tr>
<td>SDS</td>
<td>Sodium dodecyl sulfate</td>
</tr>
</tbody>
</table>
LIST OF TABLES

Table 4.1  Relation between Properties property sheet controls and CExper Class data members
Table 4.2  Relation between Setup property sheet controls and CFrmGrb Class data members
Table 4.3  Additional functions in CView class
Table 5.1  Contact angles obtained from five experiments using PC and UNIX systems
LIST OF FIGURES

Figure 1.1. Contact Angle and Interfacial Tensions
Figure 2.1 Flowchart of the Program
Figure 2.2 Schematic diagram of the experimental setup for analysis of sessile-drop and pendant-drop systems by ASDA PC-version
Figure 3.1 I/O Data menu
Figure 3.2 Window of the ADSA program
Figure 3.3.a Experiment Page of Pendant Drop Program
Figure 3.3.b Experiment Page of Sessile Drop Program
Figure 3.4 Image of a pendant drop and Cut-off Level
Figure 3.5 Calculation Page
Figure 3.6 Grid Page
Figure 3.7 Output Page
Figure 3.8 Source Page
Figure 3.9 Video Type Page
Figure 3.10 Sync Sentinel Page
Figure 3.11 Video Area Page
Figure 3.12 Frame Page
Figure 3.13 Spatial Relationship of Video and Frame Areas
Figure 3.14 Select Device dialog Box
Figure 3.15 Acquire dialog box with three types of acquisitions
Figure 3.16 Open Image Dialog Box
Figure 3.17 Display Dialog Box
Figure 4.1  Hierarchy chart of ColeServerDoc class
Figure 4.2  Hierarchy chart of CView class
Figure 4.3  Hierarchy chart of CDoc class
Figure 5.1  Experimental results of water on PMMA surface derived from ADSA PC-version
Figure 5.2  Experimental results of water on PMMA surface derived from ADSA UNIX version
Figure 5.3  Experimental results of water on PMMA surface derived from ADSA PC-version and ADSA UNIX version during the advancing period
Figure 5.4  Experimental results of n-Hexadecane derived from ADSA PC-version
Figure 5.5  Experimental results of n-Hexadecane derived from ADSA UNIX version
Figure 5.6  Experimental results of n-Hexadecane derived from ADSA PC-version and ADSA UNIX version
Figure 5.7  Experimental results of n-Undecane derived from ADSA PC-version
Figure 5.8  Experimental results of n-Undecane derived from ADSA UNIX version
Figure 5.9  Experimental results of n-Undecane derived from ADSA PC-version and ADSA UNIX version
Figure 5.10 Experimental results of SDS solution from ADSA PC-version
Figure 5.11 Experimental results of SDS solution derived from ADSA UNIX version
Figure A.1  Comparison between experimental points and a Laplacian curve. The curve $u=u(s)$ is a theoretical profile based on the Laplace equation of capillarity. While points $U_i$, $i=1,2,\ldots, N$, are points selected from the meridian of an experimental drop profile. The deviation of the $i^{th}$ point from the Laplacian curve, $d_i$, can be calculated. The purpose of objective function of ADSA is to minimize the sum of the square of the minimum distance, $d_i$. Since the coordinate systems of experimental profile and the predicted Laplacian curve do not necessarily coincide, their offset $(x_0, z_0)$ and rotation angle $(\alpha)$ must be considered. Both the offset $(x_0, z_0)$ and the rotation angle $(\alpha)$ are optimization parameters. Since the program does not require the coordinates of the drop apex as input, the drop can be measured from any convenient reference frame, and all measured points on a drop profile are equally important.
Figure A.2  Schematic diagram of the experimental setup for analysis of sessile-drop and pendant-drop systems by ASDA UNIX version

Figure B.1  Horizontal Video Signal

Figure B.2  Horizontal Input Control

Figure B.3  Vertical Video Signal

Figure B.4  Vertical Video Signal
1.1 Introduction

The significance of interfacial and capillarity phenomena in surface science has been increasingly recognized in recent years. These phenomena occur whenever a liquid is in contact with another fluid or solid. The most common examples are menisci and drops formed by liquids in air or in another liquid and thin films. Many processes such as lubrication, adhesion, separation, etc. depend on interfacial behavior of the materials involved.

As early as the fifteenth century, many scientists tried to explain the phenomena like the capillary rise, but it was not until the early nineteenth century that the theory of capillarity was simultaneous introduced by Young [1] and Laplace [2]. The latter considered the pressure difference across a curved surface as a consequence of the curvature and derived the Laplace equation given as:

$$\gamma \left( \frac{1}{R_1} + \frac{1}{R_2} \right) = \Delta P$$  \hspace{1cm} (1.1)

where $R_1$ and $R_2$ are the two principal radii of curvature. As a result, the Laplace equation relates the change in pressure $\Delta P$ and the surface tension $\gamma$ (i.e., a measurable force existing in all liquid surfaces). The surface tension arises directly from an imbalance of the cohesive forces that exist between the constituting molecules of the liquid. Physically, $\gamma$ represents the reversible work required to create a unit area of surface, and it is defined as
\[ \gamma = \left( \frac{\partial \Omega^{(A)}}{\partial A} \right)_{T, \mu} \]  

(1.2)

where \( \Omega, T, \) and \( \mu \) are free energy, temperature, and chemical potential, respectively.

Based on the same theory, Young's equation is also derived as:

\[ \gamma_{sv} - \gamma_{sl} = \gamma_{lv} \cos \theta \]  

(1.3)

where the subscripts \( lv, sv, sl \) refer to liquid-vapor, solid-vapor and solid-liquid interfaces, respectively, and \( \theta \) is the contact angle formed between the liquid and the solid surface as illustrated in Figure 1.1. It is noted that the contact angle is a common and useful measure of wettability, and it is used in studies of adhesion and floatation.

![Figure 1.1 Contact Angle and Interfacial Tensions](image)

Surface tension and contact angle are two of the most important interfacial properties, and their measurements are essential to the experimental research in surface thermodynamics. Many techniques have been developed to measure the interfacial properties. For measuring the surface tension, the Wilhelmy slide has been widely used due to its good precision and relative simplicity. However, this method relies on perfect wetting (zero contact angle) with the withdrawing surface, a condition that cannot always be ensured. The contact angle is often measured from a drop of liquid resting on a flat surface of the solid, as illustrated in Figure 1.1.
In this technique, the angle is measured by aligning the tangent with the drop profile at the point of contact. The measurement can be carried out either using a telescope with a goniometer scale or a photograph of the drop. This technique is very subjective and depends on the experience of the operator. The precision of this method is estimated about ±2°. An alternative method is the capillary rise \([3,4]\) at a vertical plane. In this method, the solid surface is brought into contact with the liquid to measure the capillary rise at the vertical plane. In addition to the capillary rise, the liquid surface tension, density difference, and gravitational acceleration are required for determining the contact angle. The precision of this technique is about ±0.1°, an order of magnitude better than the previous method; however, the disadvantage of this method is that large liquid volumes are required.

Drop shape analysis methods have been developed to determine the surface tensions of liquids from measuring the shape of the sessile or pendant drops. In principle, these methods fit the Laplace equation of the capillarity (Eq. 1.1) to the measured shape of the drop. For sessile drops, once the surface tension and the principal radii of curvature of the drop have been entered as inputs, the Laplace equation can be readily integrated to obtain the contact angle.

It was found that sessile drops are very sensitive to the quality of the solid surfaces. A slightly imperfect solid surface can lead to contortions in the three-phase line that render the drop non-axisymmetric. Pendant drops are preferable for higher precision in the interfacial tension because of the axial symmetry of the drop. Drop shape methods have the advantage that only small quantities of the liquid are required, and they can be used in many difficult experimental conditions such as studies of temperature or pressure dependence of liquid-fluid interfacial tensions.
Bashforth and Adams [5] developed a drop shape method by generating sessile drop profiles for different values of surface tension and radius of curvature at the apex. This was long before digital computers appeared and their work required tremendous labor. Hence, the task of determining interfacial tension and contact angle from the actual profile became a matter of interpolation from their tables, which contained the solutions of the differential equations describing the profile.

Hartland and Hartley [6] collected numerous solutions for determining the interfacial tensions of axisymmetric fluid-liquid interfaces of different shapes. Using a FORTRAN computer program to integrate the appropriate form of the Laplace equation, they presented the results in tabulated form. However, the major source of errors in their method is related to data acquisition. The description of the whole surface of the drop is reduced to the measurement of a few preselected critical points, which are compatible with the tables used. Since these points are related to special characteristics (e.g., inflection points on the surface), they must be determined with high accuracy.

Maze and Burnet [7,8] have developed a more satisfactory scheme for the determination of interfacial tension from the shape of a sessile drop. They developed a numerical nonlinear regression procedure to generate a theoretical drop shape, which is used to fit a number of points selected from an experimental drop profile. In order to generate a theoretical drop shape, initial estimates of the drop shape and size are required. The initial estimates have to be reasonable; otherwise, the theoretical drop profile will not converge to the experimental drop profile. To this end, initial estimates are usually obtained from the tables of Bashforth and Adams [5]. Despite the progress in strategy, there are several deficiencies in this algorithm. For example, the error function, i.e., the difference between the theoretical drop profile and the experimental drop
profile, is computed by summing the squares of the horizontal distance between the theoretical drop profile and the experimental profile. This computation is not particularly suitable for sessile drops strongly influenced by the gravity forces (large drop size with low surface tension). Such shapes tend to flatten near the apex; hence, any data near the apex will cause a large error.

Rotenberg [9,10] developed a more powerful technique, the axisymmetric drop shape analysis (ADSA), which fits the measured profile to a Laplacian curve using a nonlinear procedure. In the ADSA method, the objective function, which is used to evaluate the discrepancy between the theoretical Laplacian curve and the actual profile, is the sum of the squares of the normal distances between the measured points and the calculated curve. In addition, the location of the apex of the drop is assumed to be unknown and the coordinates of the origin are considered as independent variables of the objective function. Thus, the drop shape can be measured from any convenient frame. ADSA uses the Newton-Raphson method in conjunction with incremental loading to minimize the objective function. This numerical procedure unifies both the methods of the sessile and pendant drops. No table is required, nor is there any restriction on its applicability. However, this method has failed for large flat sessile drops, apparently because of round-off numerical errors.

Cheng et al. [11-13] evaluated the performance of ADSA for both pendant and sessile configurations using synthetic drops. The randomness choice of the data as input to ADSA was evaluated. The data at five difference locations of the profile were individually perturbed to test the influence of each location of the results. It was found that data points near the neck of a pendant drop or near the liquid-solid interface for a sessile drop have more impact on the results than points from other locations. This first generation of ADSA was found to give very accurate results except for very large and flat sessile drops, where the program failed. In addition, it was
difficult to achieve perfect alignment of the camera with a plumb line; there are errors associated with the coordinates of the plumb lines defined manually on the screen of the computer using a mouse.

Finally, del Río [14,15] rewrote ADSA algorithms by implementing the curvature at the apex rather than the radius of curvature and the angle of vertical alignment as optimization parameters. He also replaced the main algorithms by more efficient ones.

1.2 Motivation

The axisymmetric drop shape analysis (ADSA) program is a powerful technique to determine the interfacial tensions of liquid interfaces and the contact angle formed at liquid-solid interfaces; however, it was originally developed for a UNIX platform. This requires special training of personnel for working with the UNIX system, a fact which has limited wider use of ADSA. Therefore, it is the purpose of this thesis to develop a user-friendly program in the PC environment to alleviate the problems associated with the UNIX system. A PC-version of ADSA, would have the following advantages:

- Accessories required for experimental setup (e.g., a frame grabber interface card) more readily available for a PC than those for a UNIX system;

- Laboratory personnel are more familiar with PC computers, which can also be employed for other jobs;

- Peripheral software programs are well developed and available commercially for a PC computer that are not necessarily available for a UNIX system;
Another problem with the ADSA program is the large number of inputs required for the computation procedure, possibly being frustrating for a user. Therefore, a graphical user interface was developed for the PC-version of the ADSA program that allows the user to input the experimental parameters, and to extract the results in a faster and more effective way. This program was developed as an object-oriented program based on the Microsoft Foundation Classes (MFC) using Visual C++ language, a highly enhanced programming environment. The program also uses the multi-document feature to display the results of several experiments simultaneously.

In addition, there are several aspects of the ADSA PC-version that make this program attractive:

- **User Friendly:** Using the Window applications decorated with different dialog boxes, one may input the data and extract the results in a convenient way.

- **Adjustable Rate and Type of Image Acquisition:** The program is able to adjust the type and the rate of the image acquisition based on frame grabber setup for inherently different experiments.

- **Accuracy:** The resolution of the image for a PC computer is higher than that for a UNIX system so more accurate results are expected from the PC computer.

- **Processing Speed:** Using a traditional processor (e.g., 500 MHz), a PC computer can carry out the computation much faster than a UNIX computer.

- **Enhancement:** Being an object-oriented application, the program can be potentially considered as groundwork for future work.
CHAPTER 2

PC-VERSION OF ADSA PROGRAM

The ADSA PC-version contains two parts: 1) an interface between the user and the ADSA program. 2) a toolbox for controlling frame grabber performance.

In the first part, the interface between user and ADSA program was primarily developed to take a large number of data as input to the ADSA program in a convenient way. In addition, the speed of extracting the results has been improved in the PC-version of the ADSA software.

The second part, the frame grabber performance controller, enables the user to choose a suitable image acquisition type. Furthermore, several features of the frame grabber including video and frame type, frame size, pixel clock (frequency), etc. are contained in this part of the program. The flowchart in Figure 2.1 shows a perspective of the program.

This chapter introduces requirements for performing a sessile drop or pendant drop experiment on a PC system, and some features of the PC-version of ADSA are explained.

2.1 Requirements

Requirements for a sessile drop or pendant drop experiment are divided into three parts: 1) Experimental Setups, 2) Frame Grabber, and 3) Host Computer.
Open the Device

Have the images been acquired before?

Yes

No

Sequential Single Acquisition

Multiple Acquisition

Single Acquisition

Physical Properties:
Density, Grid file, Calibration...

Graphical Properties:
Image File and Image Names

Input File
"pendant.in" or "sessile.in"

ADSA Program

Results:
Surface Tension,
Contact Angle,
Surface Area,
Volume, and
Radius of Curvature

Figure 2.1 Flowchart of the Program
2.1.1 Experimental Setup

The schematic of the experimental setup for a sessile drop and pendant drop are illustrated in Figure 2.2. As shown in this figure, the only difference between the sessile drop and pendant drop experiments is in the drop configurations.

The other components (e.g., CCD camera, and microscope) will be discussed later.

---

**Figure 2.2** Schematic diagram of the experimental setup for analysis of sessile-drop and pendant-drop systems by ASDA PC-version
2.1.1.1 Pendant Drop Configuration

A pendant drop is formed at the end of a Teflon capillary having an inside diameter of, e.g. 0.076 inches, and an outside diameter of, e.g. 0.1 inches. To keep the capillary vertical and straight, a metal guide tube (inside diameter: 0.106 inches, outside diameter: 0.1304 inches) is used as a sleeve for the capillary. The other end of the capillary is connected to a syringe coupling connecting to a stepper motor. The volume of the drop and the surface area of the drop can be changed by pumping liquid in and out from the capillary.

To minimize the evaporation and to isolate the drop from vibration due to air currents, a quartz glass cuvette (Hellma Limited Company) is used in liquid-air pendant drop experiments. A Teflon stopper is used to seal the quartz glass cuvette, which is placed into a temperature/pressure cell.

The volume of the pendant drop is adjusted by means of a motor-driven syringe. The syringe with the capacity 2.5 ml (Gastight®, Hamilton Co.) is connected to a stepper motor (Model 18705, Oriel Corp., USA) by an aluminum coupling. The available speed range is 0 to 500 steps per second, where each step represents a linear distance of \( \frac{1}{2} \) micrometer. The entire travel distance of the stepper motor is 20000 steps.

2.1.1.2 Sessile Drop Configuration

In the sessile drop experiment, the surface stage is leveled using a bubble level. A sample surface is carefully put on the stage such that the needle must pass through the hole in the center of the surface. Drops may be formed from above or below the solid surface. Depositing a drop directly from the top can result in inconsistent and uncertain contact angles due to drop
oscillations. This can be remedied by first depositing a small drop onto a small hole in the surface. The size of the drop is then increased by feeding liquids to the drop from below the surface by a motorized syringe [15]. The motor in the motorized syringe mechanism is set to a specific speed by adjusting the voltage from the voltage controller. Such mechanism pushing the syringe plunger leads to an increase in the drop volume and in the three-phase contact radius.

2.1.1.3 CCD Camera, Microscope, and Vibration-Free Table

It is noted that the following apparatus is used in both the pendant drop and the sessile drop constellation.

A microscope fitted with a CCD monochrome camera is used to acquire images. A heavily frosted diffuser is used in front of a light source to provide a uniformly lit background and to minimize the heat input to the drop during image acquisition. These images are acquired by a frame grabber board and stored in the host computer. The frame grabber board and host computer will be elaborated on in later sections. The entire set-up is mounted on a vibration-free table (Model 78443-20, Technical Manufacturing Crop., USA).

2.1.2 Frame Grabber Board

The image acquisition is carried out using a video card interface. DT3152 board, which is a high-accuracy, programmable, monochrome frame grabber board installed on one of the PCI slots. It is suitable for both image analysis and machine vision applications. The DT3152 accepts video signals in many different monochromes and variable-scan video formats, and digitizes the signal. The board can either store the digitized data to the host computer memory and hard disc or transfer the data to the computer display controller to display images in real time. It is noted
that this program can only support DT3152, DT3155, and DT3157 boards. Some key features of the DT3152 board are summarized as follows:

- Operating on the PCI local slot interface
- Acquiring images in the size of 640 pixels by 240 pixels in this version of the program
- Digitizing 8-bit monochrome video from any of four video input channels
- Synchronizing to any of four video inputs or to an external sync input
- Accepting separate horizontal and vertical sync inputs for variable scan devices
- Providing a 256×8-bit input look-up-table (ILUT)
- Providing a pixel clock, programmable both internally and externally

2.1.3 Host Computer

The host computer is used to store images acquired by the frame grabber board in the memory for computations and/or in the hard disc for future use. Since the frame grabber uses the host computer memory to store the data, the larger the memory of the host computer the more images can be acquired without interruption. For instance, 32 Mbytes of memory will be required to acquire 100 images without interruption if the size of each image is about 320 Kbytes. Thus, the memory requirement of the host computer directly depends on the type of the experiment that is carried out with this system. In other words, the faster and the longer the experiments the larger the memory provided for the host computer must be. Also, it is the responsibility of the user to prepare sufficient space on the hard disc for storing the images.
throughout the experiment though the program gives an error message when it cannot save any more images.

The host computer can be equipped with one or more frame grabbers; however, the user must specify the active frame grabber by opening the corresponding device via a dialog box in the software prepared for this purpose. In the dialog box, each device is associated with a name that is user-defined.

Another point that must be addressed is the color system of the host computer. Since the images supplied by the frame grabber are monochrome, the ADSA program has been developed to work with the 16-bit-color or the 256-color displaying systems. As a result, the host computer must be set to one of the two systems.

2.2 Feature of ADSA PC-Version Software

In this section some features of the ADSA PC-version are introduced. In addition to the advantages of the program already mentioned, these features make it versatile, user-friendly, accurate, and fast. The program was developed as a user-friendly program employing dialog boxes, property pages, radio buttons, etc. in the graphical user interface (GUI). Additional features include: 1) Multiple Document Interface (MDI); 2) Microsoft Foundation Classes (MFC); and 3) Object Linking and Embedding (OLE), which are implemented using Visual C++ language, a highly enhanced programming environment. In this section, each feature is explained in detail.
2.2.1 Graphical User Interface (GUI)

GUI in the ADSA PC-version is designed for future commercial applications and is based on standard software controls and features. The designs of the sessile drop and the pendant drop experiments are basically identical; however, there are some exceptions that will be elaborated in this section. Furthermore, the graphs and the details of GUI may be found in Chapter 3.

In addition to window's standard pop-up menus (i.e., File, Edit, View, Window, and Help) the program has I/O Data pop-up menu items. The I/O Data contain three sub menu items: Image, Properties, and Run that invoke frame grabber functions, experimental input property sheet, and ADSA program, respectively. The Image menu item contains seven submenus (i.e., Open Device, Close Device, Setup, Acquire, Open, Last Image, Next Image) for controlling the frame grabber performance.

The frame grabber is selected and opened from the Alias combo box in Select Device dialog box, which is opened after choosing Open Device submenu. During the installation procedure, a unique name (alias) must be entered to specify the board installed. After selecting the desired frame grabber, which is connected to the camera, a command is sent from the combo box to the computer to open and initiate the board for acquiring the images. After finishing the experiment the frame grabber must be closed by selecting Close Device submenu. Devices (frame grabbers) not explicitly closed may use system resources that make the latter unavailable until the system is restarted. Choosing Setup submenu a property sheet containing five pages, which include all of the information with respect to pixel clock, frame type, video type, and frame size for setting the frame grabber, is opened. After opening a device the program
automatically sets up the frame grabber with the proper frequency dependent values. Details may be found in Chapter 4. After Initialization procedures mentioned above, the frame grabber is ready to acquire images by choosing Acquire submenu. Three methods (i.e., Single, Sequential Single, and Multiple acquisition) have been developed to acquire images. The first method, Single Acquisition acquires one image, which is immediately displayed on the screen. In the Sequential Single method, single frames are sequentially acquired for a desired period of time and the resulting images are saved in separate files. Since the file saving speed is dominant in this procedure, this type of acquisition is used when high speed (frequency) is not necessary (i.e. no more than 3 images per second, i.e. an interval of 0.33). Finally, the Multiple Acquisition method acquires several frames in a user-defined duration, but saves them in a single file. In this method, up to 30 frames per second may be acquired; consequently, it is suitable for fast changes in surface tensions where many observations should be performed in a second. In addition, there is an option for separating the images in the Multiple Acquisition method. Open submenu is used to open an image file that has already been saved on the hard drive. After selecting the image filename the Display dialog box is opened. For the images acquired by the Single or the Sequential Single method, where each image file contains only one image, the counter in the dialog box is automatically set to 1. However, if the images have been acquired with the Multiple Acquisition method, to display the $n^{th}$ image the counter has to be set to $n$ (e.g., for the second frame of the image file the counter is set to 2).

Last Image and Next Image submenus may be used, respectively, to forward and to rewind the frame counter in image files acquired through the Multiple Acquisition method. For instance, while the fourth image was opened for displaying, the third and fifth images can be displayed using these submenus.
The *Properties* menu item is associated with experimental information such as drop shape, density, and gravity. This menu item invokes the *Properties* sheet including four pages.

The exceptions in sessile and pendant drop programs, addressed at the beginning of this section, exist in the first page, *Experiment*, however, some items are identical in both including Image Format, Drop Shape, Number of Images, Starting Y Coordinate for Profile Searching, and File of Images' Names. For sessile Drop experiments contact angle information is included in *Experiment* page, however, Cut-off Level and File of Rotation Coordinates are considered for pendant drop experiments. On the second page, *Calculation*, includes the information with respect to Cubic Spline Fit, Coordinate of Center of Image, Local Gravity, No. of Iterations of ADSA, No. of Points per Iteration, Drop Coordinates, and Density. In the next page, *Grid*, two choices (i.e., Scale from Grid and Own Scale) for scale have been considered. For the first choice peripheral information (i.e., Grid File, Data File Name of Four Boundary Points, Number of Horizontal Lines and Vertical Lines of Grid, and Distance between two Grid Lines) are required, otherwise the standard scale chosen is necessary. The last page, *Output*, contains four choices for the output file. After closing the property sheet by pressing the Ok push button, the required inputs are written in a text file named “sessile.in” or “pendant.in”, for sessile drop or pendant drop experiments, respectively.

Finally, by pressing *Run*, which connects the inputs to the ADSA program, the window of ADSA is opened and the name of input file is required.

### 2.2.2 Microsoft Foundation Classes Based on Object Oriented Database

The Microsoft Foundation Classes (MFC) is a set of predefined classes in Windows programming with visual C++. Written in C++, MFC provides most codes necessary for
managing windows, menus, and dialog boxes; performing basic input/output; storing collections of data objects and so on. It is an application framework for programming in Microsoft Windows. The classes represent an object-oriented approach to Window programming that encapsulates the Windows API (Application Program Interface). In this way, the process of writing a Windows program involves creating objects from MFC classes or objects of classes derived from MFC. The objects will incorporate member functions for communicating with Windows, processing Windows messages, and sending messages to each other. In addition to the new tasks, the derived classes inherit all of the data and the member functions of their parent classes. MFC shortens development time: makes codes more portable; provides tremendous support without reducing programming freedom and flexibility; and gives easy access to "hard to program" user interface elements and technologies, like ActiveX technology, Object Linking and Embedding features (OLE), and Internet programming. MFC makes it possible to program features like property sheets ("tab dialogs"), print preview, and floating and customizable toolbars. Finally, using the base classes the program can be executed with any version of Windows.

2.2.3 Multiple Document Interface (MDI)

Multiple document interface (MDI) applications can support more than one type of documents while the documents can be opened at the same time. However, single document interface (SDI) applications allow working on one document at a time. The MDI applications have a mainframe window within which multiple child frame windows, each containing a separate document, can be displayed.
The MDI allows creating an application that casts multiple forms into a single container form (template). The child frame windows (i.e., documents) are maintained in the main window that provides a workspace for all the child windows in the application.

2.2.4 Object Linking and Embedding (OLE)

Object Linking and Embedding, perhaps better known as OLE, is a mechanism which allows the program to handle data contained within other applications. In other words, embedded and linked objects are the result of transferring objects to the container document or creating new objects of a particular type in the document. An object from an external document can be entered to the container document in two ways: 1) Linked to the document, an external object is not stored, but the document has a reference to retrieve the object. 2) Embedded in the document, an external object is included in the document.

The Microsoft Object Linking and Embedding technology (OLE) makes objects created in a document in one application available to a document created in another application. OLE objects from an OLE server, for example a Microsoft Excel spreadsheet, can be linked or embedded in our program document or vice versa. A linked OLE object displays the object's data in a document, but the data remains stored in its original file. In contrast, an embedded OLE object is a copy of the information created in the OLE server that is stored in the container document. Thus, changes made to the original file are not reflected in the document, and likewise, changes made to the embedded object are not saved back.

The MFC library defines numerous classes for supporting OLE functionality, including classes that implement the standard OLE dialog boxes. AppWizard also provides support for the automated creation of OLE container and OLE server applications.
CHAPTER 3

GRAPHICAL USER INTERFACE

Graphical User Interface (GUI) of this program is developed to relate different parts of the ADSA application including frame grabber control functions, the ADSA program, and data input-output services. The GUI of this program consists of several menus, property sheets and dialog boxes, which are specially designed to reduce the complexity of using the ADSA program for determining the interfacial properties in surface thermodynamics experiments.

This chapter is subdivided into three sections. In section 3.1 two types of menus used in this program are explained. First, windows standard menus include File, Edit, View, Window, and Help menus. Second, the I/O Data menu is additionally defined to invoke the main functions of this program. Elaborated in section 3.2, the property sheets of the program encapsulate experimental inputs and frame grabber configuration data. Finally, Open Device, Open Image, and Acquire dialog boxes are introduced in section 3.3.

3.1 Menus

The two types of menus are as follows:

- Windows standard menus: Using Microsoft Foundation Classes (MFC), the menus perform every feature of the window application. For instance in the File menu Open and Save reads input data from and writes the result to the hard disc, respectively. It is noted that two types of databases are involved in this application. First, input-output
data held in \textit{CExper} class is read and written by \textit{Open} and \textit{Save} in \textit{File}. Second, acquired and saved by frame grabber functions, image data is read by \textit{Open Image} in \textit{I/O Data}.

- \textit{I/O Data} menu: Shown in Figure 3.1, \textit{I/O Data} is used to select \textit{Image}, \textit{Properties} and \textit{Run} that invoke frame grabber functions, experiment input property pages, and ADSA program, respectively.

![I/O Data menu](image)

**Figure 3.1 I/O Data menu**

\textit{Image}, which pertains to frame grabber operations, includes seven submenu items elaborated below.

- \textit{Open Device}: Prior to image acquisition, the frame grabber device must be opened from the \textit{Select Device} dialog box, invoked by selecting the \textit{Open Device} submenu. The details of this dialog box may be found in section 3.3.

- \textit{Close Device}: After finishing the experiment, the device must be closed by selecting the \textit{Close Device} submenu. Devices that are not explicitly closed may use system resources that make the latter unavailable until the system is restarted.
• **Setup**: Setup invokes a property sheet containing all of the information for setting the frame grabber. After opening a device the program sets up the frame grabber with proper values automatically. This property sheet is elaborated in section 3.2.

• **Acquire**: The Acquire dialog box contains three modes (viz., Single, Sequential Single, and Multiple acquisition) for acquiring and saving the images. In this stage an appropriate memory allocation strategy is selected to acquire and to maintain the images for the duration of the experiment. The acquisition modes and corresponding allocation strategies are further explained in section 3.3.

• **Open Image**: This submenu item is used to open an image file that has already been saved on the hard drive. The details of the Open Image dialog box are explained in section 3.3.

• **Last Image** and **Next Image**: These submenu items may, respectively, be used to forward and to rewind the frame counter in image files while acquiring with the Multiple Acquisition method. Repeatedly selecting these submenu items, one can browse through the memory allocated for the images.

The Properties menu item invokes a property sheet associated with experimental information (e.g., drop shape, density, and gravity). This property sheet composed of Experiment, Calculation, Grid, and Output property pages is elaborated in section 3.2.

**Run** launches the ADSA program that uses the images to determine contact angle, surface tension, surface area, volume, and radius of curvature. The ADSA program in turn asks for an input file name to load the input data previously provided by the GUI, see Figure 3.2. The results of the ADSA program are eventually saved in a user-defined file.
3.2 Property Sheets

A property sheet, also known as a tab dialog box, is a dialog box that contains a series of property pages. A property page is used to display an interface for modifying a group of properties. In other words, the property sheet is applied to encapsulate a group of related properties in several property pages. Microsoft Foundation Classes (MFC) includes CPropertySheet to define a property sheet, and the class CpropertyPage to define individual tabbed pages within a property sheet. Each property page will use controls such as edit boxes, list boxes or radio buttons for the setting of individual property values. The GUI of this program includes two property sheets, Properties and Setup, which are elaborated in this section.

3.2.1 Properties Property Sheet

This property sheet collects the experimental information such as drop shape, density, and gravity in four property pages (viz., Experiment, Calculation, Grid, and Output).

In each page of this property sheet, a group of related properties are expressed with standard control such as radio buttons, edit boxes, and check boxes. These pages are as follows:
3.2.1.1 *Experiment* Property Page

The differences between the pendant drop and the sessile drop programs appear in the *Experiment* page. However, as shown in Figure 3.3.a and 3.3.b, the following properties are identical in both programs:

![Figure 3.3.a Experiment Page of Pendant Drop Program](image)

**Figure 3.3.a** *Experiment* Page of Pendant Drop Program
Figure 3.3.b *Experiment* Page of Sessile Drop Program

- **Image Format:** There are two formats, IMG and TIFF. In the present version, all images have to be saved in IMG format.

- **Drop Shape:** There are two choices, Float Up or Hang Down, which depends on the density difference between two liquids.

- **Number of Images:** The number of the images for processing is entered in this edit box.

- **Starting Y Coordinate for Profile Searching:** To start searching for the profile a coordinate from the left is entered in this edit box. Its minimum value is 6, prescribed by ADSA.
- File names of Images: The last combo box on this page is the name of the file containing the name of the images. After acquiring the images, the program saves their names in a text file named like their image file with "txt" extension. If no name is entered, the last opened image will be processed. If the user selects the No push button the program asks that the name of the image(s) in the input file be typed in.

In addition to the above information, the following properties are specified for the pendant drop experiment.

- Cut-off Level: The cut-off level, which is the level at which processing ceased, must be assigned. Two points are chosen on the drop and near the tube, by double clicking the right mouse button on both sides of drop. Then the program will choose the lower one and enter its coordinate in the Cut-off Level edit box (Figure 3.4). The ADSA program will stop processing near this coordinate. It is noted that the direction of x is from top to bottom and y is from left to right.

![Image of a pendant drop and Cut-off Level](image.jpg)

Figure 3.4 Image of a pendant drop and Cut-off Level
- File of Rotation Coordinates (the two points chosen on the vertical line): After aligning the camera with a vertical line, an image must be taken for fixing two points on the vertical line. By clicking the right mouse button on the vertical line, the program saves those points in a file named exactly as its image file with "o" extension. For instance, if the image of the vertical line is named "vertline" the file name of rotation coordinates listed in this combo box will be "vertline.o".

As it was mentioned before, the differences between the pendant drop and the sessile drop programs appear in the Experiment page. Instead of Cut-off Level and File of Rotation Coordinates, which are required for the pendant drop experiment, some qualitative information about the contact angle is necessary for the sessile drop experiment. Therefore, the following choices of the contact angle were considered in the Experiment page for sessile drop experiment:

Contact Angle: The ADSA program requires some qualitative information about the contact angle for determining the interfacial properties in the sessile drop experiment. A group of radio buttons has been designed to represent the different choices of the contact angle. If the contact angle of the sessile drop is certainly less than or certainly bigger than 90°, the first or second radio buttons is selected, respectively. If there is no stipulation for contact angle, the third radio button (viz., Don't Know) is selected. By selecting the third radio button, an edit box is enabled for entering the angle difference, which is a necessary criterion for the program to stop searching the edge. By default, the program assumes that the drop profile is a continuous curve. Thus, a significant change in the slope of the curve values means that the direction of the drop profile has changed. If the change in the slope value at a point is greater than the value of the angle difference, the program will stop searching the profile and will take the point as an edge point. However, for the fourth and
fifth radio buttons, i.e., stop searching the profile and contact angles near 90°, two points of the drop must be selected. By clicking the right mouse button on both sides of the drop, the coordinates of these points are entered in the edit boxes enabled by selecting either the fourth or the fifth radio button.

It is noted that each selected point in the above sections can be canceled and reselected by Undo menu item in Edit menu.

3.2.1.2 Calculation Property Page

Shown in Figure 3.5, the second page, Calculation, contains the following information:

- **Cubic Spline Fit**: To determine the drop profile coordinates to sub-pixel resolution, the ADSA program uses a natural cubic spline fitting method. A spline is a flexible strip held by weights so that it passes through each of the given points according to the laws of beam flexure. The cubic spline fitting method is often used to fit a set of third order polynomial equations to the points of each interval. There are four choices for cubic spline fitting in this program. The second one (One-Dimensional) is used to find the drop profile coordinates with sub-pixel resolution, and the mid-point of each natural spline fit is chosen as the drop profile coordinate. However, to establish the optimum position of the drop profile coordinates, the two-dimensional spline fitting method may be adopted to shift the drop profile coordinate perpendicular to the drop interface in either direction. The details of ADSA program may be found in Appendix A.

- **Drop Includes the Center of Image**: If the drop includes the center of the image this box has to be checked, otherwise the coordinate of a point inside the drop must be
entered. By double clicking the left mouse button, the program enters this coordinate in the edit boxes.

- Local Gravity: This edit box contains the local gravity and for the pendant drop the value must be entered as negative. This edit box was initialized by "-980.43" which is the gravity in Toronto.

![Diagram of ADSA interface]

**Figure 3.5 Calculation Page**

- Number of Iterations of ADSA: This edit box shows the number of iterations of the ADSA program. The number of iterations was initialized as 10 iterations.

- Points/Iteration: This edit box is for the number of points on the profile per iteration and 20 points were selected, as this has been widely found to be convenient.

- Show Drop Coordinate: If this check box is checked the profile coordinates of the drop will be shown.
Density: The density of the liquid must be entered in this edit box.

3.2.1.3 Grid Property Page

Most cameras and microscopic lenses produce slightly distorted images, and this distortion can cause major errors in the final results, particularly in the interfacial tension. To correct the optical distortion at different magnifications, three different metallic grid patterns (i.e., $0.05 \times 0.05 \, cm^2$, $0.025 \times 0.025 \, cm^2$, and $0.01 \times 0.01 \, cm^2$) used. Each grid pattern is formed by many small and identical linear squares. The grid is used for both correction of the optical distortion and calibration. During each experiment, an image of the grid pattern on an optical glass is acquired and saved in a file. The information about the grid file encapsulated in the Grid page (Figure 3.6) is as follows:

- **Scale**: There are two choices for scale (i.e., Scale from Grid, and Own Scale). For the second choice the standard scale chosen by the user is required.

- **Grid File**: The ADSA program needs the name of the grid file that is acquired from the grid. It is noted that the grid has to be rotated very slightly (one or two pixels) in a vertical plane out of the vertical line, otherwise the ADSA program can not process the grid file.

- **Data File Name of Four Boundary Points**: After selecting four points on the grid file by clicking the left mouse button the program saves them in a file named like the grid file with ".o" extension.
- Horizontal Line: The number of horizontal lines between the two sets of boundary points must be counted and entered in this edit box.

- Vertical Line: The number of vertical lines between the two sets of boundary points must be counted and entered in this edit box.

- Distance between two Grid Lines: This number depends on the type of grid: typically the distance is 0.05, 0.025, or 0.01 cm.

3.2.1.4 Output Property Page

The last page (Figure 3.7), Output, contains four choices for the output file. The last radio button allows selection of a name of the file to be entered into the combo box.
If the other radio buttons are selected, the result of each image will be saved in a file under the name of the image with "res" extension.

After pressing the Ok push button, the required inputs are written in the input file, which was named "pendant.in" or "sessile.in" for pendant drop or sessile drop experiments, respectively.

### 3.2.2 Setup Property Sheet

Selecting Setup a property page containing five pages is opened. These pages contain all of the information with respect to pixel clock, frame type, video type, and frame size for setting the frame grabber. In detail these pages are as follows:
3.2.2.1 Source Property Page

Shown in Figure 3.8, the first page, Source, contains the following information:

- **Input Filter**: There are four choices for input filter that is applied to the incoming video signal. It is noted that if the AC-coupled video signal has chrominance information on it, software can be used to apply a chrominance notch filter to remove the chrominance information. The chrominance notch filter for 60 Hz is set to 3.58 MHz, while the chrominance filter for 50 Hz is set to 4.43 MHz. By default, no filter (first radio button) is selected.

- **Current Timeout**: Timeout edit box sets the timeout period, in seconds, used by the device driver. The device driver (frame grabber board) will wait this length of time for an operation to be completed before generating the timeout error and terminating the operation. It is noted that a period of zero seconds indicates that timeouts is disabled so that no timeout checking is performed in the device driver.

- **Pixel Clock**: The frequency of the pixel clock determines the video signal digitization rate. To determine the desired video pixel clock, the length of the horizontal line (in time) and the number of pixels per line (including the active pixels and blank pixels) are required. Dividing the number of pixels per line by the horizontal line period yields the proper pixel clock frequency. It is noted that according to sampling theory (Nyquist theorem), a frequency that is at least twice as fast as the highest frequency component of the input is required. In this way, the aliasing error, in which high frequency components of input erroneously appear as lower frequencies after sampling, is avoided.
The frame grabber that is used with this program supports both internal and external pixel clocks. Using the internal pixel clock the frame grabber board provides a programmable clock that generates the base frequency for video input timing. This clock is subsequently phase adjusted and divided down to produce the desired digitization rate. In this program the pixel clock was designed for any frequency from 1 Hz to 20 MHz, limited by the granularity of the pixel clock controller. The default frequencies are 12.5 MHz and 15 MHz for 60 Hz and 50 Hz image formats, respectively. Using an external pixel clock, on the other hand, the acquisition process can be synchronized with an external event. When the external pixel clock is selected, image acquisition starts when a high-to-low (falling-edge) transition occurs. In addition, a low-to-high edge (rising-edge) transition is possible when the low-to-high check box is checked. The frequency of the external trigger can range between 0 and 20 MHz.
• Input Source: A typical frame grabber board supports four monochrome videos. The board can accept an AC-coupled video input signal from one of the four software-selectable video channels (0 to 3), or a DC-coupled input from video channel 3 only. For user convenience, however, the combo box value can be 0 for any channel when the frame grabber is only connected to one of the input cables.

3.2.2.2 Video Type Property Page

To digitize the incoming video signal, the frame grabber board requires sync signals. The way in which the board determines the sync information depends on the type of video source connected to the board. As shown in Figure 3.9, the frame grabber accepts the following sync signals:

• Composite Video Signals: On the frame grabber board, the composite video sync signal can come from one of two sources: 1) the current channel being digitized, or 2) one of the unused video input channels. If the current channel is used as the composite sync signal, the composite sync signal is stripped from the video signal and fed into the device sync circuitry. Nonetheless, using a separate channel the sync signal is directly fed into the device sync circuitry. The voltage level of the analog sync signal is compared with the sync threshold to determine when the sync is asserted. The Sync Threshold combo box can determine the sync threshold.
Variable-Scan Video Signals: When using variable-scan video signals, the horizontal (line) and vertical (frame) sync are directly transmitted from the video source. The frame grabber board uses the rising edge of the external sync signals to reset the horizontal and vertical counters. There are two check boxes for controlling the variable-scan video signals. When the Variable Scan Input radio button is selected, line sync and frame sync are transmitted when the high-to-low (falling-edge) transitions occur. In addition, the low-to-high edge (rising-edge) transitions in line sync and frame sync are possible when the low-to-high check boxes are checked.

3.2.2.3 Sync Sentinel Property Page

Shown in Figure 3.10, Sync Sentinel page contains the sync signal insertion information.
The Sync Sentinel circuitry provides sync continuity for the frame grabber board. It is especially useful for noisy input sources such as VCRs. The problem with these video sources is that the frame grabber may interpret a noise spike in the video signal as a horizontal or vertical sync, or the board may miss some syncs that are below the threshold.

![Setup Frame Grabber](image)

**Figure 3.10 Sync Sentinel Page**

- **Sync Sentinel Check Box:** When this check box is checked the Sync Sentinel provides a window in which the sync can be detected. On the frame grabber board, the horizontal sync search position defines the pixel location within a line at which the board begins to search for the horizontal sync. It is noted that the default value is 95.0% of the total pixels per line. If the horizontal sync is not detected before the horizontal sync insert position is searched, the board inserts a horizontal sync to synchronize the video signal. The horizontal insert position is also programmable; the default value is 101.5% of the total pixels per line. The vertical sync position defines the line location within a field at which the board begins to search for the vertical sync. The vertical search position is also programmable; the default value is
50.0% of the total lines per field. If the vertical sync is not detected before the vertical insert position is reached, the board inserts a vertical sync to stay in sync with the video signal. The default value of the vertical insert position is 115% of the total lines per field. When the Sync Sentinel is disabled, vertical syncs are not inserted and the board waits for asynchronous vertical syncs and acquires frames only when the syncs are received.

3.2.2.4 Video Area Property Page

This section defines the frame grabber functions related to setting up the input controls of the frame grabber. Many of the input controls in this page manipulate the location or interpretations of the various transitions within the incoming video signal. These transitions identify, among other things, the active video area. The active video area is defined as that part of the incoming signal that contains valid video data (i.e., not blanking or sync information). Therefore, the video active area consists of the visible portion of those lines containing visible pixel data. The details of video signals may be found in Appendix B.

In this program, Video Area property page (Figure 3.11) encapsulates the information of the frame grabber signals in a group of edit boxes, which are explained as follows:

- **Back Porch Start**: Back porch is the position after the horizontal sync. The back porch can change between 0 to 4095 pixels (nominal is 60 pixels for 60 Hz image formats and 80 pixels for 50 Hz image format).

- **Active Pixel Count**: It is the number of pixels per line in the active video area. The active pixel is programmable between 0 to 4096 pixels (nominal is 640 pixels for 60 Hz image formats and 768 pixels for 50 Hz image format).
Active Line Count: It is the number of lines per field (or non-interlaced frame) in the active video area. The active line can change between 1 to 4096 pixels (nominal is 240 pixels for 60 Hz image formats and 288 pixels for 50 Hz image format).

First Active Pixel: It is the position of the first active video signal on the line, as a pixel value offset from the beginning of the horizontal sync. The valid range for first active pixel is between 0 to 4095 pixels (nominal is 125 pixels for 60 Hz image formats and 160 pixels for 50 Hz image format).

First Active Line: It is the location of the first active video signal within the field, as a line offset from the beginning of the vertical sync. The first active line can change between 0 to 4095 pixels (nominal is 16 pixels for 60 Hz image formats and 20 pixels for 50 Hz image format).
- Clamp Start Location: The position at which the clamping circuit starts holding the blanking level portion of the video signal to a reference level. The valid range for clamp start location is between 0 to 4095 pixels (nominal is 93 pixels for 60 Hz image formats and 90 pixels for 50 Hz image format).

- Clamp End Location: The position at which the clamping circuit stops holding the blanking level portion of the video signal to a reference level. The valid range for clamp start location is between 0 to 4095 pixels (nominal is 95 pixels for 60 Hz image formats and 93 pixels for 50 Hz image format).

- Total Pixel per Lines: The total number of pixels in a single horizontal line of video, where a horizontal line is defined as the area between two consecutive horizontal sync signals. The total number of pixels per line can change between 4 to 4096 pixels (nominal is 794 pixels for 60 Hz image formats and 960 pixels for 50 Hz image format).

- Total Lines per Field: The total number of lines in a single field of video, where a field is defined as the area between two consecutive vertical sync signals. The total number of lines per field can change between 1 to 4096 pixels (nominal is 262 pixels for 60 Hz image formats and 312 pixels for 50 Hz image format).

3.2.2.5 Frame Property Page

Frame (the region of interest) that is the final result of an acquisition is defined as the portion of the active video area to be digitized. Pixels outside this area are ignored. In this program, a rectangular frame on the active area within which data is acquired to display memory
(the device memory) or to system memory (hard disk) can be defined. Frame page (Figure 3.12) contains all information about frame. It is divided into two groups:

![Frame Page](image)

**Figure 3.12 Frame Page**

- **Frame Type**: The group of radio buttons, located at the left side of the page, defines the type of the frame. For the interlaced frame, the video signal is defined as two consecutive fields, where the start of each field is identified by the falling edge of the vertical sync. Thus, for an interlaced video signal, two fields are acquired to create the complete frame. The even field contains lines 0, 2, 4, and so on; the odd field contains lines 1, 3, 5, and so on. For a non-interlaced frame, the video signal is defined as a single field, where the start of the field is identified by the falling edge of the vertical sync. For both interlaced and non-interlaced video, a horizontal line of video is identified by the falling edge of the horizontal edge of the horizontal sync, and a field is composed of a collection of horizontal lines.
• Frame Size: The following edit boxes were designed for defining the frame area:

a. Frame Left: It is the first pixel in the region of interest, relative to the first active pixel, to digitize. The valid range for frame left is between 0 to 4095 pixels (nominal is 0 for both 60 Hz and 50 Hz image formats).

b. Frame Top: It is located at the first line of the region of interest, relative to the first active line to be digitized. The valid range for frame left is between 0 to 4095 pixels (nominal is 0 for both 60 Hz and 50 Hz image formats).

c. Frame Width: It is the number of pixels per line of video to be digitized. Frame width can change between 0 to 4096 pixels (nominal is 640 pixels for 60 Hz image formats and 768 pixels for 50 Hz image format).

d. Frame Height: It is the number of lines per frame of video (or non-interlaced frame) to be digitized. Frame height can change between 1 to 4096 pixels (nominal is 480 pixels for 60 Hz image formats and 576 pixels for 50 Hz image format)

e. Input Scaling: These two edit boxes provide the scale factors for the horizontal direction (range between pixels) and the vertical direction (range between lines) separately. The minimum scale factor is 1; the maximum scale factor is 16 (nominal is 1).

The spatial relationship of these values to the active video and frame areas is shown in Figure 3.17. The actual sampling of the fields is defined by both the horizontal and vertical input setting.
It is noted that the frame height is specified in lines per frame, not lines per field. For interlaced video, this value may exceed the Active Line Count. Shown in Figure 3.13, the Active Line Count specifies the number of lines per field for a non-interlaced frame. If the frame is interlaced then the maximum number of lines per frame is the Active Line Count multiplied by 2.

![Figure 3.13 Spatial Relationship of Video and Frame Areas](image_url)

### 3.3 Dialog Boxes

A dialog box is a special window that contains one or more controls. This program contains several dialog boxes, which are described in this section.
3.3.1 Open Device Dialog Box

During the installation procedure, a name (alias) must be entered to identify the device for future operations. In fact, alias is the string that identifies the device. Each board installed in the system must have a globally unique alias. At first, the program counts the number of boards that currently have been installed in the system. If the number of devices were greater than 0, at least one board exists in the system. After selecting one of the boards from Alias combo box in the select Device dialog box (Figure 3.14), the program opens the device. It is noted that only a single device can be selected at one time and the program retrieves only information of a single device at one time. However, if more than one board has been installed in the host computer the one to be used has to be connected to the camera.

3.3.2 Acquire Dialog Box

As shown in Figure 3.15, Acquire dialog box contains three radio buttons that determine the image acquisition methods (i.e., Single, Sequential Single, and Multiple acquisition).

The first method, Single Acquisition, acquires one image and saves it with a user-defined filename. The program appends ".img" to the string in the filename edit box. In this method, the
image is immediately displayed on the screen, whereas to display images acquired by the other methods, image files must be opened by *Open* dialog box.

![Acquire dialog box with three types of acquisitions](image)

**Figure 3.15** *Acquire* dialog box with three types of acquisitions

In the *Sequential Single* method, single frames are sequentially acquired for a desired period of time and the resulting images are saved in separate files. Since the file saving speed is dominant in this procedure, this type of acquisition is used when high speed (frequency) is not necessary (i.e. no more than 3 images per second, i.e. an interval of 0.33). In this case, three parameters are required:

- **Image File name**: In this edit box, a string (e.g., “test” without any extension) should be specified as the image filename. Because the program appends the acquisition time of each image and “.img” extension to the string, the images are saved in distinct files.

- **Duration**: The desired duration of the experiment in seconds must be entered in this edit box.
- Interval: The period of time in seconds between two sequential images is specified in this edit box. For instance, 2 seconds interval means that an image is acquired every two seconds.

Finally, the *Multiple Acquisition* method acquires several frames in a user-defined duration, but saves them in a single file. In this method, up to 30 frames per second may be acquired; consequently, it is suitable for fast changes in surface tensions where the observations should be performed in a small fraction of a second. In this method, five parameters should be considered:

- Image File name: In the same manner, only one string without any extension must be entered and the program appends “.img” extension to obtain the image filename.

- Duration: The desired duration of the experiment in seconds must be entered in this edit box.

- Number of Images per Second: This number is set to 30 images by default, and this edit box is disabled in this version of the program. However, a future version of the program will be developed to adjust frequency for acquisition with different rates.

- Separate the Images: The ADSA program can process images only saved in separate files. Turning on the check box, the program separates the images and appends the acquisition time to the filename. In other words, this function makes the results of the multiple acquisitions compatible with the ADSA program.

- The read only edit box in the bottom of this dialog box shows how many images can be acquired in device memory without transferring into the host memory. In the multiple-acquisition mode, frames are initially acquired in the device memory, which
is restricted to one third of the total memory of the system. Thus, after a determinable number of images the device memory must be transferred to the host memory to make space available for the next acquisitions. Inevitably, during the data transmission from device to host, the frame grabber cannot perform its task. To alleviate this problem the available memory of the system must be extended. The program will automatically compute and report the number of images that can be acquired without interruption.

Finally, the ADSA program needs the image filenames for processing. Therefore, the program saves them in a text file with a name similar to the image filenames but with ".txt" extension. For instance, if the user enters the "test" for the image filename, the program will append the time of each acquisition and the ".img" to the base string and write the results in a text file named "test.txt". This text file is where the user wants to process the images acquired in the past.

3.3.3 Open Image Dialog Box

This dialog box is used to open an image file that has already been saved on the hard drive (Figure 3.16).

After selecting the image filename the Display dialog box (Figure 3.17) is opened. For the images acquired by the Single or the Sequential Single method, where each image file contains only one image, the counter in the dialog box is automatically set to 1. However, if the images have been acquired with the Multiple Acquisition method, to display the \( n \)th image the counter has to be set to \( n \) (e.g., for the second frame of the image file the counter is set to 2).
Figure 3.16 *Open Image* Dialog Box

Figure 3.17 *Display* Dialog Box
CHAPTER 4

DATABASE OF THE ADSA PC-VERSION

In this chapter the data structure of the PC-version of ADSA will be explained. Using C++ language, it is feasible to use both classes and structures. A class is a data type that can contain two types of data elements (i.e., single data elements, arrays, pointers, arrays of pointers or objects of other classes) as well as functions operating on an object of the class by accessing the data elements. Thus, a class combines the definition of the elementary data making up an object and the means of manipulating the data belonging to the individual objects of the class. The data and functions within a class are called data members and member functions, respectively.

In this program, each class is derived from a base class of Microsoft Foundation Classes (MFC) so that it inherits all data members and member functions of its parent. The important classes of the program are CExper, CFrmGrb, CView, and CDoc that all encapsulate member variables and member functions required for storing items and processing them, respectively. In the following sections, the data members and member function of each class are explained in detail.

4.1 CExper Class

The parent of CExper class is COleServerDoc that inherits the functionality of its base class COleLinkingDoc, as well as its indirect base classes, which are CDocument and ColeDocument, see Figure 4.1.
**CErper** class inherits a large number of functions from its parents including the `Serialize()` function. This function is responsible for serializing the data required for capturing the current state of objects. The `Serialize()` function has an argument that is an object of the `CArchive` class used to read and write the object data. The object has a member function, `IsStoring`, which indicates whether `Serialize()` is storing (writing data) or loading (reading data). Therefore, the object can be directly restored from the memory. Using the `Serialize()` function in **CErper** class, all data members of this class, which are encapsulated in the *Properties* property sheet, can be read or written from or to an archive. The codes of this function may be found in Appendix C.

Shown in Table 4.1, each data member of **CErper** Class is an object of the controls in the property pages of the *Properties* property sheet explained in chapter 3. For example, the image format radio buttons from the *Experiment* page are defined as an object named *Resolution* in the **CErper** class; also the grid file combo box from the *Grid* page is defined as *m-ComboGrid* in the **CErper** class.

**Figure 4.1** Hierarchy chart of *ColeServerDoc* class
Table 4.1 Relation between *Properties* property sheet controls and *CExper* Class data members

<table>
<thead>
<tr>
<th>Controls in the <em>Experiment</em> Property Page</th>
<th>Data members of the <em>CExper</em> Class</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name</strong></td>
<td><strong>Type</strong></td>
</tr>
<tr>
<td>Image format including IMG and TIFF</td>
<td>Radio buttons</td>
</tr>
<tr>
<td>Drop shape</td>
<td>Radio buttons</td>
</tr>
<tr>
<td>Number of the image(s)</td>
<td>Edit box</td>
</tr>
<tr>
<td>Cut-off level coordinate (for Pendant drop)</td>
<td>Check box</td>
</tr>
<tr>
<td>File of rotation coordinates (for Pendant drop)</td>
<td>Combo box</td>
</tr>
<tr>
<td>Contact angle (for sessile drop)</td>
<td>Radio buttons</td>
</tr>
<tr>
<td>Coordinates of the left and right points of the drop when contact angle is near 90° (for sessile drop)</td>
<td>Edit boxes</td>
</tr>
<tr>
<td>File containing the images names</td>
<td>Combo box</td>
</tr>
<tr>
<td>Starting Y coordinate for searching profile</td>
<td>Edit box</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Controls in the <em>Calculation</em> Property Page</th>
<th>Data members of the <em>CExper</em> Class</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name</strong></td>
<td><strong>Type</strong></td>
</tr>
<tr>
<td>Cubic spline fit</td>
<td>Radio buttons</td>
</tr>
<tr>
<td>Local gravity</td>
<td>Edit box</td>
</tr>
<tr>
<td>Number of iterations of ADSA</td>
<td>Edit box</td>
</tr>
<tr>
<td>Number of points per iteration</td>
<td>Edit box</td>
</tr>
<tr>
<td>Showing the drop coordinate</td>
<td>Check box</td>
</tr>
<tr>
<td>Density</td>
<td>Edit box</td>
</tr>
<tr>
<td>Center of the image coordinate</td>
<td>Edit boxes</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Controls in the <em>Grid</em> Property Page</th>
<th>Data members of the <em>CExper</em> Class</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name</strong></td>
<td><strong>Type</strong></td>
</tr>
<tr>
<td>Scale type including grid and own scale</td>
<td>Radio buttons</td>
</tr>
<tr>
<td>Grid file</td>
<td>Combo box</td>
</tr>
<tr>
<td>Horizontal lines of the grid file</td>
<td>Radio buttons</td>
</tr>
<tr>
<td>Vertical lines of the grid file</td>
<td>Edit box</td>
</tr>
<tr>
<td>Distance between two grid lines</td>
<td>Check box</td>
</tr>
<tr>
<td>Data File of Four Boundary Points of the grid file</td>
<td>Combo box</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Controls in the <em>Output</em> Property Page</th>
<th>Data members of the <em>CExper</em> Class</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name</strong></td>
<td><strong>Type</strong></td>
</tr>
<tr>
<td>Choices for output file</td>
<td>Radio buttons</td>
</tr>
<tr>
<td>File containing the summary of results</td>
<td>Combo box</td>
</tr>
</tbody>
</table>
4.2 CFrmGrb Class

This class contains data members and member functions that are required for controlling the performance of the frame grabber. The parent of CFrmGrb class is also COleServerDoc that inherits the functionality of its base class COleLinkingDoc, and its indirect base classes, CDocument and COleDocument (Figure 4.1).

Similar to the CExper class, the CFrmGrb class inherits a large number of functions from its parents including the Serialize() function. Using the Serialize() function in CFrmGrb class, all data members of this class, which are encapsulated in the Setup property sheet, can be read or written from or to an archive.

The data members of CFrmGrb Class, see Table 4.2, are objects of controls in the property pages of the Setup property sheet explained in chapter 3. For instance, the input filter radio buttons from the Source page are defined as an object named ulFilter in the CFrmGrb class; also the image height edit box from the Frame page is defined as m-eHeight in the CFrmGrb class.

Table 4.2 Relation between Setup property sheet controls and CFrmGrb Class data members

<table>
<thead>
<tr>
<th>Controls in the Source Property Page</th>
<th>Data members of the CFrmGrb Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Type</td>
</tr>
<tr>
<td>Input filter for noises</td>
<td>Radio buttons</td>
</tr>
<tr>
<td>Current timeout</td>
<td>Edit box</td>
</tr>
<tr>
<td>Pixel clock including internal clock and external clock</td>
<td>Radio buttons</td>
</tr>
<tr>
<td>Frequency of internal clock</td>
<td>Edit box</td>
</tr>
<tr>
<td>External clock for falling-edge or rising-edge transition</td>
<td>Check box</td>
</tr>
<tr>
<td>Input source list</td>
<td>Combo box</td>
</tr>
</tbody>
</table>
Table 4.2 Relation between Setup property sheet controls and C FrmGrb Class data members (Continued)

<table>
<thead>
<tr>
<th>Controls in the Video Type Property Page</th>
<th>Data members of the C FrmGrb Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Type</td>
</tr>
<tr>
<td>Video type including composite video and variable scan input</td>
<td>Radio buttons</td>
</tr>
<tr>
<td>Sync threshold (for composite video)</td>
<td>Combo box</td>
</tr>
<tr>
<td>Sync source (for composite video)</td>
<td>Combo box</td>
</tr>
<tr>
<td>Line sync when falling-edge or rising-edge transition occurs (for variable scan input)</td>
<td>Check box</td>
</tr>
<tr>
<td>Frame sync when falling-edge or rising-edge transition occurs (for variable scan input)</td>
<td>Check box</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Controls in the Sync Sentinel Property Page</th>
<th>Data members of the C FrmGrb Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Type</td>
</tr>
<tr>
<td>The Sync Sentinel circuitry</td>
<td>Check box</td>
</tr>
<tr>
<td>Horizontal sync search position</td>
<td>Edit box</td>
</tr>
<tr>
<td>Horizontal sync insert position</td>
<td>Edit box</td>
</tr>
<tr>
<td>Vertical sync search position</td>
<td>Edit box</td>
</tr>
<tr>
<td>Vertical sync insert position</td>
<td>Edit box</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Controls in the Video Area Property Page</th>
<th>Data members of the C FrmGrb Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Type</td>
</tr>
<tr>
<td>Back porch start</td>
<td>Edit box</td>
</tr>
<tr>
<td>First active pixel</td>
<td>Edit box</td>
</tr>
<tr>
<td>Active pixel count</td>
<td>Edit box</td>
</tr>
<tr>
<td>Total pixels per line</td>
<td>Edit box</td>
</tr>
<tr>
<td>First active line</td>
<td>Edit box</td>
</tr>
<tr>
<td>Active line count</td>
<td>Edit box</td>
</tr>
<tr>
<td>Total lines per field</td>
<td>Edit box</td>
</tr>
<tr>
<td>Clamp start location</td>
<td>Edit box</td>
</tr>
<tr>
<td>Clamp end location</td>
<td>Edit box</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Controls in the Frame Property Page</th>
<th>Data members of the C FrmGrb Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Type</td>
</tr>
<tr>
<td>Frame types</td>
<td>Radio buttons</td>
</tr>
<tr>
<td>Frame left</td>
<td>Edit box</td>
</tr>
<tr>
<td>Frame width</td>
<td>Edit box</td>
</tr>
<tr>
<td>Input Scaling for the horizontal direction (range between pixels)</td>
<td>Edit box</td>
</tr>
<tr>
<td>Frame top</td>
<td>Edit box</td>
</tr>
<tr>
<td>Frame height</td>
<td>Edit box</td>
</tr>
<tr>
<td>Input Scaling for the vertical direction (range between lines)</td>
<td>Edit box</td>
</tr>
</tbody>
</table>
Other functions implemented to the program are \textit{CreateDIBDisplay()} and \textit{CleanupDIBPointers()}. Using data obtained from an image acquisition, the first function displays the image in the client area of the application. Assuming 8 bit Device Independent Bitmap (DIB) data, this function allocates memory for the DIB header and associated palette. The second function releases the memory allocated for the DIB data explained before.

4.2 CView Class

The \textit{CView} class provides the basic functionality for user-defined view classes. A view is attached to a document and acts as an intermediary between the document and the user: the view renders an image of the document on the screen or printer and interprets user input as operations upon the document.

The \textit{CView} class inherits the functionality of its base classes \textit{CWnd}, \textit{CcmdTarget}, and \textit{CObject}, as shown in Figure 4.2.

![Figure 4.2 Hierarchy chart of CView class](image)

In addition, five functions, summarized in Table 4.3, have been implemented to perform particular tasks required for sessile drop and pendant drop experiments.
Table 4.3 Additional functions in CView class

<table>
<thead>
<tr>
<th>Function</th>
<th>Sessile Drop</th>
<th>Pendant Drop</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>OnRButtonDown()</td>
<td>To select two points located in the left and right sides of the drop when the contact angle is near 90°</td>
<td>To select two points located in the top and bottom of the vertical line used for aligning the camera</td>
<td>Clicking the right mouse button</td>
</tr>
<tr>
<td>OnLButtonDownDblClk()</td>
<td>To select the central point of the image</td>
<td>To select the central point of the image</td>
<td>Double clicking the right mouse button</td>
</tr>
<tr>
<td>OnLButtonDown()</td>
<td>To select four boundary points in the grid file</td>
<td>To select four boundary points in the grid file</td>
<td>Clicking the left mouse button</td>
</tr>
<tr>
<td>OnRButtonDownDblClk()</td>
<td>Not available</td>
<td>To select two points located in the left and right sides of the drop for determining the cut-off level coordinate</td>
<td>Double clicking the right mouse button</td>
</tr>
<tr>
<td>OnTextEdit()</td>
<td>To cancel and reselect a point selected by the above functions</td>
<td>To delete and reselect a point selected by the above functions</td>
<td>Selecting Undo menu item in Edit menu</td>
</tr>
</tbody>
</table>

4.2 CDoc Class

This class supports standard operations such as creating, loading, and saving a document. A document receives commands forwarded by the active view, which renders an image of the document in a frame window.

As shown in Figure 4.3, the parent of CDoc class is CDocument that inherits the functionality of its base classes CCmdTargete, and Object. From figures 4.1 to 4.3, it can be concluded that the CDoc, CExper, and CFrmgrb classes are all derived from the same classes: therefore, the CDoc class can access the data members and member functions of these classes by implementing their objects to CDoc class.
The \textit{CDoc} class inherits a large number of functions from its parents including the \textit{OnNewDocument()} and \textit{Serialize()} functions. The first function is required for initializing the data members of \textit{CExper}, and \textit{CFrmGrb} by default values. The second function invokes the \textit{Serialize()} member functions of the \textit{CExper}, and \textit{CFrmGrb} classes to save their data members when the \textit{File Save} menu item is selected.

In addition, the following functions have been implemented to provide particular tasks required for sessile drop and pendant drop experiments. The codes of these functions may be found in Appendix C.

- \textit{OnImageDeviceid():} The program searches for all frame grabber boards in the system, each of which is identified by a globally unique name. After selecting the \textit{Open Device} submenu item, the names of the frame grabbers appear in the combo box of the \textit{Open Device} dialog box. While a name is selected, the \textit{OnImageDeviceid()} function opens the corresponding frame grabber board and retrieves its properties including the memory size, Device ID, and type of the frame grabber board. It is noted that only a single frame grabber board can be selected at one time.

- \textit{OnImageAcquire():} This function covers three acquisition modes including single, single sequential, and multiple acquisitions.
In the single acquisition, a block of memory equal to the size of one image is allocated in the host memory, and the image is stored in the host computer. Then the image is saved in a file with a user-defined name. In this mode, the single image is immediately displayed on the screen, and the memory is released.

In the single sequential acquisition, the same process is repeated for a number of times. The number of iterations is defined based on the duration of the acquisition and interval between two sequential images. In addition, the images are saved in separate files, each of which is entitled with the acquisition time attached to a user-defined string. Then, each image may be displayed using the Open Image submenu item, but the memory is immediately released when the images are saved.

In the multiple acquisitions, a block of memory equal to the device memory is allocated. First, the images are acquired into the device memory. When the device memory becomes full, the images are transferred to the host memory so that the device is released to continue the acquisition. Thus, the maximum number of images acquired without interruption depends on the size of the device memory. In this mode, images can be saved in either one file or separate files. It is noted that the ADSA program requires separate image files, each of which is identified by the acquisition time given with a user-defined string. Then, the images may be displayed using the Open Image submenu.

- OnProperties(): When the Properties property sheet is closed with the OK pushbutton, the values of CExper data members are saved in an input file entitled "pendant.in" and "sessile.in" for pendant drop and sessile drop experiments.
respectively. The input files are required for the ADSA program while processing the images.

- **OnImageOpen()**: This function is invoked from the *Open Image* dialog box to display the images saved in the image files. For the images acquired via single and single sequential acquisition modes, the filenames are sufficient; however, for images acquired via multiple acquisition mode, the image number must also be specified. In addition, two functions have been programmed to increment and decrement the image number to show the next and the last images, respectively.

- **OnImageClosedevice()**: This function closes the device to release the system resources reserved for the acquisition process. It is crucial that an application closes all open devices before termination; devices that are not explicitly closed not only use system resources but also are unavailable until the computer is restarted.

- **OnRun()**: While all the processing inputs and images are prepared, the ADSA program may be launched. The *OnRun()* function links the graphic user interface to the ADSA program, which is a console application.
CHAPTER 5

TESTING THE ADSA PC-VERSION

This chapter outlines the results of both sessile drop and pendant drop experiments that were carried out to test the ADSA PC-version program. In order to assess the results of the program, the images were simultaneously acquired and separately processed in the PC and UNIX systems.

In the following sections, the materials used in the experiments and the experimental procedures for sessile drop and pendant drop experiments are explained. Finally, the results obtained from the experiments are illustrated.

5.1 Sessile Drop Experiment

5.1.1 Materials (Solid Surfaces and Liquid)

Poly (methyl methacrylate) (PMMA) purchased from BASF Co. was used as polymer in these experiments. A 2% PMMA/chloroform solution was prepared using chloroform (Sigma-Aldrich, 99.9+% A.C.C. HPLC grade) as the solvent. The PMMA surfaces were prepared by a solvent-casting technique on cleaned and dried silicon wafers. Finally, distilled water was used as liquid in these experiments.

5.1.1.1 Preparation of Silicon Substrate

Silicon wafers <100> (Silicon sense, Nashua, NH; thickness 525 ± 50 μm) were selected as the substrate for the polymer coating. They were obtained as circular discs of about 10 cm diameter and were cut into rectangular shapes of about 2.5 cm × 2.5 cm. For low-rate dynamic
contact angle measurements by ADSA, liquid was supplied to the sessile drop from below the wafer surfaces using a motorized syringe device. In order to facilitate such an experimental procedure, a hole of about 1 mm diameter was made in the center of each rectangular wafer surface using a diamond drill bit (SMS-0.0027) from Lunzer (New York, NY). The rectangular wafer surfaces were then rinsed with acetone to remove dirt and fingerprints from the surfaces. After drying, they were soaked in chromic acid for at least 24 hours, rinsed with doubly distilled water, and dried under a heat lamp before the polymer coating. In order to avoid leakage between the stainless steel needle (Chromatographic Specialties, Brockville, Ont.; N723 needles pt.# 3, H91023) and the hole on the wafer surface, Teflon tape was wrapped around the end of the needle before insertion into the hole.

5.1.1.2 Coating Procedure (Solvent-Casting Technique)

The solid surfaces were prepared by a solvent-casting technique. A few drops of the 2% PMMA/chloroform solution were deposited on dried silicon wafers inside glass dishes overnight: the solution spread and a thin layer of the PMMA formed on the wafer surface after chloroform evaporated. This technique produces high-quality-coated surfaces as the surface roughness is on the order of nanometers or less.

5.1.2 Procedures

In the experiments, an initial liquid drop of about 0.3 cm radius was carefully deposited, covering the hole on the surface. This is to ensure that the drop will increase axisymmetrically in the center of the image field when liquid is supplied from the bottom of the surface. Then, the motor in the motorized syringe mechanism explained in Chapter 2 was set to a specific speed by adjusting the voltage from a voltage controller. Such a syringe mechanism pushes the syringe
plunger, leading to an increase in drop volume and hence the three-phase contact radius. A sequence of images of the drop was simultaneously acquired by the frame grabber boards installed in the PC and UNIX systems at a rate of 1 image every 2.5 sec. When the three-phase contact radius increased to about 0.5 cm, the motor was reversed to decrease drop volume and hence to recede the three-phase contact line. For each low-rate dynamic contact angle experiment, at least 240 images were acquired.

5.1.3 Results and Discussion

To assess the ADSA PC-version program, five experiments were conducted using PMMA surfaces and distilled water. The experimental results derived from the ADSA PC-version and the UNIX version are illustrated in Figure 5.1 and Figure 5.2, respectively. To obtain a more effective comparison, the results of the two figures during the advancing period (the first 250 seconds) are integrated in Figure 5.3. It is observed that the contact angle, radius of curvature, and volume of the drop agree well in the two systems. However, the liquid surface tensions in both systems ($\gamma_{lv}$) fluctuate with time, which can be due to any small imperfection in the solid substrate. It is noted that the results of $\gamma_{lv}$ obtained from the PC system show less scatter than those obtained from the UNIX system. In addition, the error of $\gamma_{lv}$ measured for each image as well as the mean of errors is considerably smaller in the PC-version than that in the UNIX version. In this experiment, the mean errors of $\gamma_{lv}$ are 0.62 and 1.31 for the results obtained from the ADSA PC-version and UNIX version, respectively. The discrepancy between the results of $\gamma_{lv}$ in the two systems can be explained as follows:

Essentially, the ADSA program selects 200 points (i.e., 10 iterations with 20 points per iteration) from each image for the determination of the drop profile. Since the frame grabber
boards utilized in the PC and UNIX systems are different, the images acquired in the two systems are never identical. Likewise, the 200 points selected by ADSA from the images acquired in the PC and UNIX systems are different so that the results obtained from the two systems can be slightly different. It would appear that the PC frame grabber performs better than the UNIX frame grabber.

Finally, the mean values of advancing contact angles of five experiments are summarized in Table 5.1. The table shows that the difference between the contact angles obtained from the PC system and UNIX system does not exceed 0.1°.

### Table 5.1 Contact angles obtained from five experiments using the PC and UNIX systems

<table>
<thead>
<tr>
<th>ADSA PC-version</th>
<th>ADSA UNIX version</th>
</tr>
</thead>
<tbody>
<tr>
<td>rate (mm/min.)</td>
<td>rate (mm/min.)</td>
</tr>
<tr>
<td>θ (degree)</td>
<td>θ (degree)</td>
</tr>
<tr>
<td>-----------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>0.349</td>
<td>0.350</td>
</tr>
<tr>
<td>79.32 ± 0.36</td>
<td>79.26 ± 0.39</td>
</tr>
<tr>
<td>0.335</td>
<td>0.337</td>
</tr>
<tr>
<td>78.96 ± 0.32</td>
<td>78.87 ± 0.37</td>
</tr>
<tr>
<td>0.367</td>
<td>0.369</td>
</tr>
<tr>
<td>79.63 ± 0.28</td>
<td>79.55 ± 0.35</td>
</tr>
<tr>
<td>0.329</td>
<td>0.332</td>
</tr>
<tr>
<td>78.89 ± 0.37</td>
<td>78.95 ± 0.39</td>
</tr>
<tr>
<td>0.356</td>
<td>0.358</td>
</tr>
<tr>
<td>79.47 ± 0.46</td>
<td>79.32 ± 0.47</td>
</tr>
<tr>
<td>Mean θ</td>
<td>Mean θ</td>
</tr>
<tr>
<td>79.25 ± 0.42</td>
<td>79.19 ± 0.44</td>
</tr>
</tbody>
</table>
mean $\theta_a = 79.32 \pm 0.36^\circ$

rate $= 0.349$ mm/min

Figure 5.1 Experimental results of water on PMMA surface derived from ADSA PC-version
Figure 5.2 Experimental results of water on PMMA surface derived from ADSA UNIX version
Figure 5.3 Experimental results of water on PMMA surface derived from ADSA PC-version and ADSA UNIX version during the advancing period.
5.2 Pendant Drop Experiment

Two types of the pendant drop experiments were conducted: static and dynamic.

5.1.1 Materials

- In the static experiments, n-Hexadecane and n-Undecane obtained from Aldrich were used.
- In the dynamic cycling experiments, SDS (sodium dodecyl sulfate) obtained from Schwarz/Mann Biotech, a Division of ICN Biomedicals Inc. (cat# S11034, $M_w = 288.38$, purity: greater than 99%). SDS was dissolved in distilled water for a 8mM concentration.

5.1.2 Procedures

- For each static experiment, a well-deformed drop was formed and a time dependent study was undertaken in which surface tension was measured. During the experiments, a sequence of images of the drops were simultaneously acquired by the frame grabber boards installed in the PC and UNIX systems at a rate of 1 image every 3 sec. For each experiment, at least 40 images were acquired.
- In the cycling experiment, a large, well deformed drop (to minimize measurement errors since small drops tend to give larger errors) of aqueous SDS solution was formed. Then, the stepper motor explained in Chapter 2 was used to control the compression ratio of surface film and the speed of dynamic cycling processes. As mentioned before, the stepper motor can move a distance of 20,000 steps (each step
represents ½ micrometer). The compression ratios of surfactant films are determined by setting the number of steps. In the experiments, the number of steps was set at 400 steps to obtain the compression ratio of about 40%. Then, drop oscillations were initiated in a sawtooth manner for approximately 11 cycles. The experiments show the effect of changing the surface area of the SDS solution on the surface tension. During the experiment, a sequence of images of the drop was simultaneously acquired by the frame grabber boards installed in the PC and UNIX systems at a rate of 1 image every 0.4 sec. For each experiment, at least 250 images were acquired.

5.2.3 Results and Discussion

The experimental results derived from the ADSA PC-version and UNIX version for n-Hexadecane are illustrated in Figure 5.4 and Figure 5.5, respectively. To obtain a more effective comparison, the results of the two figures are integrated in Figure 5.6. In addition, the results of the static experiment derived from the PC and UNIX systems for n-Undecane are shown in Figure 5.7 and Figure 5.8, respectively, and to compare the results, these two figures are integrated in Figure 5.9. In both Hexadecane and n-Undecane experiments, the area and the volume of the drops increase at the first few seconds possibly because of leakage the liquid into the drop; at longer times, the area and the volume decrease presumably due to liquid evaporation. The results for volume and area of the drops obtained from the ADSA PC-version and the ADSA UNIX version for the Hexadecane and n-Undecane experiments are slightly different due to a different cut-off-level selected in the two systems. From the Figure 5.6 and Figure 5.9, it can be observed that the values of $\gamma_v$ obtained from the ADSA PC-version and the ADSA UNIX version are statistically different. Indeed, the results of $\gamma_v$ obtained from the PC system show
less scatter than those obtained from the UNIX system, again suggesting better performance of the PC frame grabber compare to the UNIX frame grabber.

Finally, the results of the cycling experiment derived from the ADSA PC-version and the UNIX version for SDS solution are illustrated in Figure 5.10 and Figure 5.11, respectively. It can be observed that the results of $\gamma_{lv}$ from the two systems have the same pattern during the cycling. However, the results derived from the ADSA PC-version show less scatter than those derived from the ADSA UNIX version.

5.3 Summary

From the results of the sessile drop and pendant drop experiments, it can be concluded that the ADSA-PC version is a reliable program for the determination of contact angle $\theta$ and liquid-vapor surface tension $\gamma_{lv}$. It would appear that the ADSA PC-version works better than the ADSA UNIX version presumably due to better performance of the PC frame grabber compare to the UNIX frame grabber.
Figure 5.4 Experimental results of n-Hexadecane derived from ADSA PC-version
Figure 5.5 Experimental results of n-Hexadecane derived from ADSA UNIX version
Figure 5.6 Experimental results of n-Hexadecane derived from ADSA PC-version and ADSA UNIX version
Figure 5.7 Experimental results of n-Undecane derived from ADSA PC-version
Figure 5.8 Experimental results of n-Undecane derived from ADSA UNIX version
Figure 5.9 Experimental results of n-Undecane derived from ADSA PC-version and ADSA UNIX version
Figure 5.10 Experimental results of SDS solution from ADSA PC-version
Figure 5.11 Experimental results of SDS solution derived from ADSA UNIX version
CHAPTER 6

SUMMARY

In this chapter, the advantages of the PC-version of the axisymmetric drop shape analysis (ADSA) program and the future work proposed for the enhancement of the performance of ADSA is discussed.

6.1 Advantages of the ADSA PC-Version

The ADSA PC-version has been developed to remove drawbacks of the UNIX version, which limit wider use and improvements of ADSA. The advantages of the ADSA PC-version can be summarized as follows:

- Developed on a Windows platform, the ADSA PC-version is associated with a user-friendly graphical interface that encourages researchers to use it in their experimental setup. On the other hand, the ADSA UNIX version requires special training of personnel. Thus, a growing demand is expected for the new software.

- Essentially, the ADSA program requires a large number of inputs for computation. In this way, the possibility of entering erroneous data in the UNIX system increases while the user is frustrated. However, in the ADSA PC-version, inputs are associated with suitable descriptions and icons that make sense to the user.

- In the ADSA PC-version, using several advanced frame grabber functions, the performance of the software has been enhanced to acquire images at different rates,
and with different pixel size. In addition, the images can be filtered for a variety of noise types, synchronized by external triggers, and acquired from different cameras.

- Written in C++ language, the ADSA PC-version is an object-oriented program allowing the integration of previous, current, and future work without reducing programming independency and flexibility; consequently, the development, modification, and improvement of the software will be more efficient in future work.

- Practical software programming will require interaction between different applications performing separate tasks. On the Microsoft Windows platform, the applications featuring Object Linking and Embedding (OLE) can share data structures among each other. In the ADSA PC-version, the data structures have been designed based on Microsoft Foundation Classes (MFC) so that the program can send and receive data to and from the Microsoft standard applications including Microsoft Office.

### 6.2 Future Work

- Although ADSA is the most powerful technique for the measurement of interfacial properties, it has failed for drop profiles close to spherical shapes. Based on the preliminary investigations, it is suspected that optical distortion caused by the presently used white light source causes the problem. Therefore, it is proposed to study the effect of different lighting systems and conditions on the performance of the methodology.
Since image-processing techniques are based on statistical or approximate-reasoning methods, there is no theoretical approach to verify the superiority of one method over the other. Thus, it is proposed to implement different image analysis techniques (e.g., Sobel, Entropic, Prewitt, Roberts, Laplacian of Gaussian, and Canny methods) into the ADSA program to select the most appropriate method for an application.

Line tension in liquid-liquid-fluid systems is currently obtained through physical observation of the slope of the drop at the three phase contact line; this method is prone to error due to the small magnitude of line tension and low quality of images in vicinity of intersection points. On the other hand, ADSA is successful in the determination of interfacial properties in liquid-fluid and solid-liquid-fluid systems so that it is proposed to generalize the ADSA program for the three phases of liquid-liquid-fluid system to find the values of line tension.

Written in ANSI C++, the kernel of the ADSA program can be compiled and executed by different operating systems (OS) including UNIX, Windows, or Linux. However, the graphical user interface prepared for one OS cannot be adopted for another one. Therefore, it is proposed to prepare a generic graphical user interface using Java applets. It is noted that Java applets are executed by Internet browsers compatible with Java language (e.g., Internet Explorer and Netscape Navigator).
APPENDIX A

AXISYMMETRIC DROP SHAPE ANALYSIS (ADSA)

The ADSA technique utilizes a combination of nonlinear least-squares fit and the Newton-Raphson method to fit the experimental drop profile to a Laplacian curve, and thus determines liquid-fluid interfacial tensions from the shape of axisymmetric pendant or sessile drops. In addition, the drop volume, surface area, and the radius of the three-phase contact line (sessile drop) are provided as output. The program requires as input several coordinate points along the drop profile, the density difference between the bulb phases, the local gravitational constant, and the distance between the base of the drop and the horizontal coordinate axis. The first step in the analysis of a drop is the determination of the deviation of its profile from the shape dictated by the Laplace equation of capillarity, Eq. 1.1. Experimental profile points \( U_i, i=1, 2, \ldots, N \), which describe the meridian surface are compared with \( u = u(s) \), a calculated Laplacian curve, by computing the normal distance, \( d_i \), between \( U_i \) and \( u \) as illustrated in Figure A.1: the \( U_i \) points are obtained from image analysis techniques.

A.1 Experimental Setup for Digital Image Processing

The block diagram of the experimental setup using digital image processing is illustrated in Figure A.2. The analog video signal of the pendant or sessile drop is transmitted to the image processor which performs the frame grabbing and digitization of the image. The digitized image contains the image data in the form of digital picture elements or pixels. The value of each pixel
is called the intensity or grey level (in the black-and-white case). The most commonly used equipment utilizes 640 × 480 pixels and 256 grey levels to represent an image where 0 and 255 represent black and white, respectively. Thus, a digitized image is mathematically represented by an array of real numbers from 0 to 255. A computer is used to acquire the image from the image processor and to perform the image analysis and computation.

A.2 Drop Profile Analysis to a Resolution of ±1 pixel

Once an image of the drop is obtained and stored in the memory, the ADSA program automatically finds the drop profile coordinates. To extract the drop profile coordinates from the image, various methods may be adopted. The most common methods in digital image processing are global thresholding, adaptive thresholding, and gradient magnitude. It is desired to use a method that is applicable to both pendant and sessile drops, requires minimal computation time, and is sensitive to lighting condition. Thus, the third method is applied to the ADSA program to obtain the drop profile. This method is based on the application of a local edge operator (or difference) operator at every pixel. The output of such an operator will have high magnitudes at pixels where the grey levels are changing rapidly, such as at the edge of the drop. In order to minimize the computation time required in searching the drop profile coordinates, a 2 × 2 or 3 × 3 edge operator is preferable over a higher-order one. Since the drop has more approximately diagonal edges than vertical edges, the Sobel edge operator is utilized in ADSA. To extract the drop profile coordinates, the output of the Sobel operator may be thresholded to produce a binary image. If the output at a pixel exceeds a certain threshold, the pixel is interpreted as a drop profile coordinate and given a value of 255; otherwise, it is given a value of 0. Unfortunately, the histogram of the output is usually unimodal, such that it is difficult to find a suitable threshold.
As a result, the drop profile coordinates identified may be several pixel wide. Another alternative is to follow the local maxima of the Sobel operator output. This accomplished by a computer program, written in FORTRAN, which finds the drop profile coordinates automatically. The details may be found in reference [16].

**A.3 Correction of Optical Distortion**

Most cameras and microscopic lenses produce slightly distorted images, and this distortion can cause major errors in the final results, particularly in the interfacial tension. To correct the optical distortion, an image of a calibration grid pattern on an optical glass is taken at the same position as the drops. The grid is used for both correction of the optical distortion and calibration. The image of the grid is established by the coordinates of the intersection points of the horizontal and vertical grid lines. Each intersection point is calculated from the intersection of two perpendicular lines formed by “darkest” pixels, i.e., the pixels of lowest grey level, on the nearby horizontal and vertical grid lines. Each grid line is approximately 1 pixel wide. This distorted grid can then be mapped onto the known undistorted grid, using the method described in reference [16].

**A.4 Drop Profile Analysis to Subpixel Resolution**

As described in the last section, both edge detection and correction of optical distortion depend on the selection of a nearest pixel. The total error in each drop profile coordinate can be expected to be 1 to 2 pixels. For a sessile drop of 5 mm in contact diameter, this corresponds to an error of 25–50 μm. To further improve the precision of the results, a digitizing board with
higher resolution, e.g., 1024 × 1024 pixels, could be used, although the cost and computation time would increase. The other alternative is the application of a subpixel resolution algorithm. This can be achieved by fitting a grey-level profile perpendicular to the drop interface with the so-called natural cubic spline fit. The end condition for the natural cubic spline fit is that the second derivative equals zero at each end, since the grey-level profile across the drop interface should approach linearity at each end. The natural cubic spline fit can also be applied in conjunction with correction of the optical distortion to improve the accuracy. The details of natural cubic spline fit method may be found in reference [16].

For the digitized drop image, it is generally impossible to fit a natural cubic spline curve exactly perpendicular to the drop interface using the existing grey-level values. This fitting can only be done in three directions: horizontal, vertical, and diagonal. Hence, the direction closest to the perpendicular is chosen from this set of three principal directions, and the natural cubic spline fit for the grey levels is calculated in this preferred direction, for each drop profile point.
Figure A.1 Comparison between experimental points and a Laplacian curve. The curve $u=u(s)$ is a theoretical profile based on the Laplace equation of capillarity, while points $U_i$, $i=1,2,...,N$, are points selected from the meridian of an experimental drop profile. The deviation of the $i^{th}$ point from the Laplacian curve, $d_i$, can be calculated. The purpose of objective function of ADSA is to minimize the sum of the square of the minimum distance, $d_i$. Since the coordinate systems of experimental profile and the predicted Laplacian curve do not necessarily coincide, their offset $(x_0, z_0)$ and rotation angle ($\alpha$) must be considered. Both the offset $(x_0, z_0)$ and the rotation angle ($\alpha$) are optimization parameters. Since the program does not require the coordinates of the drop apex as input, the drop can be measured from any convenient reference frame, and all measured points on a drop profile are equally important.
Figure A.2 Schematic diagram of the experimental setup for analysis of sessile-drop and pendant-drop systems by ASDA UNIX version
APPENDIX B

VIDEO SIGNALS AND INPUT CONTROLS OF THE FRAME GRABBER

The video signal includes all parts of the signal including non-visual portions such as horizontal and vertical blanking. For an interlaced signal, the video signal is defined as two consecutive fields, where the start of each field is identified by the falling edge of the vertical sync. For a non-interlaced signal, the video signal is defined as a signal field, where the start of the field is identified by the falling edge of the vertical sync. For both interlaced and non-interlaced video, a horizontal line of video is defined by the falling edge of the horizontal edge of the horizontal sync, and a field is composed of a collection of horizontal lines.

Many of the input controls of the frame grabber manipulate the location or interpretations of the various transitions within the incoming video signal. These transitions identify, among other things, the active video area. The active video area is defined as that part of the incoming signal that contains valid video data (i.e., not blanking or sync information). Therefore, the video active area consists of the visible portion of those lines containing visible pixel data. The controls that define the video area deal with pixels and lines of video. Pixels are available to the frame grabber at increments of Pixel Period. They may also be thought of as increments of time. Pixel measurements are relative to the horizontal reference point (Figure B.1), which is defined as the beginning of the horizontal sync. Line measurements are relative to the vertical reference point (Figure B.2), which is defined as the beginning of the vertical sync. Shown in Figure B.1, each
line of video contains horizontal sync information, blanking (the suppression of video information), and active video.

![Diagram of Horizontal Video Signal](image)

**Figure B.1 Horizontal Video Signal**

![Diagram of Vertical Video Input Control](image)

**Figure B.2 Horizontal Input Control**

In the horizontal video signal, blanking occurs during the horizontal sync and image border periods, which are defined by the front porch (before the horizontal sync) and back porch (after the horizontal sync). The back porch can be changed between 0 to 4095 pixels (nominal is 60 pixels for 60 Hz image formats and 80 pixels for 50 Hz image format). Figure B.2 shows the
relationship between the above signal and the setting of the frame grabber’s horizontal input controls.

On the other hand, each field of the video contains vertical information, blanking information, and lines of active video. Figure B.3 and Figure B.4 show the components of a signal vertical field of video and the relationship between the incoming video signal and the setting of the frame grabber’s vertical input controls respectively.

![Figure B.3 Vertical Video Signal](image)

**Figure B.3 Vertical Video Signal**

![Figure B.4 Vertical Video Signal](image)

**Figure B.4 Vertical Video Signal**
APPENDIX C

CODES OF IMPORTANT FUNCTIONS IN THE ADSA
PC-VERSION

C.1 Serialize() Function

This function is responsible for serializing the data required for capturing the current state of objects. The Serialize() function has an argument that is an object of the CArchive class used to read and write the object data. The object has a member function, IsStoring, which indicates whether Serialize() is storing (writing data) or loading (reading data). Therefore, the object can be directly restored from the memory. Using the Serialize() function in CExper class, all data members of this class, which are encapsulated in the Properties property sheet, can be read or written from or to an archive. The following code represents the Serialize() function in CExper class:

```cpp
void Serialize(CArchive& ar)  
{  
    if (ar.IsStoring())  
    {  
        // Storing data members  
        ar<<Shape<<Resolution<<m_Other<<m_Y<<Cubic<<m_eCut<<m_NTCentreImage<<m_eScale  
        <<m_eInsideY<<m_eInsideX<<m_ProfileCoor<<m_NumIterate<<m_ComboNameFile<<Scale  
        <<m_ComboRotCoor<<m_eNumPoint<<m_ComboGrid<<m_ComboData<<m_ComboSumFile  
        <<m_eDistanGrid<<m_eHorizontLine<<m_eVerticalLine<<m_eDensity<<m_eGravity<<Output  
        <<m_NumPoint<<m_eNumIterate;  
    }  
}
```
else
{
    // Loading data members
    ar>>Shape>>Resolution>>m_Other>>m_Y>>Cubic>>m_eCut>>m_NTCentreImage>>m_eScale
    >>m_eInsideX>>m_eInsideY>>m_ProfileCoor>>m_NumIterate>>m_ComboNameFile>>Scale
    >>m_ComboRotCoor >>m_NumPoint>>m_ComboGrid>>m_ComboData>>m_ComboSumFile
    >>m_eDistanGrid >>m_eHorizontLine>>m_eVerticalLine>>m_eDensity>>Output>>m_eGravity
    >>m_eNumPoint>>m_eNumIterate
}

C.2 OnImageDeviceid() Function

The program searches for all frame grabber boards in the system, each of which is identified by a globally unique name. After selecting the Open Device submenu item, the names of the frame grabbers are appeared in the combo box of the Open Device dialog box. While a name is selected, the OnImageDeviceid() function opens the corresponding frame grabber board and retrieves its properties including the memory size, Device ID, and type of the frame grabber board. It is noted that only a single frame grabber board can be selected at one time. The code of this function is as follow:

```c
void OnImageDeviceid()
{
    dev.m_CbAlias=frmgrb.m_CbAlias;
    if(dev.DoModal()==IDOK){
        frmgrb.m_CbAlias=dev.m_CbAlias;
        if(!OImgIsOkay(OImgOpenDevice(frmgrb.m_CbAlias, &DevId)))
            MessageBox(0,"Device not found","Error",MB_OK);
        frmgrb.DevId= DevId;
        frmgrb.InputSource= 0;
    }
}
```
C.3 OnImageAcquire() Function

This function covers three acquisition modes including single, single sequential, and multiple acquisitions.

In the single acquisition, a block of memory equal to the size of one image is allocated in the host memory, and the image is acquired to the host computer. Then, the image is saved in a file with a user-defined name. In this mode, the single image is immediately displayed on the screen, and the memory is released.

if(TypeACQ==Single){
    Status = OLFgAllocateBuiltInFrame ( frmgrb.DevId,OLC_FG_DEV_MEM_VOLATILE,
                                    OLC_FG_NEXT_FRAME,&(frmgrb.FrameId));
    if ( !OImgIsOkay(Status))
        MessageBox(0,"Unable to Allocate Frame.", "Acquire Error", MB_ICONSTOP | MB_OK);
    Status = OLFgAcquireFrameToHost(frmgrb.DevId, frmgrb.FrameId, frmgrb.hpAcquireBuf,
                                    frmgrb.ulMinBufSize);
    if ( !OImgIsOkay(Status)){
        MessageBox(0,"Unable to acquire.", "Acquire Error", MB_ICONSTOP | MB_OK);
        return;
    }
    fwrite(frmgrb.hpAcquireBuf,frmgrb.ulMinBufSize,1,handle);
    fclose(handle);
    frmgrb.CreateDIBDisplay(frmgrb.hpAcquireBuf, frmgrb.ulHeight, frmgrb.ulWidth, frmgrb.ulPixelDepth);
    (void) OLFgDestroyFrame(frmgrb.DevId,frmgrb.FrameId);
    if ( !OImgIsOkay(Status))
        MessageBox(0,"Unable to free frame after acquire.", "Acquire Error", MB_ICONSTOP | MB_OK);
    fprintf(filename,"\n",m_eFileName);
    fclose(filename);
}
In the single sequential acquisition, the same process is repeated for a number of times. The number of iterations is defined based on the duration of the acquisition and interval between two immediate images. In addition, the images are saved in separate files, each of which is entitled with the acquisition time attached to a user-defined string. Then, each image may be displayed using Open Image submenu item, but the memory is immediately released when the images are saved.

```c
if(TypeACQ==SingleSequential)
{
    TotalImages=m_eDuration/m_eInterval;
    CProgress prog;
    if(prog.Create(IDD_PROCESSNULL))
    {
        prog.m_ProgressBar.SetRange(0,TotalImages);
        _timeb* start=new _timeb[TotalImages];
        _timeb* finish=new _timeb[TotalImages];
        for(int i=0; i<TotalImages ; i++)
        {
            prog.m_ProgressBar.SetPos(i);
            first=clock();
            Status = OIFgAllocateBuiltInFrame ( frmgrb.DevId,OLC_FG_DEV_MEM_VOLATILE, OLC_FG_NEXT_FRAME,&(frmgrb.FrameId));
            _ftime(&start[i]);
            Status = OIFgAcquireFrameToHost(frmgrb.DevId, frmgrb.FrameId, frmgrb.hpAcquireBuf, frmgrb.ulMinBufSize);
            _ltoa( start[i].time, m, 10 );
            _ltoa( start[i].millitm, n, 10 );
            _sctv(atof(strcat(strcat(m,"."),n))-dfstart, 7, m);
            m_eFileName.Insert(acq.m_eFileName.GetLength(),m);
            m_eStrFile[i]=m_eFileName;
            handle = fopen(m_eFileName,"wb");
            fwrite(&frmgrb.image.sizeof(IMG),1 ,handle);
            fwrite(frmgrb.hpAcquireBuf,frmgrb.ulMinBufSize,1,handle);
            fclose(handle);
            (void) OIFgDestroyFrame( frmgrb.DevId, frmgrb.FrameId);
            m_eFileName=HELP;
    }
```
In the multiple acquisitions, a block of memory equal to the device memory is allocated. First, the images are acquired into the device memory. While the device memory becomes full, the images are transferred to the host memory so that the device is released to continue the acquisition. Thus, the maximum number of images acquired without interruption depends on the size of the device memory. In this mode, images can be saved in either one file or separate files. It is noted that, the ADISA program requires separate image files, each of which is entitled with the acquisition time attached to a user-defined string. Then, the images may be displayed using Open Image submenu.

```c
if (TypeACQ==Multiple) {
    for (i=0; i<TotalImages/m_eImgInterapt ; i++) {
        for(int j=0;j<m_eImgInterapt;j++)
            Status = OIFgAllocateBuiltInFrame (frmgb.DevId,OLC_FG_DEV_MEM_VOLATILE,
            OLC_FG_NEXT_FRAME,&(frmgb.FrameList[j]));
            _ftime(&start[i]);
            Status = OIFgAcquireMultipleToDevice(frmgb.DevId,m_eImgInterapt,frmgb.FrameList);
            _ftime(&finish[i]);
}
```
for( int k=0 ; k<m_eImgInterapt ; k++){
    Status=OLFgReadFrameRect(frmgrb.DevId,frmgrb.FrameList[k],0,0,frmgrb.ulWidth,
    frmgrb.ulHeight,frmgrb.hpAcquireBuf,(frmgrb.ulWidth)*(frmgrb.ulHeight)*
    (frmgrb.ulPixelDepth));
    fwrite(frmgrb.hpAcquireBuf,frmgrb.ulMinBufSize,1,handle);
    Status = OLFgDestroyFrame(frmgrb.DevId,frmgrb.FrameList[k]);
}
}
i=TotalImages % m_eImgInterapt;
if(~i){
    for(int j=û;j<i;j++)
        Status = OLFgAllocateBuiltlnFrame (frmgrb.DevId,OLC_FG_DEV_MEM_VOLATILE,
        OLC_FG_NEXT_FRAME,&(frmgrb.FrameList[j]));
    _ftime(&rstart);
    Status = OFgAcquireMultipleToDevice(frmgrb.DevId,i,frmgrb.FrameList);
    _ftime(&rfinish);
    for(int k=0 ; k<i ; k++){
        Status=OLFgReadFrameRect(frmgrb.DevId,frmgrb.FrameList[k],0,0,frmgrb.ulWidth,
        frmgrb.ulHeight,frmgrb.hpAcquireBuf,(frmgrb.ulWidth)*(frmgrb.ulHeight)*
        (frmgrb.ulPixelDepth));
        fwrite(frmgrb.hpAcquireBuf,frmgrb.ulMinBufSize,1,handle);
        Status = OLFgDestroyFrame(frmgrb.DevId,frmgrb.FrameList[k]);
    }
}
fclose(handle);
MessageBox(0,"Images were acquired",".MB_OK);
}

C.4 OnProperties() Function

When the Properties property sheet is closed with OK pushbutton, the values of CExper
data members are saved in an input file entitled “pendant.in” and “sessile.in” for pendant drop
and sessile drop experiments, respectively. The input files are required for the ADSA program
while processing the images.
void OnProperties()
{
    FILE *handle=NULL;
    FILE *filename;
    CSheet sheet("Properties");
    sheet.Add_Pages(exp);
    handle = fopen("pendant.in","wt");
    if(sheet.DoModal()==IDOK){
        exp=sheet.exp1;
        fprintf(handle,"\n\n%d\n%d\n%s\n%hO\n",exp.Resolution,exp.Shape,exp.m_ComboGrid,
            exp.m_eDistantGrid);
        fprintf(handle,"\n\n%d\n%d\n%s\n%hO\n",exp.m_ComboData,exp.m_eHorizontLine,
            exp.m_eVerticalLine,exp.m_Other,exp.m_ComboRotCoor,exp.m_Y);
        if(exp.m_Cut)
            fprintf(handle,"-1:\n");
        else
            fprintf(handle,"-1:\n");
        fprintf(handle,"\n\n%d\n",exp.m_ProfileCoor,exp.m_NTCentreImage);
        if(exp.m_NTCentreImage==0)
            fprintf(handle,"\n%d\n",exp.m_eInsideY,exp.m_eInsideX);
        fprintf(handle,"\n\n%d\n",exp.m_eDensity);
        if(exp.m_NumIterate)
            fprintf(handle,"\n%d\n",exp.m_eNumIterate);
        else
            fprintf(handle,"-1\n");
        if(exp.m_NumPoint)
            fprintf(handle,"\n%d\n",exp.m_eNumPoint);
        else
            fprintf(handle,"-1\n");
        fprintf(handle,"\n-1\n",exp.m_eGravity,exp.Scale);
        if(exp.Scale==3)
            fprintf(handle,"\n%d\n",exp.m_eScale);
        fprintf(handle,"\n\n%d\n",exp.Output);
        if(exp.Output==2)
            fprintf(handle,"\n\n%d\n",exp.m_ComboSumFile);
        if(filename=fopen(exp.m_ComboNameFile,"rt")){
            fscanf(filename,"\n%d",&TypeACQ);
            fscanf(filename,"\n%d",&TotalImages);
            if(TypeACQ==3){
                fscanf(filename,"\n\n%s\n",m_eFileName);
                fprintf(handle,"\n\n%s\n",m_eFileName);
            }
            else{
                for(int i=0; i<TotalImages;i++){
                    fscanf(filename,"\n\n%s\n",m_eStrFile[i]);
                    fprintf(handle,"\n\n%s\n",m_eStrFile[i]);
                }
                fclose(filename);
            }
        }
        else{
            if(MessageBox(0,"There is no file containing the name of image(s) or this file does not exist.
            Do you want to process the last opened image?","Warning",MB_ICONWARNING | MB_YESNO )==IDYES)
            return;
        }
    }
}
fprintf(handle,"%s\n",m_eFileName);
else
    MessageBox(0,"Please type the name of images in tasesile.in","Warning",MB_ICONWARNING | MB_OK);
}
fclose(handle);

C.5 OnImageOpen() Function

This function is invoked from Open Image dialog box to display the images saved in the image files. For the images acquired via single and single sequential acquisition modes, the filenames are sufficient; however, for images acquired via multiple acquisition mode, the image number must also be specified. In addition, two functions have been programmed to increment and decrement the image number to show the next and the last images, respectively.

void OnImageOpen()
{
    COpenImg open(TRUE,NULL,NULL,OFN_HIDEREADONLY | OFN_OVERWRITEPROMPT | OFN_ALLOWMULTISELECT, "(*.img)/*.img|All Files(*.*)|*.img|*.img*.img*",NULL);
    FILE *handle;
    CDisplay dis;
    if(open.DoModal()==IDOK){
        handle=fopen((const char *)open.GetFileName(),"rb");
        m_eFileName=open.GetFileName();
        dis.m_eNumPic=m_eNumPic;
        if(dis.DoModal()==IDOK){
            m_eNumPic = dis.m_eNumPic;
            fread(&(frmgrb.image).sizeof(IMG),1,handle);
            frmgrb.ulWidth=frmgrb.image.width;
            frmgrb.ulHeight=frmgrb.image.height;
            frmgrb.ulPixelDepth=1;
            fseek(handle,(m_eNumPic-1)*(frmgrb.ulWidth)*(frmgrb.ulHeight),SEEK_CUR);
C.6 OnImageCloseDevice() Function

This function closes the device to release the system resources reserved for the acquisition process. It is crucial that an application closes all open devices before termination: devices that are not explicitly closed not only use system resources but also are unavailable until the computer is restarted.

void OnImageCloseDevice()
{
  if(frmgrb.RGBDataFlag)
  {
    frmgrb.ulHeight = 0L;
    frmgrb.ulWidth = 0L;
    GlobalUnlock(frmgrb.hAcquireBuf);
    GlobalFree(frmgrb.hAcquireBuf);
    frmgrb.hAcquireBuf = NULL;
    frmgrb.hpAcquireBuf = NULL;
  }
  if (frmgrb.DevId != NULL)
    (void) OImgCloseDevice(frmgrb.DevId);
  memset((LPVOID)&(frmgrb.CurDevCaps), 0, sizeof(DEVCAPS));
  memset((LPVOID)&(frmgrb.ImgDevInfo), 0, sizeof(OLT_IMGDEVINFO));
}
C.7 *OnRun()* Function

While all the processing inputs and images are prepared, the ADSA program may be launched. The OnRun() function relates the graphic user interface to the ADSA program, which is a console application.

```cpp
void CPendantDoc::OnPendantRun()
{
    GlobalUnlock(frmgrb.hAcquireBuf);
    GlobalFree(frmgrb.hAcquireBuf);
    char prog [50]="apendant.exe";
    if(_execl(prog,prog,NULL)==-1)
    {
        MessageBox(0,"can't Open file .try again","Error",MB_OK);
    }
}
```
REFERENCES


