Aphelion™ Imaging Software Suite

User Guide

Version 4.3
Publication revised December 1st, 2014

ADCIS SA
3, rue Martin Luther King
14280 Saint-Contest
Phone: +33 (0)2-31-06-23-00
URL: www.adcis.net

Amerinex Applied Imaging, Inc.
P.O. Box 6473
Monroe Township, NJ 08831 USA
Phone - USA: 877-664-8772
Phone - International: +1 609-944-8855
URL: www.AmerinexImaging.com

Note: While the hardcopy version of this User Guide is not updated with every release of Aphelion™ software, the electronic version found on the DVD is always current with the Aphelion software provided on the DVD. If you are unsure about a topic in the hardcopy version, please refer to the electronic version on the DVD. An electronic version of this document that is current with the most recent release of Aphelion software can be downloaded from our websites (*www.adcis.net* or *www.AmerinexImaging.com*).

Table of Contents

1. INTRODUCTION 13
2. FINDING YOUR WAY AROUND 15
2.1. Using Windows 15
2.2. Our notations 15
2.3. Product Support 16
2.4. Installing Aphelion 18
2.4.1. System Requirements 18
2.4.2. Aphelion Licensing 18
2.4.3. Aphelion Installation Prerequisites 19
2.4.4. Installing the USB Dongle 19
2.4.5. Aphelion Installation 21
2.4.6. Camera interface installation 23
2.4.7. Software required to use Aphelion 26
2.5. Starting Aphelion 27
2.6. Exiting Aphelion 27
2.7. Terminology 28
2.8. Graphical User Interface (GUI) 36
2.8.1. Workspace and windows 38
2.8.2. Task and work context 42
2.8.3. Menu Bar 42
2.8.4. Main Toolbar 70
2.8.5. Profile – Histogram – ROI Toolbar (Dev version only) 72
2.8.6. Task interface 76
2.8.7. Visualization Window 77
2.8.8. Chart Window 87
2.8.9. Image Gallery Window 89
2.8.10. ObjectSet Gallery Window 90
2.8.11. Status Bar 92

2.9. Tasks 93

2.9.1. Administrator Settings Tasks 93

2.9.2. Image Acquisition Task 98

2.9.3. Object Extraction Task 100

2.9.4. Object Editing Task 112

2.9.5. Measurements Task 118

2.9.6. Report Generation Task 133

2.9.7. Developer Task (Dev version only) 135

3. TUTORIAL 145

3.1. Multimedia demonstrations 145

3.2. Walk-through of an image processing and image analysis application 145

3.3. Interactive Analysis Tools (Dev version only) 151

3.3.1. Intensity Profile 151

3.3.2. Histogram 152

3.4. Using Developer functions & macros (Dev version only) 153

3.4.1. Select and run a function 153

3.5. Load and run a BasicScript macro 155

3.6. Basic knowledge in image processing & analysis 158

3.6.1. Processing Images 158

3.6.2. Extracting and Analyzing Objects from Images 160

4. LIST OF MEASUREMENTS 167

4.1. Global Measurements 167

4.1.1. List of Global Measurements 167

4.2. Object Measurements 168

4.2.1. Object types 168

4.2.2. List of Object Measurements 169

5. LIST OF FUNCTIONS (DEV AND SDK ONLY) 181

5.1. Images 181

5.1.1. Arithmetic 181

5.1.2. Color 181

5.1.3. Edge Detection 181

5.1.4. Enhancement *182*
5.1.5. Filtering *182*
5.1.6. Frequency *183*
5.1.7. Geometry *183*
5.1.8. Logic *183*
5.1.9. Matching *184*
5.1.10. Math *184*
5.1.11. Measurements *185*
5.1.12. Morphology *185*
5.1.13. Segmentation *189*
5.1.14. Utility *190*
5.1.15. Texture *190*
5.1.16. IO (Input/Output) *190*
5.2. ObjectSets 191
5.2.1. Display *191*
5.2.2. Chains *191*
5.2.3. Edgels *191*
5.2.4. Filtering *191*
5.2.5. Geometry *191*
5.2.6. Lines *192*
5.2.7. Measurements *192*
5.2.8. Polygons *193*
5.2.9. Bitmaps *193*
5.2.10. Utility *194*
6. OPTIONAL EXTENSIONS (DEV VERSION ONLY) 197
6.1. 3D Image Processing and 3D Image Visualization Extensions ... 197
6.2. Classification Extension 198
6.3. Color Extension 198
6.4. Hardware interfaces 198
6.5. Image Registration 199
6.6. Kriging Extension 199
6.7. Neural Network Toolkit 199

6.8. MultiFocus Extension 200
6.9. Stage Controller interface 200
6.10. Stereology Module 200
6.11. Virtual Image Capture 200
6.12. Virtual Image Stitcher 201
6.13. VisionTutor 202
7. MACRO SCRIPTING (DEV VERSION ONLY) 203
7.1. Aphelion Macros 204
7.1.1. Application Macros 205
7.1.2. Example Macros 206
7.1.3. Macro Example (Ceramic.apm) 209
7.2. C# Programming Language 224
7.3. Python 224
8. PROGRAMMING EXAMPLES (DEV AND SDK ONLY) 225
8.1. Native Examples 225
8.2. .NET Examples 226
8.3. Pads Example (Visual C# code) 227
9. .NET COMPONENTS (DEV AND SDK ONLY) 233
APPENDIX A APHELION DVD ARCHITECTURE 237
APPENDIX B REPORT GENERATION 239
B. 1. How to modify report-template 239
B. 2. Examples of small changes 241
B. 2. a. Change the Aphelion logo 241
B. 2. b. Change the report title 241
B. 2. c. Change displayed images 241
B. 2. d. Print report 241
B. 2. e. Add header/footer 242
B. 2. f. Change histogram to scatter plot 243
B. 3. Example 244
APPENDIX C TROUBLESHOOTING 245
C. 1. Troubleshooting for the dongle 245
C. 2. Troubleshooting for the camera interface 246

C. 2. a. Checking the installation of the driver *246*
C. 2. b. Checking camera operation *246*
APPENDIX D MAIN DIFFERENCES BETWEEN APHELION DEVELOPER 3.2 AND APHELION DEV 4.X 249
D. 1. Tasks 250
D. 2. Administrator rights 250
D. 3. Acquisition Task 250
D. 4. Object Extraction Task 251
D. 5. Object Editing Task 251
D. 6. Measurement Task 251
D. 7. Report Generation Task 251
D. 8. Developer Task 252
D. 8. a. Function Selection *252*
D. 8. b. Macro Languages *252*
D. 8. c. BasicScript Command window *252*
D. 9. Aphelion 4.x User Interface 252
D. 10. Image and ObjectSet Galleries 253
D. 11. Color space 253
D. 11. a. Default Color Space *253*
D. 11. b. Color Space Conversion *253*
D. 11. c. Accessing the band of a color image *254*
D. 12. Running a macro when starting Aphelion Dev 254
D. 13. Structuring Element definition 255
APPENDIX E DIFFERENCES IN PROGRAMMING SYNTAX BETWEEN APHELION 3.2 AND APHELION 4.X (DEV VERSION ONLY) 257
E. 1. Programming syntax 257
E. 2. BasicScript Editor 257
E. 3. BasicScript syntax 258
E. 3. a. Function to let the user enter function argument values during the macro execution *258*
E. 3. b. Function to retrieve argument values set during the macro execution *258*
E. 3. c. ObjectSet Attributes *259*

E. 3. d. Histogram value access 261
E. 3. e. Image list access 262
E. 3. f. Value Output 262
E. 3. g. List of non-supported functions in Aphelion 4.x 262
E. 4. Change of Syntax 262
APPENDIX F KERNEL AND STRUCTURING ELEMENT EDITOR (DEV VERSION ONLY) 263
APPENDIX G C# PROGRAMMING LANGUAGE (DEV VERSION ONLY) .. 265
G. 1. Developing an Aphelion macro in C# 265
G. 1. a. Introduction 265
G. 1. b. Referencing DLLs in a C# macro 266
G. 1. c. Developing a C# macro using an external IDE 266
G. 2. Adding a custom function to the Aphelion GUI in C# 267
G. 2. a. General conditions to be met by the macro function 267
G. 2. b. Overloads 269
G. 2. c. Function input parameters 270
G. 2. d. Function output parameters 276
G. 2. e. Conventions used for arguments display 277
G. 2. f. Additional attributes to customize the display of a parameter in the functions panel 279
G. 2. g. Additional attributes to customize the behavior of a method 298
G. 3. Loading a macro as an Aphelion plugin 300
G. 3. a. Compiling a macro as a plugin 300
G. 3. b. Load the generated DLL as a plugin 300
G. 4. Debugging an Aphelion macro 301
G. 4. a. Useful functions to debug a macro 301
G. 4. b. Using an external debugger 302
APPENDIX H COLOR SEGMENTATION 303
H. 1. Color Interval 303
H. 2. Color Distance 305
H. 3. Region Growing 306

H. 4. Morphological Partition 307
H. 5. Color Space Definitions 308
H. 5. a. YIQ color space *309*
H. 5. b. HSI color space *309*
H. 5. c. HSV color space *309*
H. 5. d. XYZ color space *310*
*H. 5. e. L*u*v* color space* *311*
H. 5. f. YUV color space *311*
*H. 5. g. La*b* color space* *312*
H. 5. h. LCH color space *312*
APPENDIX I OPTIONS→ADVANCED MENU 313

1. INTRODUCTION

IMPORTANT NOTE TO THE READER: Most sections of this User Guide are common to both the Aphelion Lab and Aphelion Dev products. In the common sections, the product referenced is simply Aphelion. In those sections not common to both products, the product referenced is either Aphelion Lab (or simply Lab) or Aphelion Dev (or simply Dev). The sections also related to Aphelion SDK are from *1 Introduction* (i.e., this section) through *2.4.5 Aphelion Installation*, and also section *8. Programming examples (Dev and SDK only)*.

This User Guide provides basic information about using Aphelion™ products, from installing its software to processing images and printing your results.

Throughout this guide, you will find references to additional information contained on the Aphelion DVD.

Aphelion's Online Help can be accessed from the **Menu bar** by choosing **Help→Help Contents**. This User Guide will open as an HTML Help.

Note: You can access all Aphelion documentation without installing the software; just look at the files in the *Help* folders on the DVD.

2. FINDING YOUR WAY AROUND

This chapter describes the installation of the Aphelion Imaging Software Suite products. It also gives a detailed description of the Graphical User Interface.

2.1. Using Windows

To use Aphelion, you should be familiar with the Windows® operating system installed on your computer (i.e., XP, Vista, 7 or 8.1) and basic Windows features such as clicking, double-clicking, and drag-and-drop editing.

Note: Unless otherwise specified, when we talk about Windows® environments in this document, we mean the Microsoft® operating systems known as XP, Vista, 7, and 8.1. Specific behaviors of these Windows® versions are explained if necessary. Aphelion Dev is available for both the 32-bit and 64-bit versions of Microsoft® Windows operating systems. Use of the 64-bit version of Aphelion is recommended if very large and/or 3D images are to be processed.

2.2. Our notations

In this document, some shortcuts are used to simplify the text. The following describes those shortcuts.

Installation Path – Installation path names will vary depending on whether a 32-bit or 64-bit installation has been performed. Many times, the only difference in a path name is the designation of either *Program Files* or *Program Files (x86)*. To avoid needless repetition, we will write *<Aphelion Installation Path>* instead of "*C:\Program Files\ADCIS-AAI*" or "*C:\Program Files (x86)\ADCIS-AAI*."

2.3. Product Support

Problem reports, comments, and enhancement requests should be submitted via the Technical Information Request form on our websites as described below:

If DVD was provided by:	Aphelion Support Team:
ADCIS SA or unknown source	*http://www.adcis.net/en/ContactUs/InformationRequestForm_type_Technicals.html*
Amerinex Applied Imaging, Inc.	*http://www.amerineximaging.com/en/ContactUs/InformationRequestForm_type_Technicals.html*

License Code requests should be submitted via the Code Request form of our websites as described below:

If DVD was provided by:	Aphelion Support Team:
ADCIS SA or unknown source	*http://www.adcis.net/en/ContactUs/CodeRequestForm.html*
Amerinex Applied Imaging, Inc.	*http://www.amerineximaging.com/en/ContactUs/CodeRequestForm.html*

Note: Technical support is handled only in an electronic way. Support messages must include 1) the name of the company or organization to which the software license is granted, 2) the email address of the person requesting support, 3) a phone number that can be called by our customer support team to ask for more technical information about the problem, 4) the License Code and version number of the software, 5) the version of the Windows operating system, and 6) a detailed description of the problem with the list of operations to call to demonstrate the problem.

For additional technical help, visit *www.adcis.net/en/Support/Login.html* or *www.amerineximaging.com/en/Support/Login.html*, and then enter the following information:

Username: Name issued to you when you first requested support.

Password: Password you entered when you first registered.

License Code: This code is available in Aphelion's Help→About window.

Support is available at the following locations:

In France:

ADCIS SA
3, rue Martin Luther King
14280 Saint-Contest
France
Fax: +33 (0)2-31-06-23-09
Contact: *http://www.adcis.net/en/ContactUs/index.html*

In the USA:

Amerinex Applied Imaging, Inc.
P.O. Box 6473
Monroe Township, NJ 08831
USA
Fax: +1 609-944-8855
Contact: *http://www.amerineximaging.com/en/ContactUs/index.html*

For other locations, please contact your Aphelion representative. A list of official representatives is available on our websites at:

http://www.adcis.net/en/ContactUs/WhereToBuy.html

or

http://www.amerineximaging.com/en/ContactUs/WhereToBuy.html

2.4. Installing Aphelion

2.4.1. System Requirements

Aphelion requires the following minimum system configuration:

- ✓ Windows® XP, Vista, 7 or 8.1 operating system;
- ✓ Windows® 32-bit or 64-bit operating system;
- ✓ Super VGA monitor (1024x768 resolution, 65,536 colors);
- ✓ Intel® Pentium® 4 processor, or equivalent;
- ✓ 2 GB of RAM;
- ✓ Hard drive with at least 300 MB free space;
- ✓ A network interface adapter supporting the Ethernet® protocol and compatible with NetBIOS®;
- ✓ A local or network DVD drive (for software installation).

2.4.2. Aphelion Licensing

The licensing mechanism is based on the network adapter of the computer on which Aphelion software is being installed. Refer to the sections *2.4.3 Aphelion Installation Prerequisites* and *2.4.5 Aphelion Installation* to learn more about the installation procedure.

An evaluation version of Aphelion is time limited. The duration of the evaluation period is computed based on the date the evaluation License Code is actually issued. For a standard 15-day evaluation period, if the installation happens 5 days after receiving the License Code, then Aphelion will run during 10 days. If Aphelion has not been installed during the period of 15 days, call your Aphelion software supplier and ask for a new License Code. To install the evaluation version, follow the instructions provided in the sections *2.4.3 Aphelion Installation Prerequisites* and *2.4.5 Aphelion Installation.*

Evaluation and permanent installations of Aphelion software involve two coded character strings: an Identification Code that pops up during installation and a License Code that is typed in during installation. Each pair of codes is unique for a given installation. They comprise a sequence of alphanumeric character groups, usually four characters per group, such as: 1S12-129J-RM7W-LG4D-LH4S-PEF0-YA33, except the first group of the identification Code has only two characters.

License Codes are provided via return electronic mail from our Support Staff. A License Code is sent in response to our receipt of a completed Code Request form sent from our website (see *2.3 Product Support*), by the requester. The Code Request form must include the Identification Code generated during the installation.

Please be sure that your computer system meets the conditions described in sections *2.4.1 System Requirements* and *2.4.3 Aphelion Installation Prerequisites*. If it does, then follow the instructions given in section *2.4.5 Aphelion Installation* to install Aphelion on your system.

2.4.3. Aphelion Installation Prerequisites

Aphelion can only be installed on a computer equipped with either a properly configured network adapter (non-virtual adapter) or with an optional Aphelion USB dongle. If your computer does not have an appropriate network adapter, please contact your Aphelion supplier for information about purchasing an Aphelion USB dongle.

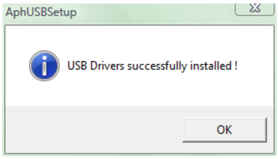
2.4.4. Installing the USB Dongle

The Aphelion USB dongle is an optional device that provides users with two capabilities. The dongle enables installation of Aphelion on a computer that does not have a suitable network adapter (see the section immediately above). The dongle can also be used to enable non-concurrent execution of Aphelion on multiple computers simply by moving the dongle from one computer to another.

Note: Only a single copy of the software can be used at a time; the dongle must always be connected to the computer on which the Aphelion license is used.

DO NOT perform the following procedure unless your Aphelion installation will be using an Aphelion USB dongle.

1. DO NOT connect the dongle to the USB port until Step 4 (i.e., not before the complete installation of the driver).
2. Open a Windows session with Administrator rights.
3. Run the program ***\Device Drivers\Dongle\AphUSBSetup.exe*** which can be found on the Aphelion DVD. This step will install the dongle's driver and end by displaying the window shown below.


AphUSBSetup

USB Drivers successfully installed !

Note for Windows 7: The Program compatibility assistant window opens. Select the option "*This program installed correctly*," then click the **OK** button to complete the installation.

4. Connect the dongle to your computer. After a few moments the wizard Found New Hardware opens.
5. Select the option "*Install the hardware automatically (recommended)*" then click the **Next** button. Hardware device detection can take a few minutes. Note: Windows XP and Vista will display the warning window "*The software that you are installing…was not completed.*" Click on the button **Continue...** to complete the installation.
6. The final window of the wizard Found New Hardware opens. Click on **Finish** to close this window. The popup window CBUSB Ver 2.0, Device driver software installed successfully appears above the Windows desktop icon tray. In case of trouble, refer to *Troubleshooting for the dongle* in section *APPENDIX C*.

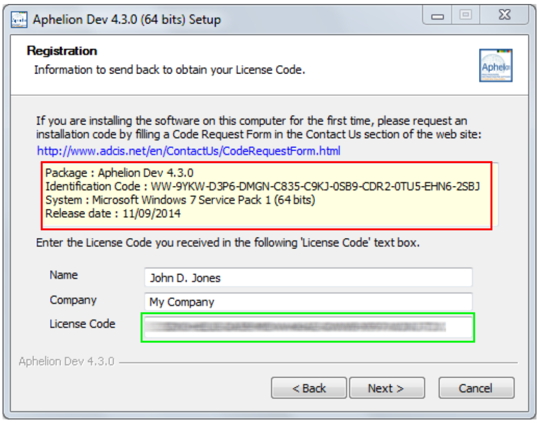
2.4.5. Aphelion Installation

Important note: Be sure to use a Windows session with Administrator rights to install the software. If you have doubt about your rights, contact your local system administrator.

Tip: Your computer must have at least 300 Mbytes of free hard drive space for error-free extraction of the Aphelion installation files. If the *C:* partition of the hard drive has fewer than 300 Mbytes of free space, then the files extraction can be performed in another partition that does have enough free hard drive space. To use a larger partition, create a temporary folder in this partition (e.g., *<selected partition>:\tmp*) as follows:

1) Click the **Start** button, then select the ***Run…*** program; 2) Type "*cmd*" in the Open field and left-click the **OK** button; 3) At the cmd prompt, type "*set tmp=<selected partition>:\tmp*" and press the **Enter** button; and 4) At the cmd prompt, type "*Exit*" to close the command window.

1. If you received the Aphelion software on a DVD, please assure that the DVD is in the DVD drive.
2. Execute the installation program ***\AphelionDev\AphelionDev Setup 4.x.y (32 bits).exe*** or ***\AphelionDev\AphelionDev Setup 4.x.y (64 bits).exe*** for Dev or ***\AphelionLab\AphelionLab Setup 4.x.y.exe*** for Lab. This file can be found in the corresponding folders of the DVD if you received the software on a DVD, or on your hard drive if it was downloaded from the website of ADCIS S.A. or Amerinex Applied Imaging, Inc.
3. Follow the instructions as they appear on the screen during the installation.
4. After you have read and accepted the End User License Agreement, the Registration window appears on the screen. The software being installed, its version, its release date, the operating system, and the Identification Code will all be displayed in the


Aphelion Dev 4.3.0 (64 bits) Setup

Registration
Information to send back to obtain your License Code.

If you are installing the software on this computer for the first time, please request an installation code by filling a Code Request Form in the Contact Us section of the web site:
http://www.adcis.net/en/ContactUs/CodeRequestForm.html

Package : Aphelion Dev 4.3.0
Identification Code : WW-9YKW-D3P6-DMGN-C835-C9KJ-0SB9-CDR2-0TU5-EHN6-2SBJ
System : Microsoft Windows 7 Service Pack 1 (64 bits)
Release date : 11/09/2014

Enter the License Code you received in the following 'License Code' text box.

Name	John D. Jones
Company	My Company
License Code	[redacted]

Registration window (see red box above). If you are installing the current version of Aphelion for the first time, please copy (e.g., **Ctrl+C**) the four text lines labeled *Package*, *Identification Code*, *System*, and *Release Date* found in the Registration panel (see above graphic), and paste (e.g., **Ctrl+V**) those lines into the *ContactUs/CodeRequestForm* page on our websites (see section *2.3 Product Support*).

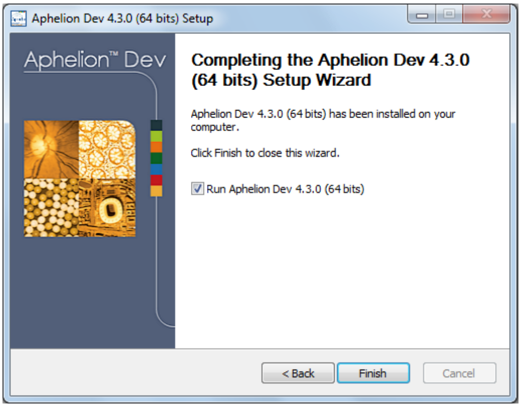
5. By return email, within one business day, the Aphelion Support Team will send you the License Code needed to complete your installation of Aphelion (see green box above).

6. Fill in the *Name*, *Company*, and *License Code* fields using the License Code you received from the Support Team. If this is a reinstall, use the License Code you received the first time you installed the current version of Aphelion.

7. Select the folder in which Aphelion is installed. Note that the default folder is *<Aphelion Installation Path>\Aphelion 4.x.y\Dev* for Dev and *<Aphelion Installation Path>\Aphelion 4.x.y\Lab* for Lab, where the x.y characters stand for the release number. By keeping this default folder structure, you will install each new Aphelion release into a folder whose name will correspond to the version number. You will then be able to execute a previous version at any time.

Note: The License Code depends on the computer configuration and on the Aphelion version. If you try to install an update version of Aphelion on the same computer as a previous installation and the License Code is not accepted, then contact your Aphelion representative or the Aphelion Support Team to request a License Code for this newer version.

8. Continue the installation until the window Completing the Aphelion Dev 4.x.y Setup Wizard for Dev or Completing the Aphelion Lab 4.x.y Setup Wizard for Lab pops up, and then click on the **Finish** button to complete the installation.



Completing the Aphelion Dev 4.3.0 (64 bits) Setup Wizard

Aphelion Dev 4.3.0 (64 bits) has been installed on your computer.

Click Finish to close this wizard.

[✓] Run Aphelion Dev 4.3.0 (64 bits)

2.4.6. Camera interface installation

IEEE1394 Compatible Cameras

Important note: The interface for IEEE1394 cameras, compliant with the IIDC version 1.31 specifications, is only available in the 32-bit version of Windows.

If you have an IEEE1394 compatible camera that complies with the IIDC version 1.31 specifications, then you will be able to interface this camera to Aphelion by installing the included *CMU 1394 Digital Camera Device* driver. After installation of Aphelion, install this driver by proceeding as follows:

1. If Aphelion is running, you must exit from Aphelion before connecting the camera. After Aphelion has been closed, connect the camera to the computer using a FireWire cable. Windows will automatically install a generic driver for the camera that you just connected. By default, the driver is the *Generic 1394 Desktop Camera* driver.

Note: If your computer does not have a FireWire port, install a FireWire adapter card compatible with the IEEE1394a or IEEE1394b protocols, depending on the protocol supported by your camera. Please refer to the documentation provided with your adapter card to perform this installation.

2. Open the Windows Device Manager window.
3. Open the *Imaging devices* item and select the *Generic 1394 Desktop Camera*.
4. Right-click the selected device to open its contextual menu, and choose *Update Driver…*
5. After the Hardware Update Wizard opens, select the *Install from a list or specific location (Advanced)* radio button, and then click **Next**.
6. Select the *Don't search. I will choose the driver to install* radio button, and then click **Next**.
7. Click the **Have Disk…** button.
8. Click on the **Browse** button and move to *<Aphelion Installation Path>Aphelion 4.x.y\Dev\Drivers\IEEE1394* for Dev or *<Aphelion Installation Path>Aphelion 4.x.y\Lab\Drivers\IEEE1394* for Lab, then select the file named *1394Camera.inf*, click the **Open** button, and then click the **OK** button of the Install From Disk panel and finally click the **Next** button of the Hardware Update Wizard panel.
9. If the warning message "*The software you are installing for the hardware: CMU 1394 Digital Camera Device has not passed Windows Logo testing…*" pops up during the installation, click the **Continue Anyway** button.

10. Click on the **Finish** button to complete installation of the selected IEEE1394 driver (i.e., *CMU 1394 Digital Camera Device*) that is now available in the *Imaging devices* section of the Windows Device Manager.
11. Run the ***Aphelion*** software. Camera settings are automatically loaded and camera functions can then be controlled through the Aphelion interface.

If the camera device is not accessible in Aphelion (see *2.9.1 Administrator Settings Tasks*), please read *Troubleshooting for the camera interface*the in section *APPENDIX C*.

If the camera is working correctly in the demo program (***1394CameraDemo.exe***) but not in the Aphelion program, then please contact the Aphelion Technical Support via email (see section *2.3 Product Support*).

DirectShow Compatible Cameras

Aphelion can also control a DirectShow compatible camera. The following procedure checks that the camera interface is working properly:

1. If Aphelion is running, you must exit from Aphelion before connecting the camera. After Aphelion has been closed, connect the camera to the computer using the cable provided with the camera. Windows automatically installs the driver associated with the camera.
2. Open the Windows Device Manager applet.
3. Open the *Imaging devices* group and verify that the camera is listed in this group. The driver installation may take a few minutes.
4. Run the ***Aphelion*** software. The camera settings are automatically loaded and camera functions can be controlled from the Aphelion interface.

Third party cameras

Extension interfaces for third party cameras are available in Aphelion Dev. To install one of the camera drivers, refer to the associated installation guide located in the \Device Drivers folder that can be found on the Aphelion Imaging Software Suite DVD.

2.4.7. Software required to use Aphelion

Microsoft Visual C++ 2013 Redistributable Package and Microsoft .NET Framework 4.0 are required to run Aphelion software. If not already installed on your computer, these packages will be installed automatically during the Aphelion installation process. The absence of either of these Microsoft packages will prevent Aphelion from executing properly.

Note: Future versions of Aphelion software may require newer versions of these required Microsoft products. Be sure to install the newer versions when asked to do so during the Aphelion software installation process.

Software capable of executing AVI files is required to run the multimedia demonstrations that show how to use Aphelion. These demonstration files are included on the Aphelion DVD in the folder, *\Multimedia Demos*.

2.5. *Starting Aphelion*

Once Aphelion is installed, start the application by clicking on the appropriate **Aphelion** icon (Dev or Lab) located on the desktop, or by clicking on **Start→All Programs→Aphelion Dev (or Lab) 4.x.y**. While the application is loading, a splash screen pops up and displays copyright information and the software version number.

This program is protected by U.S.A. and international laws as described in the program's Help About.

After a few seconds, the splash screen vanishes and Aphelion's main graphical user interface ("GUI") opens.

2.6. *Exiting Aphelion*

To quit Aphelion, select **File→Exit**. If loaded images have been modified during the work session, or if new images have been created, then a Save Files dialog box opens. The user selects the images and ObjectSets to be saved, then clicks on the **Save** button. If no files should be saved, the user clicks on the **Don't save** button.

2.7. Terminology

To facilitate your use of Aphelion, please become familiar with the following terms. You may find it helpful to refer to *Figure 2.8-1* in the following section.

Graphical User Interface ("GUI") – The Aphelion GUI is divided into two regions: the Workspace and the **Menu Bar** plus toolbars. Each of these is further divided into sub regions, and is described in more detail below. *Figure 2.8-1* shows the Workspace sub regions (i.e., windows) as rectangles outlined in blue and the menu plus toolbars sub regions as rectangles outlined in red.

Menu Bar plus Toolbars – This GUI region contains two rectangular areas in which the **Menu Bar** and various toolbars are displayed. The **Menu Bar** and its associated toolbar appear in the rectangular area at the top of the GUI. The rectangular area at the far right of the GUI contains the selected task's **Contextual toolbar** (see detailed explained below).

Workspace – The Workspace is a fixed region of the Aphelion GUI that contains multiple windows. The number of windows and their functions is determined by the current step in the image analysis process (i.e., the Active Task). While each window has a default size and location within the GUI, their sizes and locations can be altered by the user. An example of the default Workspace layout is shown in *Figure 2.8-1*.

Windows – The Workspace windows are user interface structures that provide information and controls used in the image analysis process. There are two categories of such interfaces: display interfaces and control interfaces.

Display Interface – This interface type provides on-screen representations of image and ObjectSet data. Image data includes full images and thumbnail images displayed as pixels arrays. ObjectSet data (information extracted from a source image) consists of spatial attributes (e.g., Bitmap, Chain, Line) and a set of numeric attributes (e.g., area, length, max and min pixel intensity) for each object in the ObjectSet. Each object's spatial attribute is rendered in a graphics overlay in the Visualization window, generally on top of the ObjectSet's source image, although another image can be selected. ObjectSet data can also be displayed in the form of grids (row-column arrays) and charts (histograms and scatter plots). When needed, these ObjectSet representations can be viewed concurrently in different Display Interfaces (i.e., Visualization, Grid, and Chart).

Control Interface – This interface type is the means by which a user selects functions and specifies parametric values that apply to an image, graphic object, measurement, or image capture device.

Task – Aphelion divides the image analysis process into six distinct steps for Lab and seven steps for Dev called tasks. These are named **Camera settings**, **Calibration settings**, **Image Acquisition**, **Object Extraction**, **Object Editing**, **Measurements**, and **Report Generation**. A task named **Developer** is available in the Dev version. A task is selected by clicking its icon on the **Task Bar**. The first two tasks are only available to users with administrator rights. When the user selects a task, the Workspace is automatically reconfigured to provide the Display and Control Interfaces appropriate for the selected task, including the corresponding **Contextual Toolbar** displayed on the right side of the GUI.

Task Bar Control Interface – This control interface displays the icons for the tasks used to perform an image analysis and to define the settings.

Visualization window – This window contains the Visualization Display Interface, the vehicle through which images and ObjectSets can be displayed. The user can choose to display either one image (i.e., a 1x1 array) or multiple images as an MxN array of images, where M and N are Administrator defined parameters via Tools→Options →Advanced→Images (see section *2.8.4 Main Toolbar*). A set of image manipulation tools exists on the **Main Toolbar** to perform image zooming and panning. These include **Fit to Window**, zoom to Actual Size, **Zoom in area of interest**, **Zoom In**, **Zoom Out**, and **Pan**. Tools are also provided for drawing or deleting graphic objects in a graphics overlay. This window can also concurrently display an ObjectSet as an overlay on the image from which it was extracted. The Visualization window has some unique properties not present with the other Workspace windows (e.g., it is always visible).

Image Gallery – The Image Gallery displays a thumbnail version of each image currently open in a work session, along with the image's name. Selecting (double-clicking on) a thumbnail image in the Image Gallery makes it the Active Image by displaying it in the Visualization window.

Active Image – Any open image that appears in the Visualization window and that has the focus as noted by the image's icon being highlighted in the Image Gallery. The Active Image is always the default target image for image processing functions executed from the **Main Toolbar**, a **Contextual Toolbar**, or the **Developer** Task.

Object – An object is a grouping of adjacent pixels in an image (e.g., blob, line, chain). The objects derived from a single, source image compose an ObjectSet. A list of object types is specified in the Tools→Options→Measurements window. More information is provided about objects in subsequent sections.

ObjectSet – A set of objects derived from a single, source image. Four tasks can be used to create objects: the **Object Extraction** Task, the **Object Editing** Task's interactive editing tools, the **Measurements** Task's interactive drawing tools, and the **Developer** Task's segmentation tools (Dev version only). When measurements are computed for an object's attributes, they are stored with their respective object in its ObjectSet. An ObjectSet is visualized as a graphics overlay on its source image. ObjectSets can be saved as .aso (recommended) or .tks (for backwards compatibility with previous versions of Aphelion).

ObjectSet Gallery – For each ObjectSet currently opened in a work session, the ObjectSet Gallery displays a common thumbnail (see adjacent) labeled with the ObjectSet's name. A newly created ObjectSet appears in the ObjectSet Gallery with a default name that is the source image name appended with the text (n) in case of multiple instances of the same ObjectSet, (e.g., SourceImageName (2)) when there are multiple ObjectSets derived from the same image. If these ObjectSets are saved to storage media, the extension .aso is appended to the default name (e.g., SourceImageName (2).aso). The user is also given the option to rename the ObjectSet file before it is saved.

Interactive shapes are also represented in the ObjectSet Gallery as if they were an ObjectSet. The default name of a set of interactive shapes is the text string "I_" followed by the source image name.

Note that only the **Developer** Task enables the user to specify ObjectSet names when the ObjectSet is created. Double-clicking on an ObjectSet thumbnail displays the ObjectSet in the Visualization window.

Active ObjectSet – The ObjectSet that is visible in the Visualization window that lies on top of all other ObjectSets concurrently displayed in the Visualization window. ObjectSets displayed in the Visualization window are layered (i.e., one on top of the other) such that the most recently opened ObjectSet lies on top of the stack of displayed ObjectSets (i.e., is the Active ObjectSet). This ObjectSet will also have its icon highlighted in the ObjectSet Gallery.

Note: For all tasks except the **Developer** Task, the Active ObjectSet is the only ObjectSet displayed in the Visualization window.

Any open ObjectSet can be made the Active ObjectSet by double-clicking its icon in the ObjectSet Gallery. This action brings the ObjectSet on top of all the others. If any ObjectSet measurements grids exist for the Active ObjectSet, those grids will be opened and displayed in the **Measurement** and **Report Generation** Tasks. Repeated double-clicking of an ObjectSet icon alternately turns on and then off its status as the Active ObjectSet and the display of its objects and measurements grids. However, its icon in the ObjectSet Gallery will remain highlighted until a different ObjectSet icon is clicked or double-clicked.

Attribute – A text or numeric data element describing some aspect of an image or object. Images and objects can have multiple attributes. In Aphelion, we define two types of attributes: spatial and numeric. Spatial attributes are descriptors that provide shape and other geometric information about objects in the form of text (e.g., Bitmap, Line, Circle, Rectangle, Chain, and Point). The shape types (spatial attribute values) are specified in the Tools→Options→Measurements window.

Numeric attributes have measurable, scalar values that describe some aspect of an image (e.g., max pixel value, average pixel value) or an object (e.g., area, length, width, elongation). For each numeric attribute, there is a defined mathematical measurement. A list of the measurements provided in Aphelion is specified for each object shape type in the Tools→Options→Measurements window.

Measurement – A function that computes the value of an attribute defined for an image or ObjectSet. Sometimes, the value itself is also referred to as a measurement.

Grid – A grid displays the attribute measurement values computed for a specific ObjectSet, a set of shape objects, or a set of global measurements computed for an ObjectSet or a set of shape objects. A grid looks very much like a spreadsheet, having one or more columns and one or more rows. A grid row holds the attribute measurement values for one object of the ObjectSet. A column holds the measurement values for a specific attribute for all objects in an ObjectSet. This structure also describes the grid for interactive shapes and for global measurements.

Since the grid for an image displays measurements based on all pixels in the image, the grid is said to contain global measurements. Such grids also include object feature measurements aggregated for all objects in an ObjectSet (Object Global Measurements grid) or for all interactive shapes (Interactive Shape Global Measurements grid). Consequently, a single row in a global measurements grid will contain global measurements for one ObjectSet. Multiple rows can be used to compute global measurements for multiple ObjectSets.

Grids are initially displayed in window B (see *Figure 2.8-1*) of the Aphelion Workspace. Four grid types are provided: Object Measurements, Interactive Shape Measurements, Object Global Measurements, and Interactive Shape Global Measurements. The latter two are considered measurements on the Active Image. For more information about the four grid types, see *Contextual Toolbar – Compute Tools* in section *2.9.5 Measurements Task*.

Message Passing – A very useful software feature that responds to a user action in one display interface by initiating a corresponding action in one or more other display interfaces. Message passing, also called Image Synchronization, is implemented for four combinations of display interfaces:

- between an ObjectSet and its measurement grid;
- between an ObjectSet, its measurement grid, and a 1D or 2D histogram computed from one or two attributes of the ObjectSet;
- between multiple images displayed in the Visualization window;
- between a Profile graph in the Chart window and the corresponding line shape drawn on an image; and
- between a Histogram plot and the corresponding line or closed shape drawn on an image.

Between ObjectSet and Grid – With an ObjectSet displayed in the Visualization window and its measured attribute values displayed in a measurements grid, selecting an object in the Visualization window will cause that object's row to be selected simultaneously in the grid. Conversely, selecting an object row in a grid will cause the simultaneous selection of that object in the Visualization window. Handles will appear around the object. Message passing also works when the user selects multiple objects or rows. Clicking anywhere outside a selected object or object row will deselect all previously selected objects or object rows.

Between Images – Message passing between images is a method by which changes in zoom and pan in one image cause the same zoom and pan operations to occur simultaneously in other selected images. Message passing is available in either One image or Multiple Images modes. Enabling message passing is essentially the same for both modes.

To activate message passing, the user selects those images for which message passing should be active. This is done by clicking on the window banner of each of image while depressing the **Ctrl** key. An icon with opposite-facing arrows ⇆ is displayed in the window banner of each selected image. The presence of this icon indicates the image is active for message passing. In the One image mode, since the images are stacked, it may be necessary to first click on an image's tab in order to display the image and access its window banner.

ROI (Dev version only) – Image processing operations are usually performed either on an entire image or on a portion of an image referred to as a region of interest (ROI). ROIs let you focus on selected areas of an image. Regions of interest are useful to enhance a part of an image, provide faster processing since only a subset of the whole image is processed, or produce a count of objects for a selected region. Two ways to create ROIs in an image are 1) using the mouse to draw one or more ROIs or 2) convert the spatial attributes of an ObjectSet to ROIs. ROIs can have any shape and any size (rectangle, polygon, ellipse, or freehand contour).

Connectivity – Pixel connectivity defines the number of adjacent neighbors a pixel has. A pixel with 4-connectivity is defined to have only four nearest neighbor pixels–the adjacent pixels directly above, below, left, and right. A pixel with 8-connectivity also includes the four pixels diagonally adjacent. Connectivity is often used to determine if two adjacent pixels are part of the same object. With 4-

connectivity, two adjacent pixels are said to be in the same object if they are adjacent vertically or horizontally. With 8-connectivity, two pixels are said to be in the same object if they are adjacent vertically, horizontally, or diagonally.

Docking panel – The Workspace of the Aphelion GUI is a docking panel. This means that the Workspace is subdivided into areas in which windows can be docked. Docked windows are fixed in location while docked. Undocked windows can be moved to virtually any location inside or outside the Workspace. By default, a window is docked until the user undocks it. Note that the Visualization window is always docked. Further details on docking are provided in section *2.8.1 Workspace and windows*.

Calibration – Calibration is a process that defines the size of an image pixel in real-world unit (e.g., inches, microns). This pixel size is called Resolution. The mathematics to compute the resolution is contained in the **Calibration settings** Task. However, the user must provide the real-world measurement of a known object (e.g., gauge, optical micrometer) whose image is captured and displayed on the computer screen and is used to calibrate the image acquisition system.

Resolution – Measurement of the smallest object that can be seen, i.e., the measurement in real-world units of an object represented by a pixel. The resolution depends on the acquisition system (camera and optical system) configuration and the distance between the object and the acquisition system. The resolution is defined by a calibration process for each acquisition system configuration and camera-object distance.

Important note: The calibration process must be performed each time an element of the acquisition system is changed (e.g., camera, lens, distance from the scene or sample to the camera sensor, etc.).

2.8. Graphical User Interface (GUI)

This section describes in detail the Aphelion graphical user interface ("GUI"), the different windows comprising the GUI, and the behavior of each specific window and the user interface it provides. The software rigorously implements the new capabilities of Visual Studio .NET, such as user control of window sizes, functionality, and docking states.

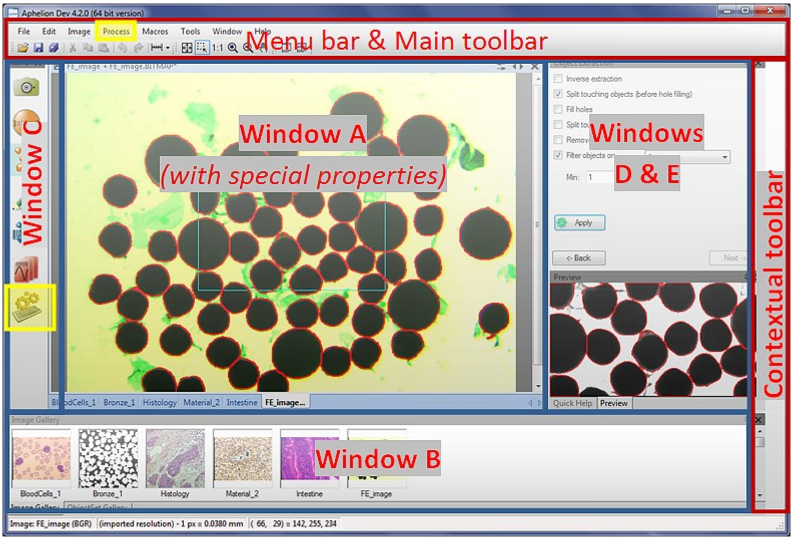


Figure 2.8-1: Aphelion GUI Example – **Object Extraction** Task configuration, showing its GUI subdivisions. Features highlighted in yellow are only available in the Dev version

The Aphelion GUI contains a **Menu Bar**, **Main Toolbar**, task **Contextual Toolbar** (red rectangles in above figure), and a Workspace with four or five windows in a pre-defined layout based on the Active Task (see blue rectangles in above figure). Tasks with only four windows will have window E removed and window D expanded downward.

The following table shows the initial windows configuration for each task. Names in the window columns are descriptive of the user interface function provided by the corresponding window.

Table 2.8-1: Initial windows configuration for each task at start-up. Shaded windows are Control Interfaces, unshaded are Display Interfaces.

PRIMARY TASKS	**W I N D O W**				
	A	**B**	**C**	**D**	**E**
Acquisition	Visualization	Image & ObjSet Galleries	Task Bar	Camera settings	Histogram & Quick Help
Extraction	Visualization	Image & ObjSet Galleries	Task Bar	Object Extraction	Preview & Quick Help
Object Editing	Visualization	Image & ObjSet Galleries	Task Bar	Quick Help	
Measurements	Visualization	Image & ObjSet Galleries, Grids	Task Bar	Quick Help	Chart
Report Generation	Visualization	Image & ObjSet Galleries, Grids	Task Bar	Quick Help	
Developer (Dev version only)	Visualization	Image & ObjSet Galleries, Output, BasicScript Cmd Window, Python Cmd Window	Task Bar	Functions, BasicScript coding, Python coding, Visual Basic & C# coding	Chart & Quick Help
SETTINGS TASK	Administrator mode only				
Camera settings	Visualization	Image & ObjSet Galleries	Task Bar	Camera settings & Options	Histogram & Quick Help
Calibration settings	Visualization	Image & ObjSet Galleries	Task Bar	Calibration settings	Quick Help

Image and ObjectSet galleries, measurement grids, output values, and command prompt interfaces all coexist at various times in window B as stacked windows (see Stacked pad topic in next section).

When Aphelion is started, the initial Active Task is the **Acquisition** Task (in User mode) or the **Camera settings** Task (in Administrator mode) so that the user can capture an image by just clicking the **Snap** icon on the task's **Contextual Toolbar**.

2.8.1. Workspace and windows

The Workspace is divided into a number of rectangular windows that always fill the entire Workspace area and are predefined for each task. The size and proportions of a window can be changed by the user dragging a window's boundary to a new position. This change in the window causes the adjacent window(s) to gain or lose area accordingly. Window sizes may also change automatically. When a docked window is undocked, adjacent windows will expand to systematically fill the Workspace area previously occupied by the newly undocked window. Similarly, if an undocked window is docked into the Workspace, adjacent windows will shrink in a systematic way to accommodate the newly docked window.

For descriptive convenience, we define two windows types: Visualization window, of which there is only one, and task windows which vary in number depending on the Active Task. All task windows exhibit identical behaviors. The following table describes the major differences between the two window types.

Table 2.8-2: Differences between Visualization and task-specific windows

FUNCTION	VISUALIZATION WINDOW	TASK-SPECIFIC WINDOWS
When displayed	Always	Depends on Task
Docked or Undocked	Always docked	Yes
Dockable with Auto-Hide	No	Yes
Dockable with Stacking	No	Yes

Docked Window mode

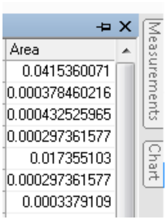
A docked window can be moved to a new location in the Workspace by use of the standard Windows' technique of dragging and dropping. This method undocks the window from the Workspace and then docks it in any dockable location in the Workspace. Dockable locations are indicated by the appearance of docking pads when an undocked window is passed over another docked window.

Note: While a docked task window will always shrink to provide space for another task window being docked adjacent to it, the Visualization window never shrinks to accommodate a newly docked task window unless that task window is being docked on a docking pad that is pointing to a boundary shared between the Visualization window and the Workspace.

Auto-Hide mode

There are two methods for hiding a docked task window in the Workspace: *Auto-Hide* mode and *Stacked*. *Auto-Hide* is described in this section, while the *Stacked* method is described in the *Moving a window* section below.

A docked window can be placed in *Auto-Hide* mode by clicking on the pushpin icon in its window banner. When *Auto-Hide* mode is activated, three changes occur on screen:



Area
0.0415360071
0.000378460216
0.000432525965
0.000297361577
0.017355103
0.000297361577
0.0003379109

- The pushpin rotates to a horizontal position (see graphic on right);
- The window expands to the full length of the boundary it shares with the Workspace, overlaying adjacent windows; and
- An access tab appears aligned along that Workspace boundary.

Clicking in a different window causes the auto-hide window to collapse (i.e., hide) into the Workspace boundary, enabling the overlaid windows to be visible again. There are two ways to reveal hidden auto-hide window: a) simply move the cursor over the window's tab or b) click on the window's tab. In both cases, the hidden window will be displayed. However, in the first case, the displayed auto-hide window will hide again once the cursor is moved into an adjacent window. In the second case, the window won't hide until one clicks in a different window.

The *Auto-Hide* feature can be used to hide any docked task window along any border of the Workspace so that it is easily accessible when needed. This allows the user to make more Workspace area available to view other windows. For example, the Visualization window can be made to fill the entire Workspace by auto-hiding all other windows.

To remove a window from *Auto-Hide* mode, simply click on the window's pushpin. The window will be restored to its prior docked, non-auto-hide state.

Undocked windows

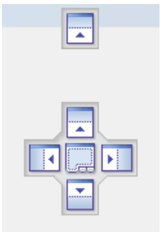
An undocked window is said to be floating in that it can be moved to any location on the computer's display screen that is not a docking pad (see next section, *Moving a* window). Its size and proportions (aspect ratio) can be freely adjusted by the user within the operating system's constraints.

A docked task window (i.e., not the Visualization window) can be undocked in two ways. If the user double-clicks on its banner, the window will become undocked and repositioned to its last undocked location, and conversely. A task window also becomes undocked when the user drags it from its docked position.

An undocked window can be docked in two ways. Double-clicking its banner will cause it to become re-docked at its last docked location. Alternatively, an undocked window can be dragged/dropped on a docking pad.

Help: A set of multimedia demonstrations is available on the Aphelion DVD showing how to use the software.

Moving a window



When a window is dragged to a new Workspace location, a docking control (see graphic at right) appears as the mouse pointer passes over a docked window. The docking control will have one or more pads (small boxes). The symbol on a docking pad indicates where the undocked window will become docked relative to the already docked window (above, below, left, right, or stacked) after the windows are reconfigured to fit in the space previously occupied only by the docked window(s). When the mouse pointer is positioned over one of the docking control pads, a preview of the reconfigured windows appears showing their locations and proportions if the undocked window is dropped on that pad.

Arrowhead pads – In the case where the undocked window is dropped on an arrowhead pad belonging to a task window, the undocked window and the underlying docked window will be reconfigured to share equally the Workspace area formerly occupied only by the previously docked window. The position of the newly docked window will be dependent on which of the available pads is used (above, below, left, and right).

Stacked pad – If the undocked window is dropped on the **Stacked** pad (center icon of the docking control), the undocked window will be reconfigured to the same dimensions as the underlying docked window(s) and the undocked window will be stacked on top of the docked window(s) (similar to an up-turned deck of playing cards). For each undocked window dropped on the **Stacked** pad of a window, the stack in that window area will increase by one and a corresponding tab will be added at the bottom edge of the stack. Clicking on a tab will cause its window to be displayed (to rise to the top of the stack). The value of stacking windows is that the window size is not decreased as new windows are added to the stack.

When an area of the Workspace is vacant as a result of a docked window being moved, the adjacent window(s) will systematically expand to fill the area vacated by the relocated window.

Note: A group of stacked windows can be undocked as a group and can be moved as a group in the same manner as for a single docked window. A window in a stack of windows can be removed from its stack by dragging its tab to a new location or by double-clicking its tab. In the latter case, the window becomes undocked and located where it last existed as undocked.

2.8.2. Task and work context

Use of Aphelion is accomplished through a set of tasks. A predefined work context is associated with each task and is comprised of a set of control and display interfaces, each represented in window form on the screen. A task's work context also includes a contextual toolbar that displays along the right-hand border of the GUI. *Table 2.8-1* gives a list of all window interfaces provided for each task.

Note: A window that was moved to a new location or reconfigured in the work context of a given task will cause the corresponding window in all subsequent tasks to be identically moved or reconfigured. For instance, moving the Histogram/Quick Help window of the **Acquisition** Task (i.e. Window E) will have the same effect on the Preview/Quick Help window of the **Object Extraction** Task and the Chart window of the **Measurements** Task. (See *Table 2.8-1* for these correspondences.)

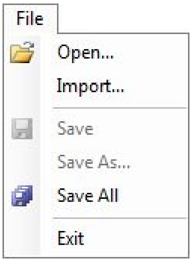
2.8.3. Menu Bar

The **Menu Bar** provides user access to the **File**, **Edit**, **Image**, **Process**, **Macros**, **Tools**, **Window,** and **Help** menus. Each menu contains a list of entries that perform specific operations, as described below. A menu entry can be exercised by a single mouse click.

Note: Some menu entries also can be exercised by clicking on a shortcut found on the **Main Toolbar**. An entry that has a toolbar shortcut will have that shortcut's icon displayed at the left of the entry.

File Menu

Open: Opens a Windows files browser and lets the user select a file to load. The filter in the files browser adapts to the context, and to the highlighted window (with the focus) in the Aphelion GUI. For example, in a BasicScript Macro window, Open only displays .apm macro files. The value of the filter can be changed to display other files or all files present in the current folder.



If the user chooses to open a 2-D image file, the image is loaded into the Image Gallery as a thumbnail and displayed in the Visualization window as the Active Image.

A 3-D image is also loaded into the Image Gallery as a thumbnail, but with a 3D frame overlaid on it (see example adjacent). However, a 3D image cannot be displayed in the Visualization window until a rendering method has been chosen and there are two ways to do this.

3D_Bone

If the parameter DisplayThreeDImagesOnOpen is set to False, a 3D image will not automatically open in the Visualization window. In this case, to display the 3-D image in the Visualization window, the user right clicks on the 3D image's thumbnail and, from the dropdown menu, chooses the Show entry and then selects from the next dropdown menu, the desired rendering method: *Isosurface*, *Volume*, or *Section*.

A 3D rendering method can also be specified as a default condition by going to the Options→Advanced→Appearance→Images window and then setting the parameter *DisplayThreeDImagesOnOpen* to *True* and the parameter *DefaultThreeDViewer* to the desired rendering method: *IsoSurface*, *Volume*, or *Section*. In this configuration, all 3D images will be opened automatically in the Visualization window using the default rendering method.

While multiple copies of a 3D image rendered using different methods can be open concurrently in the Visualization window as stacked windows, they can also be opened concurrently in a single window. To do this, drag and drop the 3D image thumbnail into the desired window. This causes the thumbnail's contextual menu to open. From this menu choose Show and then a select a rendering method (*Isosurface*, *Volume*, or *Section*) from the next dropdown menu.

Import: Starts the Image Importation Assistant tool to guide the user through the two-step importation process. This process will import either a non-standard 2D image consisting of one or more bands or a 3D image generated from a set of 2D, grey-scale images of successive sections.

Image Importation Assistant (Step 1) – The first step begins with the user specifying whether the image to be imported is 2D or 3D. If ***Import a 3D image from multiple 2D image files*** is selected, the user proceeds to Step 2 by clicking the **Next** button.

If ***Import a 2D image from a raw file*** is selected (see example at right), then the window's 2D parameter list becomes active to enable the user to provide values for the imported image's key parameters: *Image size* (in pixels), the *Number of bands*, the *Pixel data type*, and the *File header size*.

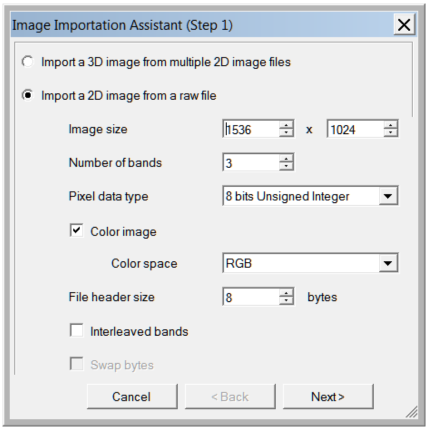


Image Importation Assistant (Step 1)

○ Import a 3D image from multiple 2D image files
● Import a 2D image from a raw file

Image size 1536 x 1024
Number of bands 3
Pixel data type 8 bits Unsigned Integer
☑ Color image
Color space RGB
File header size 8 bytes
☐ Interleaved bands
☐ Swap bytes

Number of bands – When the *Number of bands* selected is greater than 1, the *Interleaved bands* checkbox is made available for selection. If the *Number of bands* selected is 3 or 4, the user is provided a checkbox to indicate that the imported image is a color image. In this case, the user also identifies the appropriate color space from a dropdown list of applicable color spaces.

Pixel data type – The user selects the *Pixel data type* from a dropdown list of available data types. If the user selects a data type containing more than eight bits, the *Swap bytes* checkbox is made available for selection.

When these parameters have been entered, the user clicks the **Next** button to proceed to Step 2 (see below, Step 2 for 2D image importing).

Image Importation Assistant (Step 2) – The content of Step 2 depends on whether a 2D or 3D image is to be imported as is detailed below.

Step 2 for 2D image importing – For 2D image importing, this step's sole purpose is to enable the user to specify the file path for the image to be imported and to complete the importing process. Clicking on the Open File icon opens a file browser. Simply select or enter the path to the file to be imported. Once the path is specified, click Import button to complete the 2D image importing process.

Step 2 for 3D image importing – If *Import a 3D image from multiple 2D image files* was selected in Step 1, then the Step 2 process opens a new window divided into four dialog areas. These include *Files*, *Sub Volume*, *Sub Sampling*, and *Image Registration* (see following below).

Files – First, select the 2D image series by clicking on the ... button next to the Files dialog box. A new window opens, ***Files to import***, with four icons in the right-side toolbar. Click first on the + icon to add the desired series of images. A file browser appears to enable the user to navigate to the folder containing the desired image series. It is beneficial if the file names comply with the following convention:

> [*filename*][*index*].[*ext*], where *filename* is any valid Windows name, *index* is an integer indicating the sequence order position of the

image, and *ext* is the file type extension and must be one of the file types supported by Aphelion.

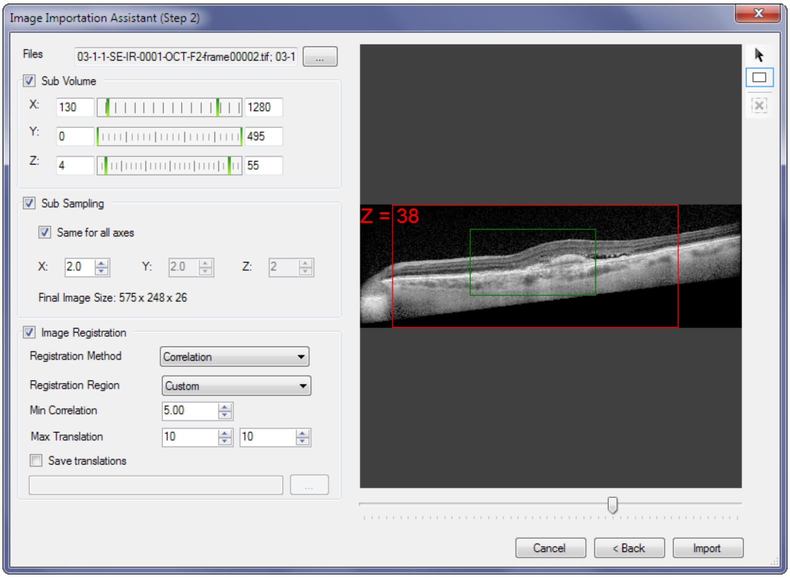


Image Importation Assistant (Step 2)

Files 03-1-1-SE-IR-0001-OCT-F2-frame00002.tif; 03-1

[✓] Sub Volume

X: 130 1280

Y: 0 495

Z: 4 55

[✓] Sub Sampling

[✓] Same for all axes

X: 2.0 Y: 2.0 Z: 2

Final Image Size: 575 x 248 x 26

[✓] Image Registration

Registration Method: Correlation

Registration Region: Custom

Min Correlation: 5.00

Max Translation: 10 10

[ ] Save translations

In this case, the user can simply click on any file in the series and then click the **Open** button to cause the entire series to be loaded in proper (i.e., numeric) order. If the file names do not follow this type of convention, the user can manually select each file by successively **Ctrl** clicking on the files in the desired order. When finished, click the **Open** button to see the ordered file list. If needed, reposition individual files in the list using the ↑ and ↓ icons, or delete a file by clicking on the ✖ icon.

Sub Volume – After the files list has been selected, a sub volume of the 3D volume can be defined by specifying the X_1X_2, Y_1Y_2, and Z_1Z_2 ranges of the volume being generated. These ranges can be defined by either typing in the value for range end points or use the slider graphic to set the end points. The *Sub Volume* checkbox has to be checked to let the user define these values. The sub volume portion of the image is dynamically displayed in the window outlined by a red rectangle (see above screen capture).

Sub Sampling – If the 2D images and/or the number of images are large, sub sampling can be used to speed up the display and processing, and to reduce the amount of data of the 3D volume being generated. Click on the *Sub Sampling* checkbox to activate the sub sampling parameters. The user can specify the sub sample rates for the X, Y, and Z axes of the 2D images; the Z axis corresponds to the number of images. A checkbox enables the user to choose whether rates of sub sampling are different or the same for all the three axes by setting or clearing the *Same for all axes* checkbox.

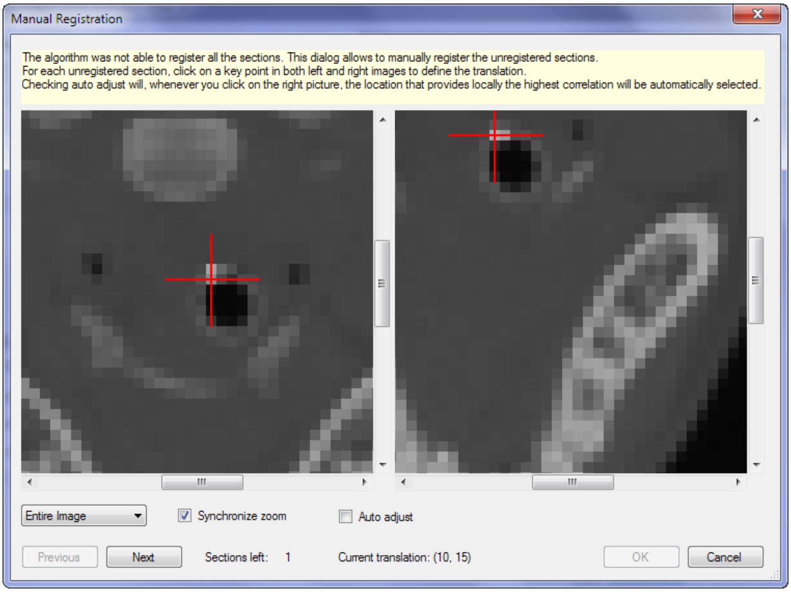
Image Registration – Clicking on the *Image Registration* checkbox causes the image registration parameters to become active. The proper setting of these parameters will result in the registration of the individual 2D images to be adjusted so that the 3D sections are perfectly aligned for an accurate 3D image generation.

- The *Registration Method* parameter is set either to *Correlation* or *Use saved translations*. If the Correlation method is selected, then the following settings are available, otherwise the ... button is enable to select the file containing the saved translations.
- The *Registration Region* can be *Entire*, *Sub Volume*, or *Custom*. If *Entire* is selected, then the registration of two successive sections (2D images) is achieved by searching the minimum correlation in the entire sections. If *Sub Volume* is selected, then the registration of two successive sections is performed determining the minimum correlation in the sub areas of the two sections defined by the X_1X_2 and Y_1Y_2 ranges in the *Sub Volume* subwindow. If *Custom* is selected, then click on the ▭ rectangle icon in the right-side toolbar. Then draw a rectangle outline that specifies a region of interest in one of the sections (see above screen capture). Note multiple regions of interest can be defined. In this latter case, the registration of two successive sections is performed searching for the maximum correlation in the sub areas of the sections defined by the regions of interest.

- The *Min Correlation* parameter defines the minimum correlation value. If the resulting correlation value is lower than the defined value, then the Manual Registration window pops up letting the user manually register two successive sections once the registration process completes.
- The *Max Translation* parameter defines the maximum translation (i.e., distance) allowed between two adjacent sections, defined in pixel unit, when performing the registration process. If the resulting translation value is higher than the defined value, then the Manual Registration window pops up letting the user manually register two successive sections once the registration process completes.
- If the *Save translations* checkbox is checked, then the user can define the path and the name of the file containing the translation values computed during the registration process. Click on the ... button to open a filer.

Once all parameters are defined, click on the Import button to process the set of 2D images and generate the 3D image.

Manual Registration – If the registration process fails to register one or more sections due to a low correlation value or a high translation value, then the Manual Registration window pops up. The first unregistered section and its previous section are respectively displayed in the right and left image view windows. The user must then specify a similar point in both the left and the right images to define the translation to be applied to register the two sections. Checking Auto Adjust determines the point in a neighborhood of the mouse pointer in the right image for which the correlation value with the specified point in the left image is maximum, even if a point has been manually specified in the right image. Moreover, the correlation value at this point is displayed when the user clicks on the right image. The **Next** button is available when at least one section is not registered. Note the **OK** button is greyed out.


Manual Registration

The algorithm was not able to register all the sections. This dialog allows to manually register the unregistered sections.
For each unregistered section, click on a key point in both left and right images to define the translation.
Checking auto adjust will, whenever you click on the right picture, the location that provides locally the highest correlation will be automatically selected.

Entire Image; [✓] Synchronize zoom; [ ] Auto adjust

Sections left: 1

Current translation: (10, 15)

Clicking on the **Previous** button displays the previous unregistered section, disables key points previously specified and lets the user specifying new key points. Once all key points are specified for all unregistered sections, then the **OK** button becomes available. Clicking on the **OK** button completes the import process and registers all sections. Clicking on the **Cancel** button cancels the import process.

Save: Saves the objects with the focus in the highlighted window. The **Save** entry is only enable if the objects with the focus have been modified. If there is more than one object in the window (e.g., an image and an ObjectSet), the menu changes to reflect the modified objects displayed in the window. Typically, the menu entry can be:

Save Image *ImageName*

Save ObjectSet *ObjectSetName*

Save *Multiple Objects…*

In case of multiple objects, a window pops up letting the user select the objects to be saved. The names of the output files are the names of the objects and the extensions are the default ones (defined in the **Tools→Options→Files** menu for the images and ObjectSets – see *Files Options* in this section for more details).

Tip: To save multiple images/ObjectSets, select the corresponding thumbnails in the Image/ObjectSet Gallery, and then click on the **Save** entry.

Save As…: Save the unmodified entities with the focus in the highlighted window. In case there are more than one object in the window (e.g., an image and an ObjectSet), the menu changes to reflect the objects displayed in the window. Typically, the menu entry can be:

Save Image *ImageName* As…

Save ObjectSet *ObjectSetName* As…

Save *Multiple Objects* As…

In case of multiple objects, a window pops up letting the user select the objects to be saved. The names and extensions of the output files can be edited if desired.

Note: In case multiple images are selected in the Image Gallery window or multiple ObjectSets are selected in the ObjectSet Gallery window, then the last selected entity is the one with the focus. It means the save operation will affect this object.

Save All: Save all modified objects available in the Aphelion GUI, including images, ObjectSets, color palette, etc. The list of objects that will be saved can be customized in the **Tools→Options →Advanced** menu. In case multiple objects are to be saved, a window pops up letting the user select the objects to be saved. The names and extensions of the output files can be edited if desired.

Note: Image resolution is only saved with .tif and .asi file formats. Since the .asi (Aphelion serialized image) format is proprietary to the Aphelion Imaging Software Suite products and is not compatible with any other imaging software or devices, it is recommended that images be saved in TIFF format if they may have to be imported into a third party software product.

Note: The .aso file format is an Aphelion 4.x proprietary format for saving ObjectSets. It includes and extends the capabilities of the .tks format which was used for the Aphelion 3.2 software products. It is recommended that the .aso format be used for most Aphelion situations. The .tks format should be used only when working with ObjectSet files generated by an Aphelion 3.2 software product.

Note: The icon is grayed out if no object can be saved in the highlighted window.

Exit: Ends the current project work session and frees all memory that was being used, and closes the Aphelion program. During this closing process, the user is given the chance to save any unsaved image and ObjectSet data.

Edit Menu

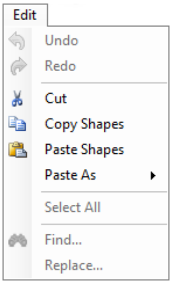
The **Edit** menu provides tools that enable the user to edit selected entities depending on the context, such as text in a Macro window or a graphic object displayed in the Visualization window.

Undo: Undo only the last editing operation.

Redo: Redo only the last Undo operation.

Cut, Copy, Paste: Standard object editing behavior.

Paste As…: Create an image that contains the data copied in the clipboard if the Image option is selected, or paste one or more selected objects in an Aphelion ObjectSet in a graphic or an ROI.



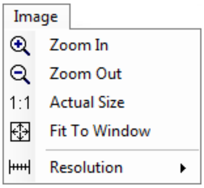
Note: To quickly convert selected objects in an ObjectSet into graphic objects or ROIs, select the **Developer** or **Measurement** tasks, select the objects and copy them in the clipboard, then select the **Developer** Task, click on **Edit→Paste As…** menu and choose Graphic or ROI to paste the objects in the active image respectively as a graphic overlay or a ROI.

Select All: Select all image or ObjectSet thumbnails displayed in the Image or ObjectSet gallery. Selected objects can then be closed by right-clicking on the Close entry in the contextual menu.

Find: Find a character string specified by the user in the active macro window.

Replace: Find a character string specified by the user and replace it with a new character string in the active macro window.

Image Menu



The **Image** menu provides tools that enable the user to alter the display of the Active Image in the Visualization window. Shortcut buttons for these functions are provided on the **Main Toolbar**.

Zoom In: Zooms into the Active Image. The zooming factor is times $\sqrt{2}$. One click on **Zoom In** increases the inter-pixel distance by a factor of 1.414; clicking twice doubles the inter-pixel distance.

Zoom Out: Zooms out from the Active Image. The zooming factor is times $1 \div \sqrt{2}$. One click on **Zoom Out** decreases the inter-pixel distance by a factor of 0.707; clicking twice halves the inter-pixel distance.

Note: The zooming factor can also be altered by turning the mouse wheel.

Actual Size: Displays the Active Image so that image pixels are in a 1:1 relationship with screen dots (i.e., no zooming applied). Scrolling or panning may be necessary to view all areas of the Active Image.

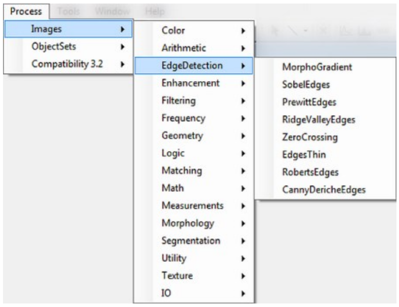
Fit To Window: Zooms the Active Image so that the entire image is displayed as large as possible in the Visualization window. The zoom factor applied is identical for both the X and Y axes in order to preserve the image’s aspect ratio.

Important note: When **Fit To Window** is selected, as denoted by its menu icon turning blue and its toolbar button gaining a blue outline, all other zoom tools are disabled. To gain access to the other zoom tools, unselect **Fit To Window** by clicking again its menu entry or its toolbar button.

Resolution: Selects a predefined resolution to be associated with the current image. If no resolution has been defined, then the menu entry is **None**.

Process Menu (Dev version only)

All Image and ObjectSet functions can be invoked from this menu. The menu has three or four panels (see below), the first providing three major function groupings: Images, ObjectSets, and Compatibility 3.2 functions. The second panel divides these major groupings into sub-groups of related functions. The third panel lists either an additional level of sub-grouping or the individual functions contained in the selected group or sub-group.



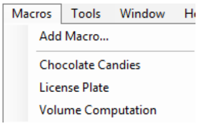
If a fourth panel is provided, it lists the functions contained in the second level sub-group. When the mouse pointer is hovered over a function, a tooltip appears providing a brief explanation of the function. The selection of any function from the **Process** menu's third or fourth panel will automatically activate the **Developer** Task and initialize the functions window in the window D location of the user interface.

The third entry of the first panel, Compatibility 3.2, provides access to all of the Aphelion 3.2 functions laid out in the hierarchy of that version. This menu entry can be hidden by selecting **False** for the option found at:

Tools→Options→Advanced→Developer→ShowCompatibilityGroups.

Note that Aphelion 4.x contains new functions that were not available in version 3.2.

Macros Menu (Dev version only)



The **Macros** Menu lets the user add a list of macros in this menu, to allow a quick run of a macro.

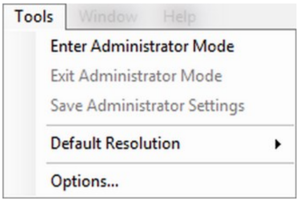
Note: To define macro sub-menus, edit with Notepad or a similar program the user.config file located in the C:\Documents and Settings \<User>\Application Data\ADCIS\Aphelion Dev\4.x.x or C:\Users \<User>\AppData\Roaming\ADCIS\Aphelion Dev\4.x.x folder respectively for Windows XP and Windows Vista/7, and add a string of character after the double quote ending with a "." as shown in the example below:

```
<Project>
    <MacroShortcuts>
        <MacroShortcut Type="Aphelion4.Scripting.MacroShortcut">
        <MacroShortcut Name="MyMacros.Ceramic"
        FileName="C:\Program Files\Aphelion
        4.1.1\Dev\Macros\C#\Ceramic.cs" Arguments="" />
        </MacroShortcut>
    </MacroShortcuts>
</Project>
```

The AphUserMacroAdd BasicScript command can also be used to add an entry in the Macros menu. This command allows defining sub-menus in this menu.

Tools Menu

The **Tools** Menu contains a variable number of entries depending on which Aphelion Extensions are installed. If none are installed, the menu will be as pictured to the right with five basic entries. When certain Extensions are installed, they will have an entry in this menu to enable a user to call or start the Extension (e.g., Classifier Builder).

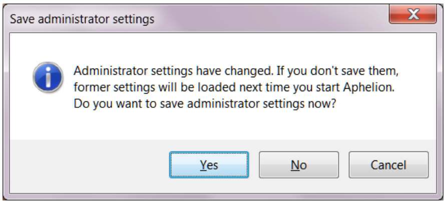


Note: In some cases, the Extension may start in the background of the Aphelion Dev program. Click on the Extension icon in the Windows Task bar usually located at the bottom of the screen to activate the Extension.

Enter Administrator Mode: Clicking on this menu entry will cause the Aphelion system to immediately activate Administrator Mode. If a password was previously created, then a password entry box is displayed before entering Administrator Mode. When Aphelion is in Administrator Mode, the user has access to all of Aphelion's settings. These settings enable an Administrator to configure Aphelion for optimal performance of the user's projects and are accessed via the Options menu entry (see below). Note that settings for cameras and settings for calibrating the acquisition system and images coming from another source are accessed via entry on the Task Bar that only displays when the Administrator Mode is active (see section *2.9 Tasks*).

Note: If no password has been defined, and if the *AutoEnter-AdministratorMode* parameter is set to *True* in the *Identification* options of the Advanced settings (**Tools→Options→Advanced**), then Aphelion starts in Administrator mode for all types of users. This parameter is set to *True* by default. However, the *True* setting is ignored if a password is defined.

Exit Administrator Mode: Clicking this entry deactivates the Administrator Mode. If any Options settings were changed since the last time settings were saved, the adjacent Save window appears giving the user a chance to save any changed settings prior to deactivating Administrator Mode. If the user clicks on the No button at this time, then settings changes are retained for the current work session but are not saved for future work sessions. In this case, when the user ends the current Aphelion work session (terminates Aphelion execution), if the changed settings have not previously been saved, the same Save window appears to provide a last opportunity to save the changes for the next Aphelion work session. If the user clicks on the Yes button, then the settings changes are saved. If there are any unsaved settings changes when the user terminates the Aphelion work session, the Save window appears, providing a last chance to save changed settings.


Save administrator settings

Administrator settings have changed. If you don't save them, former settings will be loaded next time you start Aphelion. Do you want to save administrator settings now?

Save Administrator Settings: Saves the changed settings, but Aphelion remains in Administrator Mode. This operation is only available in Administrator Mode.

Default Resolution: Clicking on the Default Resolution menu entry displays a list of all resolutions previously created and named by an Administrator using the Calibration Task. By selecting the appropriate named resolution from this list, the user causes that resolution to be attached to the Active Image for use in computing measurements on this image. For more information on calibration and resolution, please see Section *2.7 Terminology*.

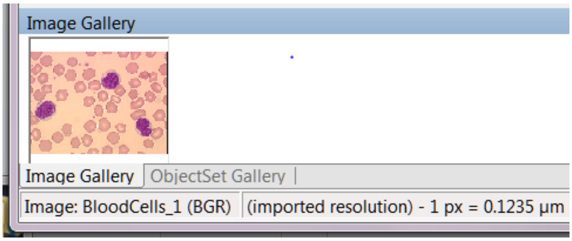


Image: BloodCells_1 (BGR) | (imported resolution) - 1 px = 0.1235 µm

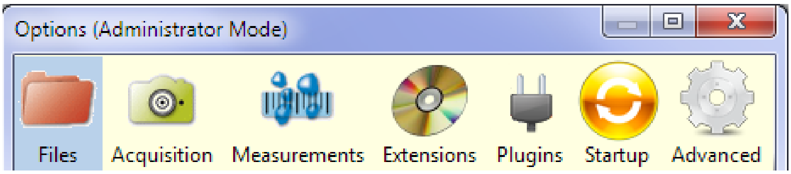
When an image opened from storage media includes a resolution that corresponds to none of the resolutions that were defined using the **Calibration settings** Task, then this imported resolution appears in Aphelion's status bar.

Note: If a resolution is attached to the current camera configuration (refer to *2.9.1 Administrator Settings Tasks*), then this resolution is applied to the captured image, over-riding the selected **Default Resolution**.

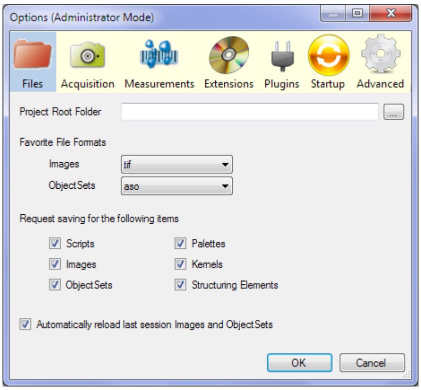
Options…: This entry on the Tools menu provides access to most of the settings available for configuring the Aphelion software. The remaining settings are accessed via the Taskbar and its **Camera Settings** and **Calibration Settings** tasks. Non-Administrator users do not have access to the **Camera Settings** task. Instead, such users have access to the **Image Acquisition** task which has limited settings.

Administrator settings include parameters for image acquisition, project data, measurements, and Advanced settings. Advanced settings include specification of interface colors, the separator in the grid, the compatibility with previous versions of Aphelion, and other settings that typically do not need to be updated frequently.

The Options window is organized into groups of functionally or operationally related settings. These groupings are selectable by clicking on an icon located along the top edge of the window, as is shown below.



Files Options



Project Root Folder:

Favorite File Formats
Images: tif
ObjectSets: aso

Request saving for the following items
- [✓] Scripts
- [✓] Palettes
- [✓] Images
- [✓] Kernels
- [✓] ObjectSets
- [✓] Structuring Elements

[✓] Automatically reload last session Images and ObjectSets

Selection of **Tools→Options →Files** opens the Files control interface containing settings specific to how files are managed by Aphelion. From this interface, the user can specify the *Project Root Folder* into which data and measurements files will be stored. This interface is also used for specifying the default formats for images and ObjectSets and the types of data that are to be saved. The user can also specify which Aphelion objects can be saved when exiting the program (e.g., Aphelion macros or scripts, images, ObjectSets, kernels, structuring elements, and color palettes). If the last checkbox is selected, the last user session will be reloaded when Aphelion starts, including images and ObjectSets that were open in the Galleries in the previous session.

Tip: If the *Project Root Folder* is not specified by the user, then the default folder for storing the user's images and ObjectSets is:

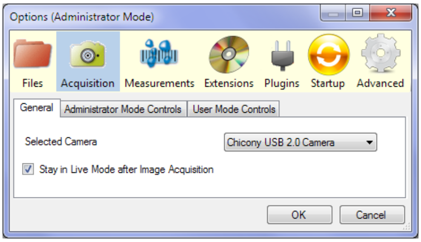
	Aphelion Lab	Aphelion Dev
XP	*"C:\Documents and Settings\User name\Documents\…*	
	…Aphelion Lab"	*…Aphelion Dev"*
Vista, 7, & 8	*"C:\Users\User name\Documents\…*	
	…Aphelion Lab"	*…Aphelion Dev"*

Using these default folder names ensures that data saved by one user are kept separate from data saved by other users.

Acquisition Options

The Acquisition Options are divided into three groups, each group having its own tab: General, Administrator Mode Controls, and User Mode Controls. The contents of the three tabs are described in the following paragraphs.

General: Cameras that have previously been registered with the Aphelion system will appear in the Selected Camera dropdown list. (The camera registration process is described in the section titled "Administrator Mode Controls" following below.) The user selects the camera that will be used in the current work session to capture images or video.

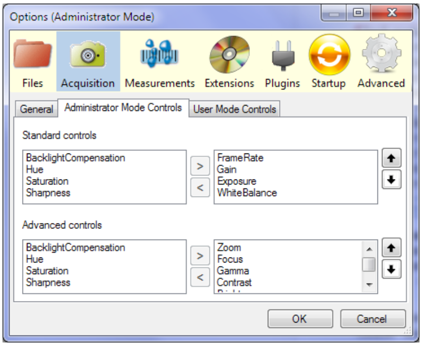


Selected Camera: Chicony USB 2.0 Camera
☑ Stay in Live Mode after Image Acquisition

Check-marking the *Stay in Live Mode after image acquisition* box under the General tab specifies that after an image is acquired (i.e., **Snap** button was pressed), the live image remains displayed in the Visualization window. If the box is not check-marked, the live image is replaced by the acquired image in the Visualization window. The *Select Camera* box is used to select the acquisition device to be used to receive live video and to snap images.

Note: Only administrators can select the acquisition device.

Administrator Mode Controls: This interface is used by an Administrator to specify those camera controls that only an Administrator can change. Changing these controls is accomplished with the **Camera settings** Task.



Standard controls

BacklightCompensation	FrameRate
Hue	Gain
Saturation	Exposure
Sharpness	WhiteBalance

Advanced controls

BacklightCompensation	Zoom
Hue	Focus
Saturation	Gamma
Sharpness	Contrast

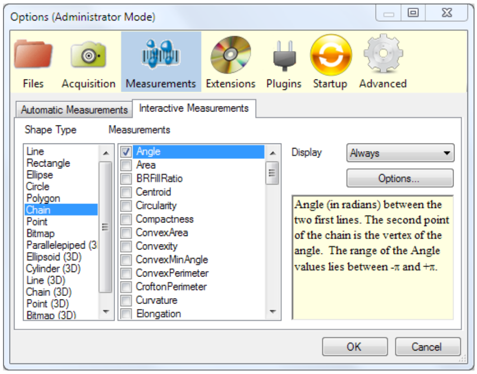
The Administrator Mode Controls dialog box divides camera controls into two categories, *Standard controls* and *Advanced controls*. Camera controls that are displayed in the window depend on the driver interfaced to Aphelion and the manufacturer specifications. They are automatically registered with Aphelion when the camera is connected to the computer while it is running Aphelion. When registered, all of the controls initially appear listed in the left box of both Administrator Mode Controls and User Mode Controls dialog boxes, and the right box of those dialog boxes remain blank until an Administrator moves controls from the left box to the right box. An Administrator can then move any listed control between the left and right boxes by selecting a control and then clicking on the < and > direction buttons as needed. Moving a control from the left window to the right window will cause that control to be visible in the Camera settings control interface visible when the **Camera settings** Task is selected in Administrator Mode.

User Mode Controls: This interface is used by an Administrator to specify those camera controls, both standard and advanced, that a non-administrator user will be allowed to change when using the **Image Acquisition** Task in the User Mode. Moving a control from the left box to the right box will cause this control to be accessible by non-Administrator users. This enables the Administrator to limit what changes can be made by non-Administrator users.

Note: To learn more about the selected camera's modes and controls, please refer to the camera manufacturer's documentation.

Measurements Options

Selection of the **Tools→Options→Measurements** option enables an Administrator to specify the measurements that will be computed on objects extracted from images or manually drawn or edited by the user. The Measurements control interface lists each Shape Type available in Aphelion and a check box menu of Measurements available for each Shape Type. The **Measurements** Task will compute the check-marked measurements for each object in an ObjectSet based on the object's Shape.



Shape Type: Chain

[✓] Angle

Display: Always

Angle (in radians) between the two first lines. The second point of the chain is the vertex of the angle. The range of the Angle values lies between -π and +π.

Object types are divided into two categories: Automatic and Interactive.

- Automatic: Measurements computed on these object types are generated in the **Developer**, **Object Extraction**, or **Object Editing** Tasks, and are displayed in a grid in the Object Measurement Tab found in the Gallery. The Automatic Measurements panel enables the user to select the measurements to be automatically computed for each shape type in this category.

- Interactive: Measurements computed on these object types are generated in the **Measurements** Task using the interactive drawing tools available in the Task, and are displayed in a grid in the Interactive Shape Measurements Tab found in the Gallery. As an option, interactive Measurements values may also be displayed in the image overlay. The Interactive Measurements panel enables the user to select the measurements to be computed in real-time for each shape type in this category.

Available shape types are: Line, Rectangle, Ellipse, Circle, Polygon, Chain, Point, Bitmap, and 3D shapes (Parallelepiped, Ellipsoid, Cylinder, Line, Chain, Point, Bitmap), only available when the 3D Image Processing extension is licensed and loaded.

Bitmap is a shape only available for objects generated either in the **Developer**, **Object Extraction**, or **Object Editing** Task.

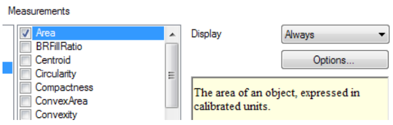
Line, Polygon, and Chain are shapes available for objects generated either in the **Developer** or **Measurements** Task.

Rectangle, Ellipse, and Circle are shapes available in objects generated in the **Measurements** Task.

Available measurements depend on the shape that is computed or drawn.

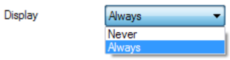
Note: All graphic objects drawn by the user in the **Measurements** Task are assigned a *Shape* type.

When a measurement is highlighted by the user, a brief description of that measurement appears in the help box at the right-side of the Measurements panel. If a measurement is both selected and highlighted, two display settings are enabled immediately above the help box, as shown beside.



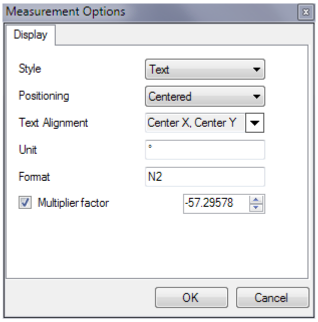
The area of an object, expressed in calibrated units.

Display: Define whether the measurement is "*Always*" or "*Never*" displayed in the image overlay.



Options: Provides a pop-up menu of display parameters that enable the user to define the location and format of the displayed measurement.

Style: Specify the text displayed as a string or in a tooltip. Currently, only a *Text* string default is available.



Measurement Options

Display

Style	Text
Positioning	Centered
Text Alignment	Center X, Center Y
Unit	°
Format	N2
☑ Multiplier factor	-57.29578

Positioning: Specify the default position of the text relative to the object. Currently, only a *Centered* default is available.

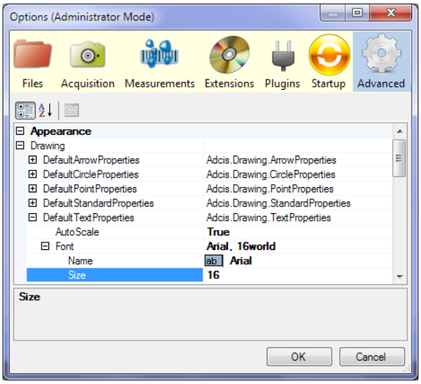
Text Alignment: Define the text alignment.

Unit: Specify the text to add after a measurement value to identify applicable units (e.g., µm, pixel).

Format: Specify the format of the text. The available formats are *Numeric (N)*, *Fixed-Point (F)*, *Exponential (E)*, and *General (G)*. The digit after the letter specifies the number of decimal digits except for G. For this latter case, it gives the number of significant digits.

Multiplier factor: Specify a constant as a multiplier of measurement results. This is useful for rescaling or converting to a different unit. For example, the number 57.296 (i.e., 180/π) can be set as the Multiplier factor to display angle measurements in degrees that were computed in radians

Tip: To change the size of the font, open the Tools→Options… →Advanced window, select *Drawing→DefaultTextProperties →Font*, and then update parameter values there.



Options (Administrator Mode)

⊟ Appearance	
⊟ Drawing	
⊞ DefaultArrowProperties	Adcis.Drawing.ArrowProperties
⊞ DefaultCircleProperties	Adcis.Drawing.CircleProperties
⊞ DefaultPointProperties	Adcis.Drawing.PointProperties
⊞ DefaultStandardProperties	Adcis.Drawing.StandardProperties
⊟ DefaultTextProperties	Adcis.Drawing.TextProperties
AutoScale	True
⊟ Font	Arial, 16world
Name	Arial
Size	16

Size

Automatic Measurements

Measurements selected in the Automatic Measurements panel are automatically computed on the objects generated in the **Object Extraction** and/or **Object Editing** Tasks when the button is clicked.

Interactive Measurements

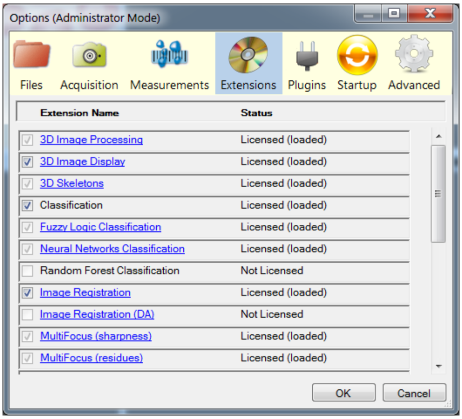
The measurements selected in the Interactive Measurements panel are computed on the fly when a shape is drawn using the drawing tools available in the **Measurements** Task. The drawing tools used to create geometric shapes are: Line, Circle, Polyline, Polygon, Caliper (Line with start and end bars) and Angle (Polyline made of three points). Their shape types are then, respectively, Lines, Circles, Chains, Polygons, Line, and Polyline, and the measurements selected for these types will be computed in the **Measurements** Task as soon as the shape is drawn.

Extensions *(Administrator mode only)*

An Aphelion extension is a set of tools that is usually used in the Aphelion GUI. However, note that some extensions are stand-alone applications that can be used outside Aphelion. Typically, a specific license code is required to be able to use one or more Aphelion extensions, although some extensions do not need a specific license code (i.e. BasicScript, C#, and Python scripting languages, camera emulation using AVI files).

Note: Aphelion includes a set of default camera drivers to control most common cameras. Drivers for certain other cameras are available as extensions of Aphelion.

Selection of **Tools→Options→Extensions** opens the **Extensions** control interface. It displays the list of all of the Aphelion extensions, not including standalone extensions, that were available for purchase at the time Aphelion was installed on the current computer. Extensions that are licensed on the current computer are denoted by the word "*Licensed*" in the *Status* field. If the text "*(loaded)*" appears adjacent to "*Licensed*", then the necessary libraries and driver(s) were loaded for that extension when the current execution of Aphelion was started. If the *Status* field of a listed extension shows only "*Licensed*", then one or more necessary drivers or libraries were not included when the current execution of Aphelion was started.



	Extension Name	Status
✓	3D Image Processing	Licensed (loaded)
✓	3D Image Display	Licensed (loaded)
✓	3D Skeletons	Licensed (loaded)
✓	Classification	Licensed (loaded)
✓	Fuzzy Logic Classification	Licensed (loaded)
✓	Neural Networks Classification	Licensed (loaded)
	Random Forest Classification	Not Licensed
✓	Image Registration	Licensed (loaded)
	Image Registration (DA)	Not Licensed
✓	MultiFocus (sharpness)	Licensed (loaded)
✓	MultiFocus (residues)	Licensed (loaded)

The text "*Licensed*", with or without the text "*(loaded)*" appended to it, only appears for the extensions included in the License Code that was used when Aphelion was last installed on the computer.

The checkbox for each extension has four states: Checked+notGreyed (CnG), Checked+Greyed (CG), NotChecked+Greyed (nCG), and NotChecked&NotGreyed (nCnG). These states have the following meanings:

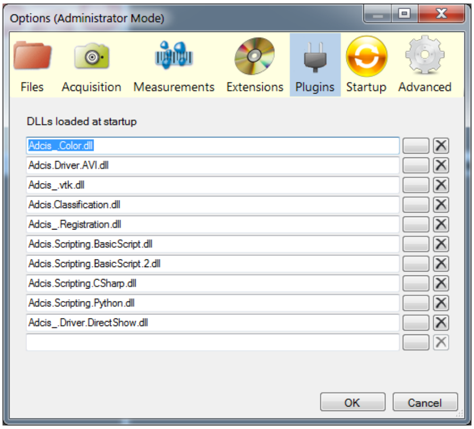
CG	Extension is always loaded and cannot be unloaded by user
CnG	Extension is licensed & loaded and can be unloaded by user
nCG	Extension is not licensed
nCnG	Extension is licensed and can be loaded by user

The first time Aphelion is started after an installation, all licensed extensions are configured to be loaded into main memory. By unchecking a licensed extension whose checkbox is not greyed, the user is able to prevent the extension's software from being loaded into main memory. This can have a positive impact on Aphelion start-up time and free up a portion of main memory. It can also prevent warning messages that may occur if the extension requires the presence of non-Aphelion software that has not been loaded or absent external device. An example of this situation is presented in the following Note.

Note: It is recommended to not load a camera driver if the camera is not connected to the computer. Some drivers make calls to software that is provided only by the camera manufacturer to enable communication between the computer and the camera. Such a situation may result in troublesome warning messages or in unpredictable behaviors by the camera.

***Plugins** (Administrator mode only)*

A plugin is a DLL or a programming library that is included in Aphelion Dev. At startup, Aphelion can load one or more plugins. Some Aphelion extensions may need to load a set of plugins, while other extensions do not (i.e., 3D Image Processing, 3D Skeletonization, Image Registration, and MultiFocus). Note an Aphelion user can create his/her own plugin. New user functions are usually loaded in Aphelion through the use of a plugin, although a C# script can also be loaded to add new image processing functions in the Aphelion GUI.



DLLs loaded at startup

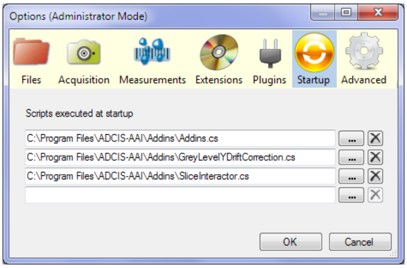
Adcis._Color.dll
Adcis.Driver.AVI.dll
Adcis._vtk.dll
Adcis.Classification.dll
Adcis._Registration.dll
Adcis.Scripting.BasicScript.dll
Adcis.Scripting.BasicScript.2.dll
Adcis.Scripting.CSharp.dll
Adcis.Scripting.Python.dll
Adcis._Driver.DirectShow.dll

Selection of **Tools→ Options→Plugins** opens the **Plugins** control interface. It lists the plugins that are loaded when starting Aphelion. To unload a plugin, click on the ☒ checkbox to clear it and then click the **OK** button. In the Save administrator settings window that appears, click on the **Yes** button to save the Administration Settings and to restart Aphelion. When Aphelion restarts, the plugin is not loaded and does not appear in the *Plugins* list. Remember that since the plugin is no longer loaded, any extensions that use it will no longer function.

To load a plugin that was not loaded (i.e., not in the *Plugins* list) when Aphelion was last started, locate the blank entry (i.e., last entry) in the list and click on its ... button. This causes the file browser to open to the directory containing all of the Aphelion plugins provided with the current Aphelion installation. From there, the user can choose an Aphelion plugin or navigate to any directory containing the plugin the user wishes to install. Select (highlight) the desired plugin (a .dll file) and click the **Open** button. Next, click the **OK** button in the Plugins list (DLL loaded at startup) window. This causes opening of the same Save administrator settings window that opens when a Plugin is unloaded. Click the **OK** button and then, in the window that appears, click on the **Yes** button to close Aphelion and have it automatically restart. Be sure to check to see that the desired plugin is now loaded. If you know what extensions require the newly loaded driver, check the extensions list to see that those extensions now show that they are loaded.

Startup *(Administrator mode only)*

This menu lets the user define the list of macros /scripts executed when starting the program. These scripts can be a mix of BasicScript, C#, or other scripts developed in the supported languages.



Scripts executed at startup

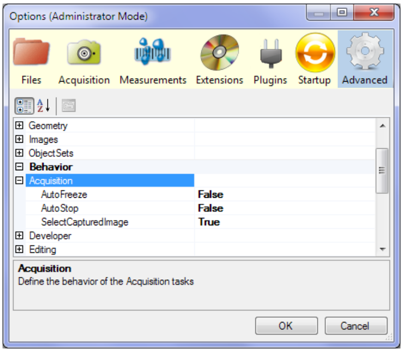
C:\Program Files\ADCIS-AAI\Addins\Addins.cs
C:\Program Files\ADCIS-AAI\Addins\GreyLevelYDriftCorrection.cs
C:\Program Files\ADCIS-AAI\Addins\SliceInteractor.cs

Advanced Options *(Administrator mode only)*

Advanced options can be configured in this panel. The five main parameter categories are:

Appearance: Display/Change image and ObjectSet appearance settings.

Behavior: Display/Change settings for **Image Acquisition**, Object Extraction, and **Developer** Tasks.



Geometry
Images
ObjectSets
Behavior
Acquisition
AutoFreeze False
AutoStop False
SelectCapturedImage True
Developer
Editing

Acquisition
Define the behavior of the Acquisition tasks

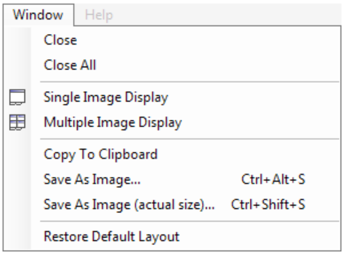
Files: Specify the Save action settings for files.

General: Specify administrator password, user name, language of the user interface, license code, and other system information.

Misc: Display/Change the graphical user interface location, size, status (maximized, minimized, normal).

Refer to *APPENDIX I - Options→Advanced Menu* to get a detailed description of each parameter available in the Options→Advanced window.

Window Menu



Close: Close the Active Image and free the memory space used by the image data. Note that clicking on the Visualization window's ✕ button will also close the image displayed in the corresponding window. This operation is not available if there is no Active image.

Note: ObjectSets can be individually closed by right-clicking on an ObjectSet's icon in the ObjectSet Gallery window, then clicking **Close**.

Close All: Close all images and ObjectSets currently opened respectively in the Image Gallery and in the ObjectSet Gallery window. A window pops up prompting the user which images and ObjectSets need to be saved before being closed. This operation is not available if there is no loaded image and no loaded ObjectSet.

Single Image Display: Display one image in the Visualization window.

Multiple Image Display: Displays multiple images in the Visualization window as an array specified by the MIVTColumnCount and MIVTRowCount parameters in the Advanced Options (2x2 images by default).

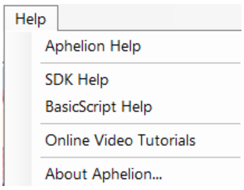
CopyToClipboard: Copy the displayed part of the active image to the clipboard.

Save As Image…: Saves the entire contents of the Active Image window (i.e., Window A) as a TIFF image. The saved image will have the same zoom factor, size, and visible non-image border (if any) as appeared in the Active Image window.

Save As Image (actual size)…: Saves the entire image and overlay (if any) displayed in the Active Image window as a Tiff file, preserving the actual size and aspect ratio (i.e., zoom value of one) of the image.

Restore Default Layout: Restore the default Aphelion Graphical User Interface layout, as it appears in *Figure 2.8-1* earlier in the document.

Help Menu



Help Contents: Opens the online Aphelion User Guide.

SDK Help: Opens the online Aphelion Image and ObjectSet Processing Function Guide.

BasicScript Help: Opens the online BasicScript macro language Guide.

Online Video Tutorials: Lets the user play the video tutorials explaining the use of the software. It opens a web interface that displays the list of available videos.

Note: An internet connection must be available to play these tutorials.

About Aphelion: Displays a window containing software version and legal information about Aphelion (i.e., Aphelion version, release number, Identification Code, License Code, and copyrights).

2.8.4. Main Toolbar

The **Main Toolbar** provides shortcuts for many of the operations provided by the menus on the **Menu bar**. See section *2.8.3 Menu Bar* for a detailed description of these operations. The **Main Toolbar** also contains some operations not available on the **Menu bar**. These are described below.

File Menu Shortcuts:

Open, Save Image, and Save All: see section *2.8.3 Menu Bar*.

Edit Menu Shortcuts:

Cut, Copy, Paste, Undo, and Redo, see section *2.8.3 Menu Bar*.

Resolution: Used to select a resolution that is only applied to the Active Image.

Help: The resolution associated with an image is saved with the image if the format is either TIFF (.tif or .tiff) or Aphelion Serialized Image (.asi). In these cases, the resolution will be applied automatically when loading the image. However, the resolution will be lost if the image is saved in a format other than .tif, tiff, or .asi.

Image Menu Shortcuts: 1:1

Fit to Window, Zoom in area of interest, Actual Size, Zoom In, and Zoom Out, see section *2.8.3 Menu Bar*.

Zoom area of interest: Enables use of the mouse to draw a rectangular area of interest in the Visualization window, which then results in that area is being expanded to optimally fill the window. This button is highlighted in blue when active and it remains active until it is clicked again.

Pan: Enables use of the mouse to move the image in the horizontal and vertical directions. Press the left button and move the image while pressing the button.

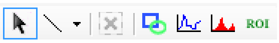
One Image: Displays one image in the Visualization window (i.e., the Active Image). This is the default mode when starting Aphelion. All images open in the Image Gallery will be open in this mode as a stack of images with the Active Image on top of the stack. If the One Image button is clicked when the Visualization window is in *Multiple image* mode, the multiple images will be reconfigured into a stack of images, with the Active Image on top and the image tabs in the order in which the images existed in the image array. Any image in the stack can be displayed (moved to the top) by clicking on its tab.

Multiple Images: Displays multiple images in the Visualization window as an MxN array where M and N are specified by the MIVTRowCount (i.e., M) and MIVTColumnCount (i.e., N) parameters of the Tools→Options→Advanced→Images window (2x2 array by default). The first K images open in the Image Gallery will be used to populate the MxN array. This array will be displayed as a single panel of images having a tab at the bottom labeled Panel 1. If the number of images open in the Image Gallery exceeds K, additional panels will be created as needed.

Note: The message passing feature is available in both the One image and Multiple images modes.

Developer tools: These tools are used in the **Developer** Task to create histograms, profiles, and regions of interest. Please see following section, *2.8.5 Profile – Histogram – ROI Toolbar (Dev version only)*.

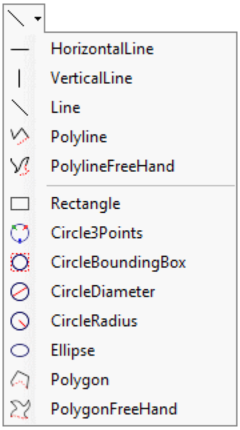
2.8.5. Profile – Histogram – ROI Toolbar (Dev version only)



This toolbar is enabled only when the **Developer** Task is selected. It is used to compute profiles and histograms, and to define a Region of Interest ("ROI") or multiple ROIs drawn in the graphics overlay. An ROI is a structure that enables execution of an image processing function on a sub-region (aka, sub-image) of an image. While some functions can only process rectangular regions, ROIs can be defined as virtually any shape using drawing tools provided on the toolbar. A profile or histogram is displayed in real-time in window E for the selected (i.e., highlighted) line or region.

Selection button : Used to select shapes previously drawn with one of the shape tools found on the toolbar. The selection is indicated by the appearance of handles (small white boxes) on the shape's bounding rectangle (for a circle or ellipse) or on the shape's boundary line.

Select a shape by left clicking on it. To select multiple shapes, hold the **Shift** or **Ctrl key** down while left clicking the shapes. Left clicking outside the selected shapes without holding down the **Ctrl key** will cause all selected shapes to become unselected. Any single shape can be individually unselected by left clicking on it. Unselected shapes can be selected as a group by using the selection tool and tracing a rectangle around the target shapes.

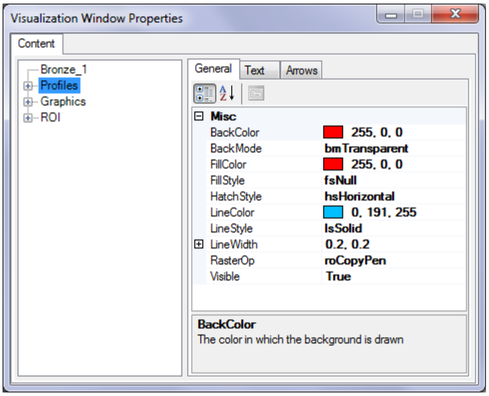


Shape button : Used to choose the shape type to be drawn (see shapes list in adjacent graphic). Straight lines can be drawn in any direction and have any length. Geometric shapes can be drawn to any size. Refer to section *2.9.5 Measurement Task, Contextual Toolbar – Interactive Drawing Tools* to get a detailed description of each shape.

Select the shape type by clicking on the shape button as described in the paragraph below.

When a shape is drawn, it is immediately selected and its handles appear, and the action corresponding to the selected operation mode (i.e., Profile or Histogram) is automatically performed. To alter the shape drawing, drag any of its handles. The angle of a line shape can be changed by dragging either one of its handles as needed. Any shape can be moved by grabbing its interior and dragging it to the desired location.

Note: The **Shape** button has to be clicked each time a shape is to be drawn, as the **Selection** button is automatically selected after a shape is drawn.



Misc

BackColor	🟥 255, 0, 0
BackMode	bmTransparent
FillColor	🟥 255, 0, 0
FillStyle	fsNull
HatchStyle	hsHorizontal
LineColor	🟦 0, 191, 255
LineStyle	lsSolid
LineWidth	0.2, 0.2
RasterOp	roCopyPen
Visible	True

BackColor
The color in which the background is drawn

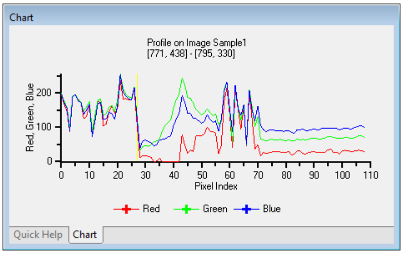
Each individual graphic object property can be altered. Select a graphic object and right-mouse click to open the Visualization Window Properties. To enable transparency, select the FillStyle as Solid, and add an alpha value (0 to 255) in front of the three default values of FillColor. For example, if the color is red and fully opaque, no alpha value is needed (255,0,0). If the color is red and 50% transparency is desired, insert alpha = 128 in front resulting in (128, 255, 0, 0). The alpha value affects all color channels equally.

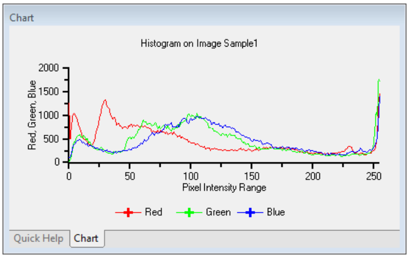
Note: The color of a drawn shape depends on the mode that was enabled when the shape was drawn. The four modes (i.e., Graphic, Profile, Histogram, and ROI) are mutually exclusive.

Delete button : Used to delete any shape or ROI or set of shapes or ROIs drawn in the graphics overlay using the **Shape** drawing tools. A shape or ROI must be selected before it can be deleted. Note that a shape and an ROI cannot be concurrently selected. Thus, selecting a shape will cause any selected ROIs to become unselected, and vice versa.

Graphic button **:** A click on this button enables the Graphic mode. All shapes drawn when Graphic mode is enabled are converted into a graphic overlay.

Profile button : Enables the **Profile** mode, which is used to compute and display in the Chart window a graph of the pixel intensity values along a straight line drawn in the Active Image. When the Profile mode is enabled, each time the user draws a new line shape or selects or modifies an existing Profile line shape in the Active Window, the profile of that line is updated in real-time in the Chart window. When the **Profile** mode is enabled, a blue box appears around the **Profile** icon on the **Menu bar**. Right clicking in the chart window displays a contextual menu that lets the user zoom in and out of a user defined region of interest in the Chart window. Message passing is active between the chart and the image. Select a range of values on the chart by pressing the mouse on the chart and holding down to define a range of intensity values. The corresponding section on the profile is highlighted with two vertical lines defining the two extremities of the section manually defined. Zoom in the chart to help refining the segment selection.





Histogram button : Enables the **Histogram** mode, which is used to compute and display in the Chart window a plot of the *number-of-pixels* versus the color or gray-scale *intensities* for pixels contained in a closed shape or along a line. When the Histogram mode is enabled, each time the user draws a new line or closed shape or selects or modifies an existing Histogram line or closed shape in the Active Window, the histogram of that line or closed shape is updated in real-time in the Chart window. When the **Histogram** mode is enabled, a blue box appears around the **Histogram** icon on the **Menu bar**. Right clicking in the chart window displays a contextual menu that lets the user zoom in and out of a user defined region of interest in the Chart window.

Note: Clicking on one of the color band labels at the bottom of the chart window results in display of a graph only for the selected color band. Clicking outside these labels restores the concurrent display of graphs for all three color bands. Also, the chart window's contextual menu allows the user to concurrently display any combination of the available bands.

ROI button ROI: A click on this button enables the ROI mode. Each closed shape drawn when the ROI mode is enabled is converted to a region of interest (ROI). When the **ROI** mode is active, a blue box appears around the **ROI** icon in the **Main Toolbar**.

Multiple ROIs of various shapes and sizes can be defined in an image. To add another ROI to an image, make sure the **ROI** mode is enabled. If it is not, click on the **ROI** button. Next, select the desired shape from the **Main Toolbar** and draw the ROI. ROIs are automatically saved with an image if the image is saved in TIFF or ASI format. However, ROIs are lost when an image is saved in all other formats.

Tip: To change the color of a ROI, select it with the button, and then right-click to open the Properties window. Select the target ROI in the Content panel of the window, and then select the ROI color attribute to change (e.g., LineColor) in the General panel. Change either the numeric values of the color bands or use the drop-down color chart.

Note: Any shape can be copied from one image (source) and pasted into another (target) image and converted into any mode (i.e., Graphic, Profile, Histogram, or ROI). Begin by selecting the desired shape in the source image and use the Ctrl+C sequence to copy that shape to the Windows clipboard. Next, make the target image the Active Image and then enable the desired mode for the target image. Finally, use the Ctrl+V sequence to paste the selected shape onto the new Active Image. When the desired shape appears after a few seconds, it will have the characteristics of the mode enabled for the target image and, if appropriate, the Chart window will be updated.

Help: An ROI can be used to create a new image from a portion of the Active Image. First draw the ROI on the Active Image, then use the *SubCopy* function in the *Process→Image→Utility* group to transform the ROI's contents into an image.

2.8.6. Task interface

The **Task Bar** lets the user select a task to be performed. The tasks are listed in a sequential order typical for accomplishing an image processing or analysis project. A user would typically execute them from top to bottom. But, depending on the user's needs, tasks may be skipped, repeated, or performed in a different ordering.

When a task is activated, a task-specific workspace is displayed that is composed of various interfaces. These interfaces are designed to facilitate the user's work for the imaging project step addressed by the task. Refer to section *2.9 Tasks*.

for more information on the tasks available in Aphelion.

A typical work session will usually involve the following tasks:

- **Camera settings** to set up the acquisition system, display a live image, and capture still images (only in Administrator mode);
- **Calibration settings** to define the resolutions (only in Administrator mode);

- **Image Acquisition** to receive a live image stream from a camera, during which one or more still images can be snapped and captured. When no camera device is available, Aphelion can receive images stored on local or network storage media;
- **Object Extraction** to quickly filter and segment the Active Image and generate objects;
- **Object Editing** to manually amend the **Object Extraction** results using the object editing tools provided;
- **Measurements** to compute a set of measurements for objects previously extracted (see note following);
- **Report Generation** to generate user-modifiable reports in MS Excel format and to export images to the Windows clipboard; and
- **Developer** (Dev version only) to provide the user with a wide range of image and ObjectSet processing functions. ObjectSets generated in this task can be edited in the **Object Editing** Task and can have attributes computed for their objects using the **Measurements** Task.

Note: The **Measurements** Task also provides tools for drawing interactively shapes in the graphics overlay. These enable the user to create and compute ad hoc measurements for the image and for extracted and drawn objects. For complex images where the **Object Extraction** process fails, it is possible to skip the two preceding tasks (**Object Extraction** and **Object Editing**) and immediately draw objects and compute associated measurements. The interactively drawn shape objects are independent of the objects created using the **Object Extraction**, **Object Editing**, and **Developer** Tasks.

2.8.7. Visualization Window

The Visualization window is used to display and manipulate images. This window has two image array mode buttons on the **Main Toolbar**: for the *One image* mode, which displays a single image (i.e., 1x1 array) and for the *Multiple Images* mode, which displays multiple images as an array of images. When the *Multiple Images* mode is selected, the Visualization window is automatically reconfigured into one or more panels, each configured to display M time N images. Sufficient panels will be created so that all of the 2D images in the Image Gallery will be displayed in sequence, beginning with the left-most image. Multiple panels are stacked in the Visualization window with reference Tabs appearing in the Status bar at the bottom of the window. Any panel can be displayed by clicking on its Tab or by double clicking on one of its images in the Image Gallery.

The banner of an open image window provides the name of its image at the far left end of the banner. If an open image is also an Active Image, the letter A is inserted to the left of the image name and its banner is highlighted.

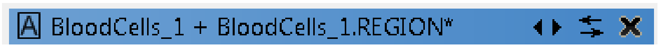
A close button is provided at the far right end of the banner. Clicking on the close button closes the image window in the **Visualization** window and removes its thumbnail from the Image Gallery. However, the image remains in Aphelion's active memory (i.e., RAM).

If there are multiple Active Images, then the banner for each of these image windows will also have a pair of opposite-pointing, horizontal arrows displayed adjacent to the close button. These arrows indicate that the Active Images will be synchronized with respect to zooming and panning operations.

In addition to the above, the banner of a color image will have two, juxtaposed arrowhead buttons pointing left and right. These can be used to display any single color channel of that image. As these arrowheads are clicked, the name of the displayed color band appears in parentheses to the right of the image name.

If one or more ObjectSets are open in the graphics overlay and were derived from an image open in the Visualization window, the name(s) of such ObjectSets will appear in the banner of that image.

Following is an example image widow banner for a color image having one open ObjectSet.



From left to right, the listed information is:

- Boxed letter A indicates an Active Image.
- Name of the image (BloodCells_1).
- Name of an associated ObjectSet with its spatial attribute appended (BloodCells_1.REGION*). If more than one ObjectSet is named, the last named ObjectSet is always the Active ObjectSet.
- Icons used to sequence through the bands (channels) of a multi-band image. For a color image (Red), (Green), or (Blue).
- Icon showing image synchronization is active.
- Icon to close the image.

Visualization Window Contextual Menu

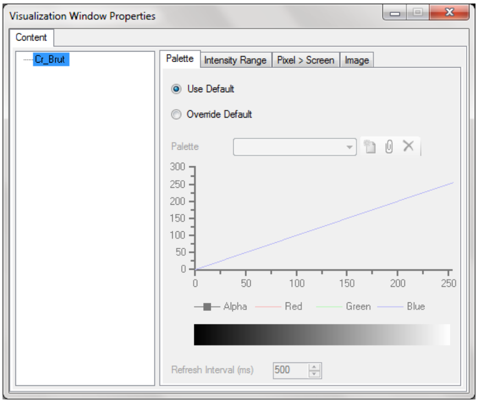
Right clicking in the Visualization window opens a pop-up menu. The entries in this menu vary depending on what Task is active on the Task Bar (window C). The following table specifies the contents of this pop up menu. These entries are described in the following table.

TASKS	Histogram	Statistics	Properties
Camera Settings	✓		✓
Calibration Settings			✓
Object Extraction			✓
Object Editing			✓
Measurements	✓		✓
Report Generator			✓
Developer	✓	✓	✓

Histogram – This entry computes the pixel intensity histogram for the Active Image and displays it window E.

Statistics – This entry computes the minimum, maximum, mean, skewness, and kurtosis values for the Active Image and displays them in the Output window located in window B. The Output window can be cleared of prior computations and other data by right-clicking the window and choosing Clear.

Properties – This entry displays a pop-up window that provides various display settings for the Visualization window, specifically for the Active Image and any Active ObjectSets displayed in the Visualization window.



Visualization Window Properties

Content: Cr_Brut

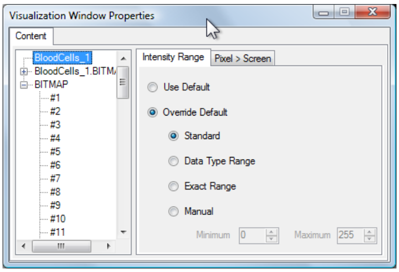
Palette | Intensity Range | Pixel > Screen | Image

(●) Use Default
( ) Override Default

Palette

Refresh Interval (ms) 500

To change the display properties of the items displayed in the Visualization window, right-click in the Visualization window. Select Properties. The Visualization Window Properties window opens (see example at right). This window's left panel lists the items currently displayed in the Visualization window (e.g., images, ObjectSets).



Visualization Window Properties

Content: BloodCells_1, BloodCells_1.BITM, BITMAP, #1, #2, #3, #4, #5, #6, #7, #8, #9, #10, #11

Intensity Range | Pixel > Screen

( ) Use Default
(●) Override Default
(●) Standard
( ) Data Type Range
( ) Exact Range
( ) Manual

Minimum 0 Maximum 255

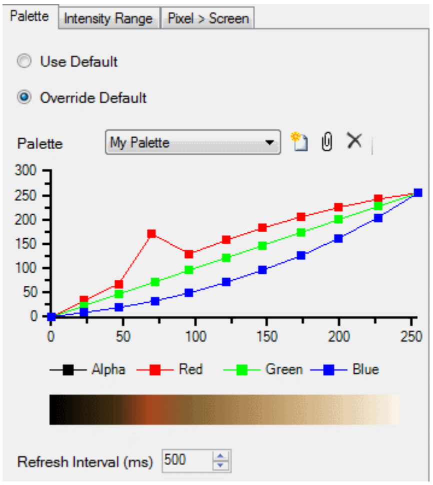
For a 2D image, only the Active Image and its displayed ObjectSets are listed. The example shows the Active Image, one ObjectSet, with an indexed list of its objects.

For a 3D image, since more than one rendering of the 3D image can be displayed concurrently, the items list will include all renderings concurrently displayed in the Active Image window. (For an explanation on how to display multiple 3D renderings in the same window and how to change their display settings, see the 3D Image Display Extension manual.

If an ObjectSet is listed (<*filename*>*.bitmap*), double-click on its name or click on its plus sign to see its contents. A numbered list corresponding to all of the objects in the ObjectSet is displayed. Display attributes for individual objects can be changed in the General tab of the right panel. To close the ObjectSet list, double-click on the on its name or click on its minus sign.

The contents of the right panels of the Visualization Window Properties window varies depending on the type of the item selected in the left panel (e.g., monochrome image, color image, 3D image, or ObjectSet).

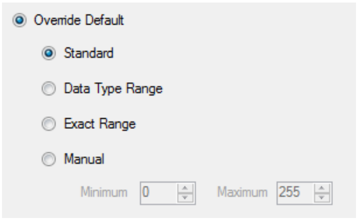
Palette Tab



When the selected Image is a monochrome Image, the right panel has three, stacked panels with tabs labeled: Palette, Intensity Range, and Pixel > Screen. Click on the Palette tab, and select Overwrite Default. The Palette dropdown list provides seven predefined palettes: Grey Scale, Inverse Grey Scale, Sepia, Rainbow, Saturation, Label, and Binary. To add a new palette to the list, click on the new palette button just to the right of the dropdown list. The New Palette dialog box opens. Enter a new name in the *Name* field. Any existing palette can be selected from the Copy From dropdown list and used as a template for the new palette. Next, click **OK**.

The selected palette template can be modified by repositioning control points along the graph lines for each color band. A graph line is formed by one or more straight line segments connected by control points, which are represented as small squares on the graph line. Control points can be added, deleted, or repositioned in the graphing area. To create a control point, position the mouse pointer in the graphing area, then right-click and select Add from the dropdown list, and then select the target color band. A small square appears on that color band's graph line directly above or below the pointer position. The control points can be individually dragged to form the desired graph shape. To delete a control point, right-click on the control point and select Remove, and then select the color band from which the control point should be removed.

Intensity Range Tab



The tab labeled *Intensity Range* provides various methods for determining which pixels will be displayed and what values will be assigned to the displayed pixels. *Intensity Range* limits the display of pixels to those pixels whose intensity values fall within a specified range of values.

Standard is the default behavior for most common imaging software products. Pixels having an intensity value of 0 are displayed in black and pixels having an intensity value of 255 are displayed in white. Values in-between produce more or less bright shades of gray. However if the range of the intensity values exceeds [0, 255] then the Exact Range strategy is applied (see below).

Data Type Range is based on the pixel intensity values data type. For example, if intensity values are specified as unsigned, 8-bit integer numbers, the intensity range will be 0 to 255; if as unsigned, 16-bit integer numbers, 0 to 65,535.

Exact Range stretches the image histogram by assigning an intensity value of zero (black) to the pixels having the minimum intensity value in the image, a value of 255 (white) to the pixels having the maximum intensity value, and proportionately equivalent values to the remaining pixels. This yields a gray-scale image having high contrast.

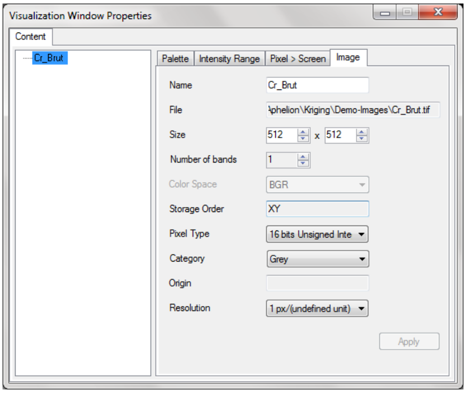
Manual option lets select which intensity value will be displayed in black (minimum), and which intensity value will be displayed in white (maximum).

The *Pixel > Screen* operation, sometimes called "transfer function" maps pixel intensities found in the image file to new values specified in a Look Up Table (or LUT). For example, pixels in an image file having intensity values less than 50 might be displayed as having intensities equal to zero. For each pixel value in an image (input value), a LUT will specify an output value for that pixel, which values may or may not be equal. Aphelion provides four transfer functions: Linear, Logarithmic, Gamma, and Modulo. The Gamma value can be adjusted if the Gamma function is selected. The Modulo function is used to display Label images.

Image Tab

If an image is selected in the left panel of this window, a list of image parameters is displayed in the Image Tab panel on the right. Some of these parameters can be modified here, while the others are only informational.

Name is the name of the image in the Aphelion interface, without its file type extension.

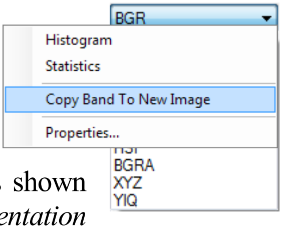


Name	Cr_Brut
File	\phelion\Kriging\Demo-Images\Cr_Brut.tif
Size	512 x 512
Number of bands	1
Color Space	BGR
Storage Order	XY
Pixel Type	16 bits Unsigned Inte
Category	Grey
Origin	
Resolution	1 px/(undefined unit)

File is the Window's path to the image file, ending with image's *Name* plus file type extension. The value of this parameter is empty if the image is not loaded from a file nor saved in a storage volume.

Size comprises two or three values, respectively for 2D or 3D images. The values are in pixel units, with the first value as the width (X axis), the second value as the height (Y axis), and the third value as the depth (Z axis). Increasing any of these values results in adding black pixels to the non-origin end of the corresponding axis. Decreasing any of these values results in discarding image pixels at the non-origin end of the corresponding axis. Remember that the origin is the left (X), upper (Y), near (Z) corner of the image.

Number of bands is the number of image bands. This parameter value is determined by the image category chosen (1 band for Binary, Grey, Label, Complex, and Frequency; 2 bands for Edge; and 3 bands for Color). It is automatically set when the *Category* parameter is altered and the Apply button is then clicked.



Color Space is the color representation of the image. By default, a color image is represented in the BGR (Blue, Green, and Red) color space. The *Color Space* parameter can be changed by selecting another color space in the pull-down list as shown beside. Refer to *APPENDIX H - Color Segmentation* to get a detailed definition of each color space.The visual appearance of a color image is the same whatever the *Color Space* value, but the appearance of each individual band component will vary by *Color Space* (use the browse ◀▶ buttons in the Visualization Window banner to display each band of a color image).

Tip: Right clicking in a single band of a color image displays a contextual menu. Selecting **Copy Band to New Image** extracts the current band into a new image. To view the original color image, browse through the color bands using the ◀▶ button.

Storage Order provides information on how the image is stored in memory. B stands for the band order of the color space, and XY means that the image is stored line by line. For example, if color space is RGB, the order of the bands in memory are red band followed by green band followed by blue band.

Pixel Type defines the data type of the pixels.

Category defines the image class. By default, a monochrome image has an 8 bit or 16 bit unsigned type, and a Grey category.

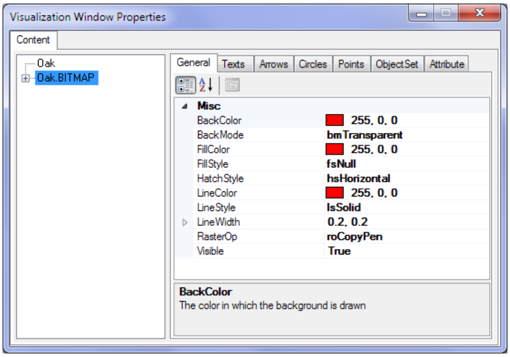
Origin is the coordinates of the pixel at the upper left corner of the image. By convention, the image origin point (x,y) for images is defined to be (0, 0) or (0,0,0) for 2D and 3D images, respectively. These origin points correspond to the left, upper corner of a 2D image. For a 3D image, the origin is the left, upper, near corner.

Resolution defines the image pixel size in real units. If the image's resolution was determined at the time of image capture and recorded in the image header, it will automatically appear in this parameter. If no resolution is displayed for this parameter, one can be selected from the *Resolution* drop-down list. If the None entry is selected, measurements will be calculated based on pixel counts.

After specifying values for the parameters, click on the Apply button to change the image properties and the display their effects on the image in the Visualization Window.

Tabs available when an ObjectSet is selected

When an ObjectSet attribute is selected, the right panel contains seven stacked panels whose tabs are labeled General, Text, Arrows, Circles, Points, ObjectSet, and Attribute. Each of these panels



Misc

BackColor	🟥 255, 0, 0
BackMode	bmTransparent
FillColor	🟥 255, 0, 0
FillStyle	fsNull
HatchStyle	hsHorizontal
LineColor	🟥 255, 0, 0
LineStyle	lsSolid
LineWidth	0.2, 0.2
RasterOp	roCopyPen
Visible	True

BackColor
The color in which the background is drawn

provides a list of parameters that affect some aspect of the display of an ObjectSet listed in the left panel. The initial values are the default values set in the Tools→Options→Advanced window. Clicking on any parameter will allow the user to make changes to its value(s). Some are changed by selecting a different value from a dropdown menu or a color palette, and others by typing in new numeric values. A parameter listed with a plus sign to the left will expand when clicked to reveal its sub-parameters. Click an item with a minus sign to the left will cause it to close and hide its sub-parameters.

Tip: Press the Delete key to reset a parameter to its default value.

General Tab

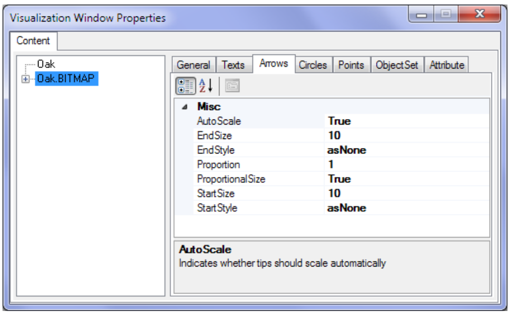
The General panel lists parameters that affect display of an ObjectSet highlighted in the left panel. These include colors and other visual attributes.

Text Tab

The Text panel lists parameters that determine the font used when text is displayed and the attributes applied to the font. Setting AutoScale to True will cause the font size to change on screen in proportion to the zoom factor.

Arrows Tab

The Arrows panel lists parameters that specify the appearance of lines drawn in the image overlay, including the existence of graphics at the start and/or end of a line (e.g., arrowhead, circle, and square).



Misc

AutoScale	True
EndSize	10
EndStyle	asNone
Proportion	1
ProportionalSize	True
StartSize	10
StartStyle	asNone

AutoScale
Indicates whether tips should scale automatically

Tip: Use the Arrows panel to place graphics at the extremities of a line (e.g., add arrowheads that point to areas or objects of interest).

Circles Tab

The Circles panel lists parameters that specify the appearance of circles drawn in the image overlay, including the presence of graphics at the center of a circle (e.g., dot, cross, cross hairs).

Points Tab

The Points panel lists parameters that specify the appearance of points drawn in the image overlay (e.g., dot, cross, cross hairs).

ObjectSet Tabs

The ObjectSet Tab provides information on the ObjectSet, including the name, the file path and name if the ObjectSet has been loaded from a file or saved in file, and the number of objects contained in the Objectset.

Attribute Tab

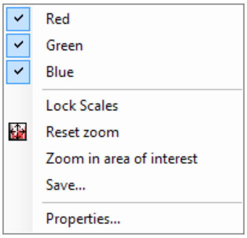
The Attribute Tab provides information on the ObjectSet including the spatial attribute type and the resolution. Note the resolution can be altered by clicking the arrow at the right of the resolution value and selecting one of the defined resolutions from the drop down list that pops up.

2.8.8. Chart Window

A chart computed in Aphelion is displayed in window E. The set of charts that are computed in Aphelion includes:

- Intensity profile along a line;
- Histogram of pixel intensities within an area of interest;
- Histogram of an attribute;
- Scatter plot between two attributes.

A contextual menu is available in the Chart window (see adjacent). The menu’s entries are tools to enable the user to explore the information provided in the chart. These tools include the ability to:



- Display any combination of the image's color bands;
- Lock the vertical and horizontal scales of the chart to prevent their changing when the data are updated;
- Zoom in to see finer detail in the data;
- Reset the chart to its original state;
- Save the chart as an XML file including the chart data and settings for future use in another program (e.g., C#);
- Access and alter the chart's display properties.

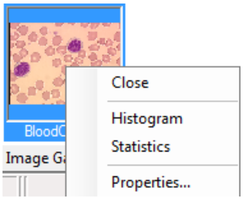
Message passing is available between a Profile chart and a corresponding profile line shape drawn on the image. Click on the profile graph in the Chart window to mark a reference point on the graph, denoted by a small +. A short, perpendicular line will appear on the drawn line shape at the location corresponding to the + in the Chart window.

A section of the profile graph can also be referenced by placing reference marks (+) at the two extremities of the section along the graph. Place the mouse pointer at one extremity, hold down the left mouse button and then drag and drop the mouse pointer on the other extremity. The + mark will appear at the section extremity points and the chart area between those points will be shaded. At the same time two short perpendicular lines will intersect the drawn line shape at points that correspond with the section extremity points on the graph.

Message passing is also available between a histogram chart, the measurement grid, and the ObjectSet overlaid on the image displayed in the Visualization window. Define a section on the histogram by dragging the mouse along the horizontal axis of the histogram. The corresponding objects belonging to that section are highlighted in blue in the grid.

2.8.9. Image Gallery Window

The Image Gallery window contains a thumbnail representation of each image currently loaded (i.e., opened) in the Aphelion environment. This window enables the user to quickly find an image and display it in the Visualization window by double-clicking its thumbnail. If a 3D image has been loaded in the Aphelion environment, then a 3D frame is displayed around the image's thumbnail. To display the 3D image in the Visualization window, select the Show entry in the thumbnail's contextual menu, and then the desired rendering method.



The contextual menu associated with the image thumbnail includes the following entries:

Close: To close the image. The image is removed from the visualization window, and if it was created during the work session, then a window pops up prompting the user to save the image.

Histogram: To compute the histogram of the whole image and to display it in the Chart window located in window E.

Statistics: To compute the minimum, maximum, mean, standard deviation, skewness and curtosis of the image and to display these values in the Output window located in window B. If the image is a color image, then these values are displayed for each color band.

Properties: To display the properties of the image in a window, as shown beside.

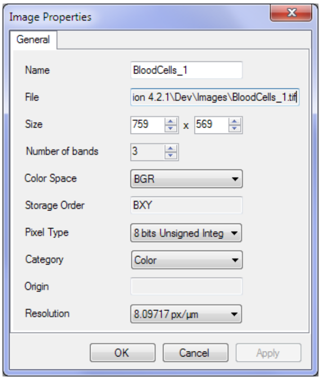


Image Properties

General

Name	BloodCells_1
File	ion 4.2.1\Dev\Images\BloodCells_1.tif
Size	759 x 569
Number of bands	3
Color Space	BGR
Storage Order	BXY
Pixel Type	8 bits Unsigned Integ
Category	Color
Origin	
Resolution	8.09717 px/µm

Note: Refer to the description of a similar window in the *2.8.7 Visualization Window - Image Tab* section.

Tip: To change the image color space, pull down the *Color Space* list, and select a new *Color Space*. Then click on the **Apply** button to apply the change. Refer to *APPENDIX H - Color Segmentation* to get a detailed definition of each color space.

2.8.10. ObjectSet Gallery Window

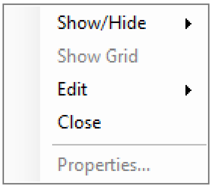
The ObjectSet Gallery window contains a common thumbnail representation ( ) for each ObjectSet or set of Interactive shapes loaded (i.e., opened) in the Aphelion GUI. This window enables the user to quickly find an ObjectSet and display it in the Visualization window by double-clicking its thumbnail. There are three methods to generate an ObjectSet:

- An ObjectSet created in the **Object Extraction** Task. Its name is the name of the image from which it was extracted;
- An ObjectSet created in the **Object Editing** Task. If there is not an Active ObjectSet when this task is used, the first object drawn will cause an ObjectSet to be created and it will be given the name of the Active Image. Otherwise, the drawn objects become members of the Active ObjectSet;
- An ObjectSet created in the **Developer** Task using its available functions. The ObjectSet's name is the name given by the user in the Output ObjectSet dialog.

Note: The Interactive shapes that a user draws while in the **Measurements** Task are not objects and do not form an ObjectSet. However, many Aphelion operations treat a set of Interactive shapes like an ObjectSet. Where there are differences in treatment, the differences will be noted.

A set of Interactive shapes takes the name of the source image for which they were created. The string "I_" is added at the beginning of the image name (e.g., "I_MyImage").

Right-clicking on an *ObjectSet* thumbnail opens a menu as shown below.



Close: Removes the ObjectSet from the ObjectSet Gallery, although the ObjectSet remains in Aphelion's memory until the user exits from Aphelion.

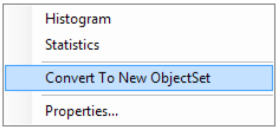
Show/Hide: Displays the ObjectSet in the Visualization window on top of the Active Image. Double-clicking on the ObjectSet thumbnail also has this effect. The Show/Hide entry has a submenu that enables the user to select the spatial attribute (e.g., *Bitmap*) to be displayed.

Show Grid (in **Developer** Task only): Displays in window B the grid(s) associated with the ObjectSet. The displayed grids are stacked with the Image Gallery and the ObjectSet Gallery and have identifying tabs.

Help: Only one grid of each type (Object Measurements, Object Global Measurements, Interactive Measurements, and Interactive Global Measurements), when computed, is displayed, even if more than one grid of each type has been computed, except in the **Developer** Task. The *Show Grid* entry in the contextual menu of an ObjectSet, available in the **Developer** Task only, allows displaying the grid of your choice, and all displayed grids are stacked.

Edit: Lets the user annotate the ObjectSet. It automatically switches to the **Object Editing** Task and enables all drawing tools as defined in *2.9.4 Object Editing Task*.

Properties: Displays the properties of an ObjectSet.



Tip: An ObjectSet can be quickly generated from a binary image. Right-clicking on the binary image displays a contextual menu. Select the entry **Convert to a New ObjectSet** to generate an ObjectSet derived from the binary image.

2.8.11. Status Bar

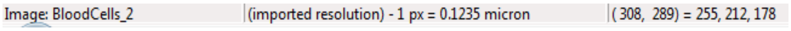


Image: BloodCells_2 | (imported resolution) - 1 px = 0.1235 micron | (308, 289) = 255, 212, 178

The Status bar, located at the bottom of the user interface, displays the following information, from left to right:

- Name of the Active Image or Active ObjectSet, depending whatever the Visualization window or a Grid window is selected;
- Resolution of the Active Image or Active ObjectSet whose name is displayed on the left;
- Mouse pointer coordinates (e.g., H=308, V=289) and pixel intensity values (e.g., R=255, G=212, B=178) at the displayed coordinates in the Active Image.

The pixel coordinates are always relative to the image under the mouse pointer, with the upper left-most pixel at 0,0 for a rectangular image. For color or multi-band images, each intensity value corresponds to a specific band (Red Green, Blue for color images and amplitude, phase, dx, dy for frequency images).

2.9. Tasks

2.9.1. Administrator Settings Tasks

The **Camera settings** and **Calibration settings** Tasks are the tools with which an Administrator specifies the value of acquisition control parameters that affect all users of Aphelion. Specifically, these parameters assure proper set up of image acquisition devices.

Upon successful entry of the Administrator password, a new version of the **Task Bar** appears that contains two new task icons corresponding to these settings tasks. These two settings tasks are described in detail below.

Important note: A user who has Administrator rights for Aphelion must also have Windows operating system rights to read/write the file ***AphelionDev.config*** for Dev or ***AphelionLab.config*** for Lab, located as follows:

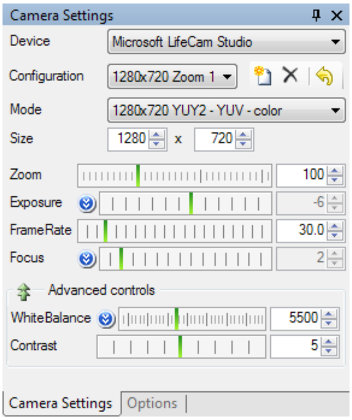
	Aphelion Lab	**Aphelion Dev**
XP	*"C:\Documents and Settings\All Users\Application Data\ADCIS\…*	
	…Aphelion Lab\4.x.y\AphelionLab.config"	*…Aphelion Dev\4.x.y\AphelionDev.config"*
Vista & 7	*"C:\ProgramData\ADCIS\…*	
	…Aphelion Lab\4.x.y\AphelionLab.config"	*…Aphelion Dev\4.x.y\AphelionDev.config"*

Camera Settings Task

The **Camera settings** Task provides interfaces used by an Administrator to specify parameters that configure image capture hardware (e.g., camera) to be used for the target project. Selection of the **Camera settings** Task opens a stack of two _jicontrol interface windows in window D and one display interface in window E.

Camera Settings (Tab) Interface

This interface is used to specify the active camera for the project and set its initial control parameter values. The *Device* parameter is used to identify the camera to be used for the project. If more than one camera is connected to the computer, then the user can select the desired camera from the *Device* drop-down list by clicking on its list display arrow. If no compatible camera[1] is connected to the computer, then the *Device* list will be empty.



Device: Microsoft LifeCam Studio
Configuration: 1280x720 Zoom 1
Mode: 1280x720 YUY2 - YUV - color
Size: 1280 x 720
Zoom: 100
Exposure: -6
FrameRate: 30.0
Focus: 2
WhiteBalance: 5500
Contrast: 5

Note: The *Device* parameter can only be defined in the **Camera Settings** task, which is only available in Administrator mode. Camera settings, which were previously selected by the Administrator, cannot be changed in the **Image Acquisition** Task.

The *Configuration* drop-down list is used to define a configuration file that contains all parameters defined in the window. It is used to associate a configuration with a device and a resolution. Click on the New icon to define a new configuration file. Every modification is automatically saved in the current configuration. The button allows to undo the last modification.

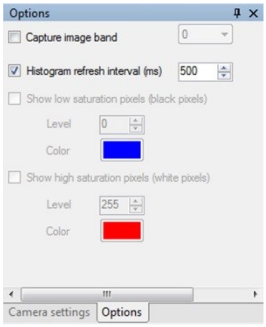
The *Mode* parameter specifies the image type (color or monochrome), the number of bits per pixel (8 or 16), the number of pixels per row and per column, and the acquisition rate (number of frames (i.e., images) per second captured by a video camera). If the camera has

[1] Refer to the sections *2.4.6 Camera interface installation* and *6.4 Hardware interfaces* for camera compatibility.

multiple modes, the user may select the appropriate mode for the project from the *Mode* drop-down list.

The remaining parameter values displayed are those presets for the selected *Device* and *Mode*. These can be adjusted by the user for the current project. The values defined using the Camera settings interface will become the default values for the **Image Acquisition** Task.

Options (Tab) Interface



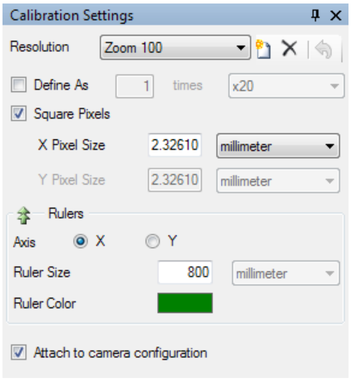
Options

- [ ] Capture image band: 0
- [✓] Histogram refresh interval (ms): 500
- [ ] Show low saturation pixels (black pixels)
  - Level: 0
  - Color: 🟦
- [ ] Show high saturation pixels (white pixels)
  - Level: 255
  - Color: 🟥

Camera settings | Options

The Options window (see right) opens when selecting the Options tab. It provides tools to control the quality of the image acquisition. The first checkbox provides the option to capture a specific color band (e.g., specify 0 for Red, 1 for Green, 2 for Blue). The second checkbox enables specification of the refresh interval of the histogram displayed in window E. The last two checkboxes, which are only available for monochrome images, are used to specify a graphics overlay in which low saturation pixels and high saturation pixels are displayed in colors selectable by the user.. The counter boxes can be used to modify the two pixel intensity saturation points, which are set to 0 and 255 by default.

Calibration Settings Task



Calibration Settings

- Resolution: Zoom 100
- [ ] Define As: 1 times x20
- [✓] Square Pixels
  - X Pixel Size: 2.32610 millimeter
  - Y Pixel Size: 2.32610 millimeter
- Rulers
  - Axis: [✓ X] Y
  - Ruler Size: 800 millimeter
  - Ruler Color: 🟩
- [✓] Attach to camera configuration

The **Calibration Settings** Task enables an Administrator to specify calibration parameters that are used to convert measurements from pixel units to real world units (e.g., microns). After these parameters are set, the measurement conversion happens automatically during the image analysis process.

The end result of the **Calibration settings** process is the computation of the acquisition system's resolution (i.e., camera, microscope, lens, and other optical components). This is done by determining the ratio between a known, real world distance and the number of pixels corresponding to that distance in a captured image.

To define a resolution, first the user must capture an image of an object of know dimensions (e.g., micrometer scale). Do this using the **Camera settings** Task and the **Snap** button on the task's **Contextual toolbar** (right edge of Workspace area). The "snapped" image will appear in the Visualization window. Return to the **Calibration settings** Task and click on the **New** button ( ) located between the **Resolution** drop-down button and the **Delete** button. In the box that pops up, enter a name for the calibration setting (e.g., to identify the acquisition system configuration) and then save the name.

Tip: The Live image can be used to define a resolution. Select the Live mode and then select the **Calibration settings** Task and process as described for a still image, except you can move the gauge under the camera to align it with the displayed ruler.

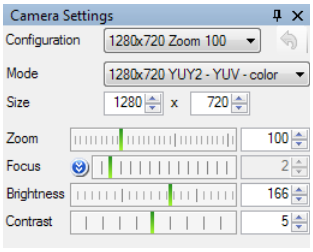
When an image acquisition system is symmetric in the X (horizontal) and Y (vertical) directions, the system is said to have *square* pixels. In this case, calibration needs only be done for the X direction. If the system is asymmetrical, then calibration must be performed for both directions. If the system is symmetrical, then checkmark the *Square Pixels* box; if it is asymmetrical, leave the box blank.

Next, a horizontal line is displayed as an overlay in the Visualization window. This line can be positioned and resized to precisely coincide with a known, horizontal dimension of the reference object. Be sure that the *Axis* radio button for *X* is selected and that the *Ruler Size* drop-down box is displaying the correct dimensional unit. Next, enter the real-world size of the reference object's horizontal dimension in the *Ruler size* box. The pixel size is now automatically computed and displayed in the *X Pixel Size* box. If you did not check the *Square Pixel* box, you will now have to calibrate in the Y direction by clicking on the *Axis* radio button for *Y* and then repeat the calibration procedure using the vertical line that appears on the screen. When the calibration steps have been completed, validate the calibration information (resolution) by clicking on the ✓ button. If you wish to redefine the current calibration, then click on the ↶ button. A validated calibration is saved when the user does any of the following: exits Aphelion and saves the configuration settings; invokes the menu **Tools→Save Administrator Settings**, or clicks on the **Yes** button when exiting the **Calibration Settings** task.

Click on the *Attach to camera configuration* option to associate the current calibration settings with the current *Device* configuration selected in the **Camera Settings** task. This option is only enabled when the calibration settings are defined for live images.

Important note: A saved resolution is specific to a) the camera and other optical equipment with which an image was captured, b) the distance between the captured object or scene and the camera, and c) the Mode of the camera when that image was captured. This resolution will not be valid if any of these factors change and cause a change in pixel size. If there is any doubt that the pixel size has not changed, the calibration should be performed again.
The Aphelion concept of a *camera configuration* requires the user to define an extended configuration that includes parameters such as the lens used to capture the image, the distance between the camera and the object, and other parameters that are not dependent on the camera. The user must select a camera configuration depending on all these parameters to define calibration settings. A camera configuration must be defined for each setting of the acquisition system.

2.9.2. Image Acquisition Task



Camera Settings

Configuration	1280x720 Zoom 100	
Mode	1280x720 YUY2 - YUV - color	
Size	1280 x 720	
Zoom		100
Focus		2
Brightness		166
Contrast		5

The **Image Acquisition** Task provides user tools for acquiring images with a camera connected to the computer.

Through the Camera settings control interface, the user can alter camera settings. The settings available to the user depend on the camera selected and the settings authorized to users by the Administrator. The graphic above shows an example. For additional camera settings information, see the **Camera settings** Task in section *2.9.1 Administrator Settings Task*.

The **Image Acquisition** Task also provides three display interfaces: the Visualization window that displays the Live or captured image, the Image Gallery that displays all open images, and a Histogram window. The Histogram window displays the histogram for the image currently being received by the camera. As the user changes camera settings, those changes can be seen as changes in the histogram.

Tip: Image Acquisition can only be performed when the current image displayed in the **Visualization** window is the Active Image (A is displayed on left of the window banner). Tool icons in the contextual toolbar are active in this case.

An image histogram plots the number of pixels (Y axis) that have a given intensity value (X axis). The intensity value ranges from 0 to 255 for a camera using 8-bit mode. Thus, increasing the brightness setting shifts the histogram plots from the left (pixels with lower intensity values) to the right (pixels with higher intensity values). While decreasing the brightness setting has the opposite effect on the histogram.

The *Gain* setting determines how much the signal delivered by the camera sensor is amplified. The signal/noise ratio is lower for a correctly exposed image with a high gain than for a correctly exposed image with a low gain. Increasing the gain stretches the histogram of intensities to the right, until saturation of the signal occurs.

The *Shutter* setting determines the length of time the camera holds the shutter open, which affects the image exposure. The higher the value, the brighter is the image. The value has to be less than the time between two image captures (i.e., the inverse of the number of captured images per second or capture rate).

A shading correction can be applied to the live image to enhance its exposure. To define an image as the Dark Current (DC) image, or the Flat Field (FF) image, click on the corresponding icon in the vertical toolbar. Once an image has been defined as the FF or DC image, a specific icon is displayed in the image banner. To apply a live shading correction on the received image, click on the corresponding icon.

Tip: To improve image quality, if the image is underexposed: 1) increase illumination of the scene, if possible, 2) increase the image exposure by increasing the shutter setting value, and 3) alter the gain value as needed. Taking these steps in this order helps to limit deterioration of image quality (i.e., limit the reduction of signal/noise ratio). If the image is overexposed: 1) decrease the gain value to have a suitable exposure, and if the image is still overexposed, 2) decrease the illumination of the scene, if possible, and 3) decrease the shutter setting value until the image is correctly exposed.

Important note: There are two ways to apply resolution profiles to images:

Resolution Tool Button – When a resolution is selected using this tool, that resolution is applied only to the Active Image. This button is on the **Main Toolbar**.

Default Resolution – After the *Default Resolution* is set (**Tools→Default Resolution**), it will be automatically applied only to images as they are snapped.

REMEMBER – The resolution must be changed each time the acquisition system is altered (e.g., a lens changed).

2.9.3. Object Extraction Task

The **Object Extraction** Task provides a pre-defined sequence of image processing functions that can effectively extract objects from a wide range of image domains. The user controls this processing sequence by specifying parameter values for the functions and observing a real-time preview of their object extraction results.

This task provides the user with specialized interfaces in windows D and E. Window D contains a set of three, sequential panels labeled as Object Extraction. It is with this window that the user specifies parameters for the object extraction processes. Window E is labeled Preview and it provides a real-time preview of object extractions resulting from the parameters chosen by the user.

The Preview window only displays that area of the Visualization window framed by a cyan rectangle. After clicking on this rectangle, the user can reposition it over a representative area of the image. The size and proportions of the rectangle can be adjusted, but only within the constraints of the Preview window's size and proportions. Since the Aphelion GUI gives its users great flexibility in configuring the size and proportions of its windows, it is possible to increase the size of the Preview window and then increase the size of the cyan rectangle accordingly. Note, however, that increasing the size of the Preview window may adversely affect its real-time character.

Objects are extracted in the form of Bitmaps. Parameters the user specified in the Object Extraction panels determine how the image's pixels will be aggregated to form those Bitmaps (i.e., objects). Red outlines of the objects are displayed in the Preview window as a graphics overlay on the previewed portion of the image. When a parameter or the preview frame is changed by the user, the effects are shown immediately in the Preview window as changes in the object outlines.

When the user is satisfied with the parameters chosen, clicking on Apply causes the object extraction processes to be applied to the entire image and the outlines for all extracted objects to appear as a graphics overlay in the Visualization window. The user can iteratively vary parameters, check the Preview window, and then see the results applied to the full image.

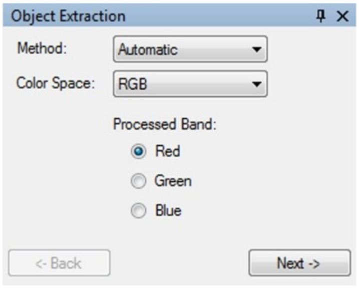
Object Extraction Parameter Settings (window D)

Two methods of object extraction are provided: Automatic and User-assisted. Both methods use a three panel sequence to specify parameter settings prior to applying the method to an image. The panels 1 and 2 are different for the two methods, but panel 3 is the same for both methods.

Note: When the Object Extraction task is selected, if the active Method is not the method you want, use the **Back** button to return to panel 1 and change the method type there using the Method dropdown menu.

Automatic Method

Panel 1 - If the active Method is not *Automatic*, select *Automatic* from the Method dropdown menu.



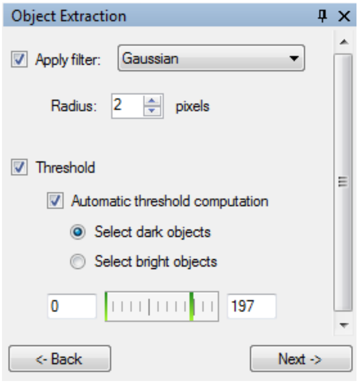
Method: Automatic
Color Space: RGB
Processed Band: [✓ Red] Green Blue

If the Active Image is a gray-scale or binary image, Panel 1 will have no parameter settings. Click on **Next**.

If the Active Image is a color image, the user specifies the Color Space in which the object extraction is to be performed by selecting from the Color Space dropdown menu. Next, select the band to be processed by clicking one of the radio buttons listed under Processed Band. Usually, best results are achieved by selecting the band in which the target objects have the most contrast with their backgrounds in the image. This contrast can be seen in the Preview window. Then click on the **Next** button to open the second panel of the **Object Extraction** Task.

Panel 2 **-** This panel provides parameter settings for two functions:

- *Smoothing Filter* to compensate for excessive image noise
- *Threshold Segmentation* to identify objects in the image based on pixel brightness.



Object Extraction

[✓] Apply filter: Gaussian

Radius: 2 pixels

[✓] Threshold

[✓] Automatic threshold computation

(●) Select dark objects

( ) Select bright objects

0 — 197

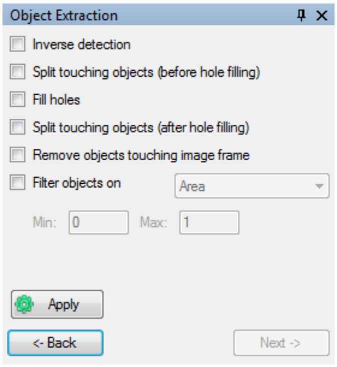
If the user selects the *Apply filter* checkbox, a Gaussian filter is performed. This filter calculates a weighted average of the pixel values within a neighborhood whose size is determined by the *Radius* counter value in the dialog box. The intensity of the center pixel in the neighborhood is assigned that weighted average value. The higher the radius value, the greater the smoothing effect on the image, and the blurrier the resulting image will appear. The smoothing effect reduces the impact of noise in the image, but it also alters the position of object boundaries.

If the user selects the *Threshold* checkbox, a threshold segmentation will be performed that will identify objects based on their brightness or darkness. The user can choose either automatic or manual setting of the lower and upper thresholds for pixel brightness. If the user selects the *Automatic threshold computation* checkbox, the user must next choose either *Select dark objects* or *Select bright objects* by clicking on the associated radio button. Aphelion will then calculate threshold settings based on intensity values of the pixels displayed in the Preview window. Only pixels whose intensity values fall between these thresholds will be assigned to objects.

However, if the user manually changes either of the threshold values, the *Automatic threshold computation* checkbox is cleared, and the *Dark* and *Bright* objects options are disabled. Threshold values can be manually changed by editing the number in the threshold boxes or by dragging the vertical green lines displayed on the dial, left line for the lower threshold and right line for the upper.

Note that whenever the filtering or threshold parameters are changed, the results are immediately displayed in the Preview window.

Note: When Automatic threshold computation is selected, the min and max threshold values are computed based on the contents of the Preview window. Thus, as the area displayed in the Preview window is changed, the min and max values will change accordingly. When the **Apply** button is pressed in Panel 3 (see following section) the min and max values for the last Preview window are used for the entire image.


Object Extraction
Inverse detection
Split touching objects (before hole filling)
Fill holes
Split touching objects (after hole filling)
Remove objects touching image frame
Filter objects on
Area
Min: 0
Max: 1
Apply
<- Back
Next ->

Panel 3 - Object Extraction's third panel is identical for both *Automatic* and *User-assisted* methods. It provides further opportunity for refining the objects extraction process. When the extracted objects are viewed in either the Preview or Visualization window, one might see that some objects are touching or have holes in them. Also, some objects may only be partially displayed because they intersect with the image frame. These conditions might limit the quality of the object extraction results. For example, touching objects might be extracted as a single object. Partially displayed objects will distort extraction statistics. Panel 3 provides a sequence of functions that complement the Panel 2 functions and can significantly enhance the object extraction results.

Checkboxes are provided for selecting processes for filling holes in objects, splitting touching objects, removing objects that are touching the image's frame, and filtering objects (i.e., removing them) based on one of its attribute values. Note the selected processes are applied sequentially from top to bottom.

Inverse detection – The result of the detection after applying the operations available in Panel 2 is inverted. In other words, the Not of the detection is computed and used as the input of the subsequent process. Note that this inversing process is not observable in the Object Extraction Task.

Fill holes – A hole is an arbitrary set of contiguous pixels located totally in the interior of an object that are not in the object's Bitmap. Generally, holes are either part of the object or are an image artifact that obscure or alter that part of the object in the image. The user must decide if the holes should be treated as part of the objects that contain them. Selecting the *Fill holes* process adds into the object's Bitmap the pixels that compose the hole(s) in the object. While checking the *Fill holes* box will cause all holes to be filled, this process also affects the results from the *Splitting touching objects* process (described next).

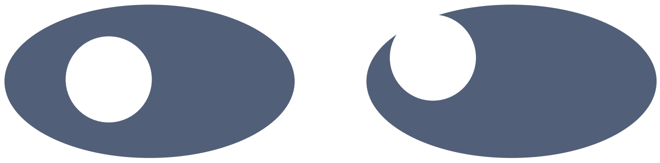


Figure 2.9-1: Left object includes one hole. Right object has no hole because the white area inside the object is not totally surrounded by the object

Split touching objects – When objects are touching or overlapping in an image, the *Filter and Threshold* processing will likely cause them to be segmented as a single object. To get the best results, such objects need to be split into separate objects. The two *Split touching objects* processes will define a boundary line between touching/overlapping objects transforming the single object into its elementary components.

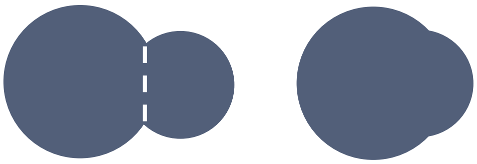


Figure 2.9-2: Left object is divided into two objects by the white boundary while the right object remains a single object when the Split touching objects processing is applied

Results from the two alternatives for splitting touching objects can differ significantly based on the source image and if and when the *Fill Holes* option is performed. The combinations of functions possible with the first three checkboxes are:

Split touching objects (before hole filling) – First the touching objects are split, then all holes are filled provided that the *Fill holes* box is checked. If the *Fill holes* box is not checked, then only the *Splitting touching objects* operation is performed.

Split touching objects (after hole filling) – First, provided that the *Fill holes* box is checked, all holes are filled and then the *Split touching objects* operation is performed.

The following table describes the possible behaviors of the checkboxes related to hole-filling:

Table 2.9-1: Results of the segmentation according to the processing options

SPLIT TOUCHING OBJECTS CHECKED?	FILL HOLES CHECKED?	RESULTS
Yes (before hole filling)	Yes	Split then fill
	No	Split only
Yes (after hole filling)	Yes	Fill, then split
	No	Split only
No	Yes	Fill only
	No	None

Remove objects touching image frame – An object is said to be touching the image frame if any pixel in the object's Bitmap is on the boundary of the image. This process simply deletes that object's Bitmap.

Filter objects on – Filtering an object is the process of deleting that object based on a geometric property of the object. After the check-marked processes described above have completed, the *Filter objects on* process is executed, provided that it has also been check marked. The *Filter objects on* process is applied based on the criterion selected from its drop-down menu and the *Min* and *Max* measurement values specified. The definition of these measurements can be found in paragraph *4.2. Object Measurements*.

For example, to remove small objects whose area is less than 500 pixels, select *AREA* as the measurement, and enter 500 in the *Min* box and the maximal object size in the *Max* box. The maximal object size is usually the size of the image, 1 million pixels for a 1024x1024 image. If the image is calibrated (has a resolution) then the Min/Max values should be entered in real-world units rather in pixels.

Tip: The order of the *Split touching objects…* and the *Fill Holes* processes can have a significant impact on the result of the object extraction. If objects are spread over the image, and if there are holes inside some of these objects due to the segmentation or the presence of other objects with different pixel intensities, then first perform a *Fill Holes* operation and next perform the *Split touching objects (after hole filling)* operation.

If objects appear to be clusters of smaller objects with holes inside those clusters due to gaps between touching objects, then first perform a *Split touching objects (before hole filling)* operation and next a *Fill Holes* operation. In the latter case, filling the holes first causes the splitting operation to make unusual (but explainable) errors.

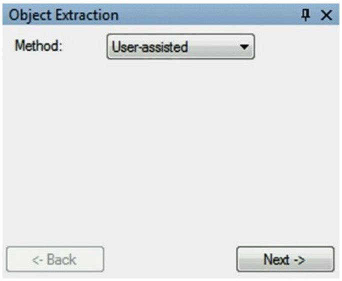
User-assisted Method

The User-assisted Method for **Object Extraction** provides the user with four segmentation methods, one of which is included with the Aphelion Dev product and the other three are available in the optional

Dev extension called Color Segmentation. The included segmentation method is called Color Interval and it is described in the following paragraphs.

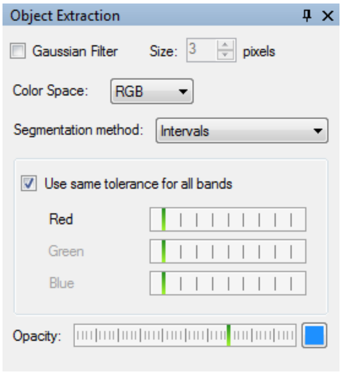
The segmentation methods included in the optional Color Segmentation extension are described in *APPENDIX H - Color Segmentation* of this user guide. The three methods are called Color Distance, Region Growing, and Morphological Partition. While Color Interval is a generally useful technique for segmentation of color images, the three additional methods significantly enhance Dev's object extraction capabilities for a wide range of image domains. Refer to the Quick Help information displayed in Window E of the user interface to learn more about the segmentation methods that are available and the settings of associated parameters.

Panel 1 - If the Method selected is not *User-assisted*, choose *User-assisted* from the Method dropdown menu. There are no other parameter settings for this panel. Click the **Next** button.


Object Extraction

Method: User-assisted

Panel 2 - Four different segmentation methods are available to the user from the Segmentation method drop-down menu. The contents of this panel will change depending on the segmentation method chosen. Three parameters are common to all four methods and they are described next.


Object Extraction

Gaussian Filter [ ] Size: 3 pixels

Color Space: RGB

Segmentation method: Intervals

[✓] Use same tolerance for all bands

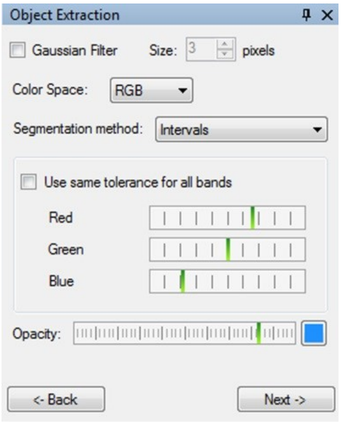
Gaussian Filter - This is a smoothing filter that is used to compensate for excessive image noise. If the user selects the *Gaussian Filter* checkbox, a Gaussian filter is performed. This filter replaces the color intensities of each pixel with the weighted average of the color intensities of the pixels in a circular neighborhood whose radius is the number of pixels specified in the Size parameter and whose center is the modified pixel. The larger the radius, the greater the smoothing effect on the image, and the blurrier the resulting image will appear. The smoothing effect reduces the impact of noise in the image during the segmentation process. But, it also alters the position of object boundaries.

Color Space – If the Active Image is a color image, the user specifies the color space in which the object extraction is to be performed. The color space is selected from the Color Space dropdown list which provides these options: RGB, HSI, Lab, YUV, LCH, YIQ, and XYZ.

Clear Selected Regions – Delete regions that are currently defined after performing a manual segmentation.

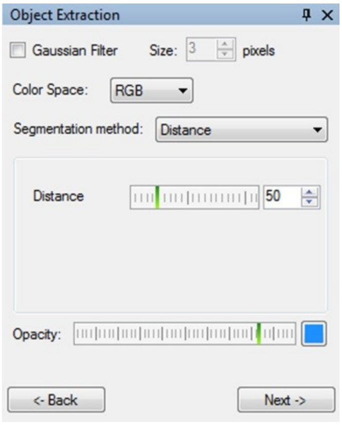
Opacity – Each object found in an image is represented by a region of color overlaid on the source image. The Opacity slider control is used to specify the degree to which the source image can be seen through the graphics overlay. Clicking on the color button seen at the right side of the slider enables the user to select from a palette of colors for the overlay.

Segmentation method – This parameter is a dropdown list containing four segmentation methods. The first method, Color Interval, is always available in Aphelion Dev. The three other methods are only available in the optional, Color Segmentation Extension. Depending on which method is selected by the user, the list of parameters displayed changes. The term reference point is used in the segmentation method descriptions. A reference point is a pixel neighborhood the user selects by clicking in the interior of a region of interest in the Active Image. The size of the neighborhood can range from a single pixel to multiple pixels, depending on the settings found at **Tools→Options→Advanced→Behavior→ObjectExtraction**.

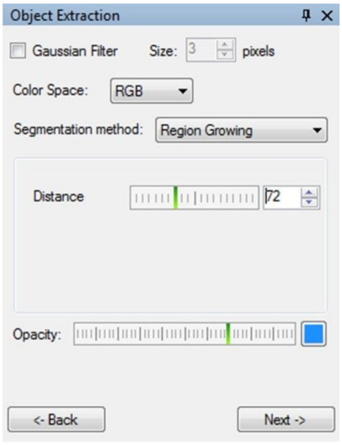


Method: Interval – This method is based on the processing of the three color bands independently. The color value is defined by clicking on a reference point inside an object to be detected. A threshold is then performed using the color band values of the reference point, plus and minus the respective band tolerance values defined by settings of the three color band sliders. A checkbox is provided that causes the lower two sliders to be locked to the setting of the top slider. The Opacity controls are explained separately below

Note: It is often the case that the interesting objects in an image have more than one significant color interval. After selecting the first reference point and setting the interval sliders, additional color intervals can be added to the segmentation. This is done by control-clicking on additional reference points as needed. Thus, each added color interval forms a union of intervals with the first color interval.



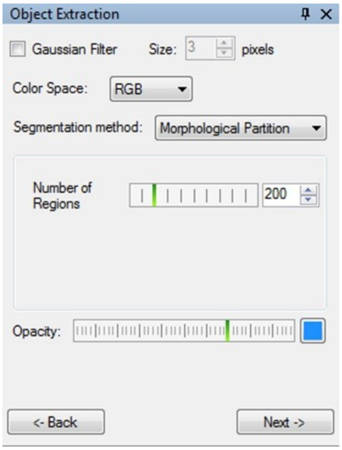
Method: Distance – This method defines an object as being a set of connected pixels such that each pixel is less than or equal to a specified color distance from a reference point. The user specifies the distance value by adjusting the Distance slider. Next the user clicks in the interior of an object to specify a reference point in the object. Based on the color of the reference point and the Distance setting, all pixels in the image that are within the specified color distance of the reference point will be marked as part of an object. The Distance slider can then be adjusted to include more or fewer pixels as being part of objects. If the object interiors have more than one color of significance, additional reference points can be specified to increase the number of pixels that meet the color distance requirement to a reference point. This is done by holding down the control (Ctrl) key while clicking to specify another reference point.



Method: Region Growing – This method of segmentation is based on manually specifying object seeds inside objects of interest using a mouse left-click or specifying non-object areas using a shift-left-click. Each seed point is an initial region in the image of usually a few pixels. The color of an initial region is the average of the pixels included in that initial region. Such seeded or initial regions are grown larger through the user's changing the color Distance parameter or placing additional object and non-object seeds on the image. The region growing algorithm processes each seeded region in sequence, adding to the region those pixels adjacent to the region's boundary pixels and that are not part of another seeded region, and whose color is within the Distance value from region's average color. This sequential processing of seeded regions is repeated until no pixels remain that are qualified to be part of one of the seeded regions. The user continues the interactive process of placing seeds in object and non-object regions until all of the desired objects are properly extracted and no non-object regions are extracted.

Method: Morphological Partition – This segmentation method is based on a morphological watershed operation that partitions an image into N regions such that each regional boundary line corresponds to an image edge (i.e., a gradient sign change in a three by three neighborhood). The *Number of Regions* parameter defines the maximum number of regions to be partitioned by the operation.

When the mouse pointer is moved into a region, the corresponding area in the object overlay turns green to preview the object. It remains green until the mouse pointer is moved out of the region.



The user must manually select all of those regions that correspond to objects of interest in the image. This is done by left-clicking or control-left-clicking in a green object preview. When the mouse pointer is moved from the green object preview after its corresponding object has been selected, its color changes to the color indicated by the Opacity color box.

If an object selection was done with a left-click, all previously selected objects become unselected. If an object selection is done with a control-left-click, all previously selected objects remain selected. To unselect an individual object, control-left-click that object. To unselect all selected objects, left-click in any object or region, which leaves that region's object selected. Then control-left-click that object to unselect it.

Note: Refer to *APPENDIX H - Color Segmentation* for more information about the four segmentation methods and the color space definitions.

Panel 3 - This panel is identical to the Panel 3 described in section *2.9.3 Object Extraction Task - Automatic Method.*

Note: By default, objects generated in the **Object Extraction** Task are defined in 8-connectivity. In other words, for the same object, neighbored pixels are either connected side by side or a corner. Object connectivity can be modified by editing the *ObjectExtract →ConnectivityGraph* parameter in the Advanced tab of the **Tools→Options** menu.

2.9.4. Object Editing Task

The **Object Editing** Task enables the user to improve on the **Object Extraction** Task results by manually altering those results. It can also be used on ObjectSets generated in the **Developer** Task or to create a new ObjectSet starting from an empty one. The **Object Editing** Task's **Contextual toolbar** provides editing tools that enable the user to split touching objects, connect (i.e., merge) unconnected objects, modify object boundaries, delete objects, and create new objects. When this task is selected, the objects outlined in the graphics overlay are automatically filled-in. An icon ( ) on the **Object Editing** Task's **Contextual toolbar** toggles the objects between filled and not filled states.

Tip: Zooming in the image will aid in drawing more accurate objects.

Object Editing Contextual Toolbar – Following is a description of the **Object Editing** contextual toolbar.

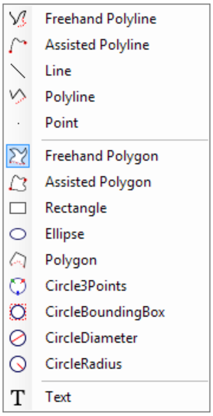
Select: Used to select an object. Point at an object then left click. The selection is indicated by an outline of the object's bounding rectangle. To select multiple objects, hold the **Ctrl** or **Shift key** down while left clicking the objects. Alternatively, hold the left mouse button down and draw a rectangle encompassing or touching objects to be selected. Left clicking anywhere other than on a selected object without holding down the **Ctrl** or **Shift key** will cause all previously selected objects to become unselected.

Draw: Used to modify objects or draw new objects (add points, lines, curves, or polygons depending on the geometry tool selected). By default, *Freehand Polygon* is selected.

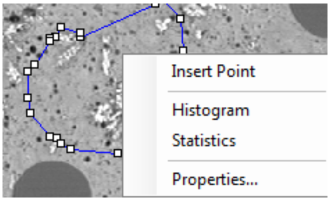
Erase: Used to modify or delete objects (remove points, lines, curves, or polygons depending on the geometry tool selected). By default, Freehand Polygon is selected.

Fill: Used to fill any contiguous background area with the color currently selected for objects. Can be used to fill holes in objects. Click anywhere inside the targeted background area. Note that if background fill causes two or more objects to share a contiguous boundary, those objects will be treated as merged into a single object.

The following geometric tools are used in conjunction with the **Draw** and **Erase** tools to edit a Bitmap. One of those tools needs to be selected first. In the **Contextual Menu**, click on the **shape** icon to pull down a list of available drawing tools.



Free-hand Polyline: To draw or erase a curved line. Click on the drawn object with the Select tool to display control points on the object boundary. Right-clicking then opens a contextual menu to let the user insert/ remove points.



Assisted Polyline: To help the user draw or erase a curved line. The drawing is automatically adjusted to the sharpest (most significant) edge and it then follows that edge. Left click to drop a point where the curved line is diverging too much from the edge. Double click at the end of the curved line to complete the drawing.

Note: After clicking on the first point of the shape, Aphelion Dev computes the optimal path. An hour glass icon is displayed during the computation.

Line: To draw or erase straight lines. Click on the icon just below to specify the line thickness.

Polyline: To draw or erase a polyline. Left click to define a vertex between two straight lines. Double click at the end of the drawing to complete the drawing

Point: To draw or erase objects made of one or more pixels. Click on the icon just below to specify the point size defined in pixels.

Free-hand Polygon: To draw or erase a closed curve defining an arbitrary shape. Click on the drawn object with the Select tool to display control points on the object boundary. Right-clicking then opens a contextual menu to let the user insert/remove points.

Assisted Polygon: To help the user draw or erase a curve to define an arbitrary shape. The drawing is automatically adjusted to the sharpest edge and it then follows that edge. Left click to drop a point where the curve is diverging too much from the edge. Double click at the end of the line to complete the drawing.

Rectangle: To draw shapes of type *rectangle*. The rectangle is defined by two, diagonally opposite corners. Move the mouse pointer to the location of the first corner. Press and hold down the left mouse button down to mark that corner. While holding the left button down, move the mouse pointer to the location for the second corner. A preview of the rectangle is shown as the mouse pointer is moved from the first to the second corner. Releasing the button marks the second corner and displays the final rectangle.

Ellipse: To draw shapes of type *ellipse*. An ellipse is uniquely defined by its bounding rectangle and, therefore, by any pair of the bounding rectangle's diagonally opposite corners. Move the mouse pointer to the location of the rectangle's first corner. Press and hold down the left mouse button to mark that corner. While holding the left button down, move the mouse pointer to the location for the second corner. A preview of the ellipse and its bounding rectangle are shown as the mouse pointer is moved away from the first corner. Releasing the left button defines the second corner and displays the final ellipse.

Polygon: To draw shapes of type *polygon*. A polygon is a closed shape comprised of three or more straight line segments (i.e., sides) such that each endpoint of a line segment intersects exactly one other line segment endpoint. Move the mouse pointer to the location of the first endpoint and click the left mouse button to define the first endpoint. Move the mouse pointer to the location of the second endpoint and click the left mouse button to define the first line segment (i.e., side). Repeat this line segment drawing process as needed to complete the polygon. A preview of the polygon begins as the second line segment is drawn. A double click of the left mouse button completes the polygon drawing. (Note that if you allow any side in a polygon drawing to intersect any other side more than twice, you will have drawn multiple, connected polygons, although they will still be treated as a single shape.).

Circle by 3 points: To draw shapes of type *circle* using three points on its circumference. This method is based on the geometric property that a circle is uniquely defined by any three points on its circumference. Each time the left button of the mouse is clicked, a point is defined. A preview of the circle is shown as the mouse pointer is moved after the second point is defined. When the third point is defined by a click, the final circle is displayed.

Circle by Bounding Box: To draw shapes of type *circle* based on the bounding box of the circle. A circle is uniquely defined as the largest circle completely contained in the drawn square. Move the mouse pointer to the location of the square's upper left corner. Click and hold down the left mouse button to mark that corner. While holding down the left button, drag the mouse pointer away from the upper left corner. The square and circle are previewed as the mouse is moved. When the circle is the desired size, release the left button to display the final circle.

Circle by Diameter: To draw shapes of type *circle* based on the circle's diameter. A circle is uniquely defined by the end points of any of its diameter lines. Move the mouse pointer to the location of the first end point. Click and hold down the left mouse button to mark that end point. While holding the left button down, move the mouse pointer to the second end point. The diameter and circle are previewed as the mouse pointer is moved. Release the left mouse button to mark the second end point and display the final circle.

Circle by Radius: To draw shapes of type *circle* based on its radius. A circle is uniquely defined by the end points of any of its radius lines. Move the mouse pointer to the location of the circle's center point. Click and hold the left mouse button down to mark the radius' first end point (i.e., circle's center). While holding down the left button, move the mouse pointer to the second radius end point (at the circle's circumference). The radius and circle are previewed as the mouse pointer is moved. Release the left mouse button to mark the radius' second end point and display the final circle.

T **Text:** To add text on an image. Select the **Text** entry in the Shape dropdown list and then click in the image. The Enter Text window opens. Enter the desired text in the empty box at the top of the Enter Text window. Alignment of the text can be changed using the Alignment dropdown menu. The text is then moved into the image depending on the options selected and the location of the mouse cursor. The Rotation Angle counter can be varied ±180º. If the Use default value box is checked, the font of the entered text will be Arial and the size and attributes of the font will be those last selected by a user. To change the font type, size, and attributes, clear the Use default value checkbox and click the **Font** button to access a font selection dropdown list. Check or clear the Auto Scale box to define if the text follows a zoom action or not. If auto Scale is set to False, the size of the text is not changed when zooming in the image. Click the Apply button to view the effects of the changed settings. After the Enter Text dialog box has been closed, the displayed text can be moved by first selecting the text (i.e., click the Select button on the main toolbar and then click the text) and then drag the text to the desired location. The Edit Text dialog box can be reopened by selecting text and then right clicking to access the contextual menu. From that menu, choose the Edit Text entry.

Note: By default, any shape drawn in the **Object Editing** Task is defined in 4-connectivity. For instance, a single line is composed of a set of pixels that are 4-connected, i.e. two pixels that are connected through a corner are not connected. Connectivity defined for drawn objects can be modified by editing the *Editing→Connectivity* parameter in the Advanced tab of the **Tools→Options** menu.

The following tools are used to remove either the selected objects or not selected objects.

Delete selection: All selected objects are deleted and all unselected objects are kept. This tool is active if at least one object is selected.

Invert Selection: Inverts the selection status of all objects in the ObjectSet.

The following tools change the display of the Bitmap.

Color: Change the color of the drawn object. Select a color and click the OK button.

Opacity: Change the transparency of the drawn object. Move the slider to alter the transparency value.

At the end of the editing process, the corresponding ObjectSet is automatically generated to reflect edits that have been made in the image overlay. Once the **Object Editing** Task is completed, the existing measurements grid should be manually updated by returning to the **Measurements** Task and re-clicking the appropriate object measurement tools. It is also a good idea to save this edited ObjectSet for future analysis and to avoid having to repeat the manual object editing steps.

Note: The Object Measurements grid is automatically updated when objects are removed using the button. The rows corresponding to the removed objects are removed from the grid. If objects were only deleted, then it is not necessary to recompute the Object Measurements. If new objects were inserted or existing objects modified, then the Object Measurements must be recomputed.

However, the Object Global Measurements must be recomputed whenever the ObjectSet is modified in any way.

2.9.5. Measurements Task

The **Measurements** Task provides a set of tools that enables calculation of attribute measurement values for either the objects resulting from the **Object Extraction** Task and/or the **Object Editing** Task or the interactive shapes drawn in this task. These tools include drawing tools used to create special-purpose objects that we refer to as "shape objects" or just "shapes." Shapes have several versatile forms that enable a user to specify ad hoc measurements that may not be possible with the standard measurements that were pre-selected by an Administrator user. Like objects, shape objects also have attributes.

The attribute measurements for objects and shapes are computed and then displayed in numeric arrays called grids. Four different grids are provided that display: Object Measurements, Object Global Measurements, Interactive Shape Measurements, and Interactive Shape Global Measurements. Object attribute values can also be displayed as histograms and scatter plots in the task's Chart window.

In addition to computing and displaying object measurements, the **Measurements** Task can also output the measurements grids in both MS Excel and Comma Separated Values (CSV) formats.

Performance of the **Measurements** Task's capabilities is provided through its **Contextual toolbar**, located at the right edge of the Workspace. The tools provided there are grouped by related functions as follows: **Compute Tools**, **Data Display Tools**, **Data Export Tools**, and **Interactive Drawing Tools**.

Please review section *2.7 Terminology* for detailed descriptions of objects, attributes, ObjectSets, Active ObjectSet, and grids.

Contextual Toolbar – Compute Tools

Each of the four **Compute Tools** computes and displays the measurements for one of the four grid types. The measurements computed are those previously selected in the **Tools** menu by an Administrator. The computations are performed on the objects created with the **Object Extraction**, **Object Editing**, and **Developer** Tasks, and the shapes drawn interactively with the **Measurements** Task's **Interactive Drawing Tools** described below.

Object Measurements: To compute and display individual attribute measurement values for each object in the Active ObjectSet that was generated by the **Object Extraction**, **Object Editing**, or **Developer** Tasks and display them in the Object Measurements grid in window B.

Object Global Measurements: To compute and display attribute measurement values aggregated for all objects in the Active ObjectSet, plus one measurement value for the ratio of the aggregate object areas to the source image area (i.e., *Fraction-OfSurfaceArea*). An aggregate measurement treats all objects in an ObjetSet as a single object. Measurements are displayed in the Object Global Measurements grid in window B.

Interactive Shape Measurements: To compute and display measurements for each shape manually drawn in the graphics overlay of the Visualization window using the **Interactive Drawing Tools** described below. Measurements are displayed in the Interactive Shape Measurements grid in window B.

Interactive Shape Global Measurements: To compute and display interactive shape attribute measurements values aggregated for the shapes manually drawn in the graphics, plus one measurement value for the ratio of the aggregate interactive shape areas to the source image area (i.e., *FractionOfSurfaceArea*). An aggregate interactive shape measurement treats all shapes listed in the selected grid as if they were a single shape entity. Measurements are displayed in the Interactive Shape Global Measurements grid in window B.

If you have computed measurements for an ObjectSet (or an interactive shapes set), and then attempt to compute measurements for another ObjectSet (or interactive shapes set), you will be given the option to save the current measurements under a different file name before the prior measurements are overwritten by the new values.

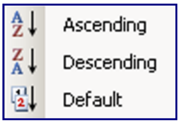
The ObjectSet aggregate measurements made by **Object Global Measurements** are a predefined set of commonly useful attribute measurements such as image name, surface area fraction, region coordinates, and various statistical measurements based on quantity and intensity of pixels. A more detailed description of those global measurements is provided in section *4.1 Global Measurements*. The first time global measurements are computed, the resulting measurement values are entered in the corresponding global measurements grid in a single row. Each subsequent time that global measurements are computed, irrespective of what ObjectSet is used for those measurements, a new row is added to the global measurements grid. Save the current grid using the **File** menu.

Any row of a global measurements grid can be removed from that grid by left-clicking the row's Keys cell, followed by right-clicking to access the rows drop-down menu, then selecting **Delete**. Select **Delete All Objects** to remove all rows from the grid.

Each object in the graphics overlay is logically linked to its entry in a measurements grid. Clicking on an object in the graphics overlay causes both the object and its row in a grid to become highlighted. Conversely, clicking on a cell in the Keys column of a grid will highlight both that row and its associated object in the graphics overlay.

Note: After at least one object has been edited using the drawing or erasing tools, the prior measurements grid computations are no longer valid and those measurements should be recomputed to update their values, as should be the corresponding global measurements, if used. The same procedure should be followed when interactive shapes are edited.

Contextual Toolbar – Data Display Tools



Sort: To sort objects (rows) in a grid based on an attribute's measurement (column). Before using this button, a column in a grid must be selected by clicking on a measurement name at the top of that column. The default use of this button is to perform the sort represented by the button's icon. Clicking on this button's drop-down menu symbol (▾) reveals a list of sorting alternatives. The icon displayed on this button will reflect the last sort alternative chosen. Following below is a brief description for each sorting alternative.

Note: Scalars or scalar values are single numbers. The area of a rectangle is a scalar number. Vectors or vector values are a set of numbers, separated by a punctuation mark, whose default setting is a semicolon. To specify a different separator mark, use:

Tools→Advanced→Appearance→GeneralAppearance→ListSeparator

A pixel location is an example of a vector value since it is written as a pair of numbers x,y.

Descending sort: To sort objects based on their values for the selected attribute (column) from lowest to highest value. Only available for measurement values that are scalar.

Ascending sort: To sort objects based on their values for the selected attribute (column) from highest to lowest value. Only available for measurement values that are scalar.

Default sort: To sort objects based on their index number in the *Keys* column (left-most column) from lowest to highest value.

Hide: To hide a grid's selected attribute column(s). Note that measurements values in a hidden column are still contained in the associated ObjectSet. They are just not displayed in the grid. Recomputing the ObjectSet's measurements will not redisplay a hidden column. To redisplay a hidden column, save the ObjectSet as an .aso file and then reopen that .aso file. All of the hidden columns will be displayed again.

Delete: To delete the grid's selected attribute column(s). Measurements in a deleted column are deleted from the active ObjectSet. If the ObjectSet's measurements are recomputed, the deleted column will be restored to the grid. You can also restore a deleted column to the grid by reopening its .aso file, provided that it was saved prior to the deletion operation and was not saved with the same file name after the deletion.

Histogram: To compute the histogram of a selected attribute (i.e., column) and display it in the Chart window. This button is disabled if more than one column is selected. Values in the selected column must be scalar (i.e., not vectors).

Scatter plot: To compute the scatter plot of two attributes (i.e., columns) that are selected. The scatter plot is displayed in the Chart window. Exactly two columns must be selected to enable this button. Press the keyboard Ctrl key and click the left mouse button on the measurement name at the top of the column to select the second column. Values in the two selected columns must be scalar (i.e., not vectors).

Contextual Toolbar – Data Export Tools

The two data export tools provide exporting of a measurement grid to either an Excel-compatible spreadsheet file or a CSV (comma separated values) file. First, select a grid by clicking on its tab. After clicking on one of the export buttons, the user is given the opportunity to specify a name and save location for the exported file. The entire grid is exported.

The user can also specify that the file be opened after it is saved. If the open option is exercised, the file will be opened by the default application for the file format chosen. If a CSV file is opened by a spreadsheet program, all of the column data for a row will appear in the first column of the spreadsheet as a set of appended text strings with all other columns empty, if the selected separator is not the right one. However, Microsoft Excel and OpenOffice Calc provide an import function (*Data→Import External Data→Import Data…*) that will allow proper parsing of a CSV file so the column headings and values appear in proper row, column format. Note, if Microsoft Excel is not installed on the PC, then the .xls file is only saved on disk.

Export as Excel file (.xls): Exports a grid's measurements to an Excel spreadsheet file format (.xls).

Export to CSV: Exports a grid's measurements to a CSV (comma separated values) file format. Note that semicolons are used to separate the values.

Help: The selection of measurements (columns) and objects (rows) in a measurement grid is done respectively by clicking on the name of the measurement (first row of the grid) or on an object's index Key (first column of the grid). The selection of multiple columns or rows is achieved by simultaneously holding down the keyboard's **Ctrl** key and clicking the left mouse button for each column or row to be selected (or unselected if already selected). To select a group of adjacent columns or rows, first click on one of the group's two outer columns or rows, then hold down the **Shift** key while clicking on the group's other outer column or row.

Contextual Toolbar – Interactive Drawing Tools

The interactive drawing tools enable the user to quickly draw geometric shapes in the graphics overlay. Attribute measurements of these shapes can then be quickly computed and displayed in the Interactive Shape Measurements and Interactive Shape Global Measurements grids.

Note: Measurements are available on a live image. First, set the *AutoFreeze* and *AutoStop* parameters in the Acquisition section of the **Tools→Options…→Advanded** window to the value False. Then, grab an image in the **Image Acquisition** Task. Finally, move to the **Measurements** Task and compute interactive measurements. Once done, we recommend updating back the parameters to the value True.

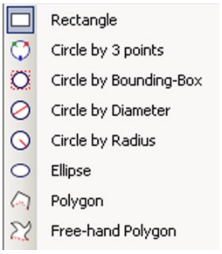
The drawing tools provided in this task are different from the drawing tools available in the **Object Editing** Task, described earlier. While the **Object Editing** drawing tools are used to create new or modify existing objects (i.e., Bitmaps), the interactive drawing tools have no effect on objects. Rather, the interactive drawing tools are used to create geometric shapes (e.g., Lines, Angles, Circles, and Polygons) whose attributes can also be measured (e.g., length, angle, radius, area). While both the interactive shapes and extracted objects are displayed in the graphics overlay, their respective sets of attribute data are displayed in separate grids and saved as separate .aso files.

Note: Interactive shapes can overlay one another and can intersect with objects, whereas objects are always distinct and separate from one another.

Detailed descriptions of each interactive drawing tool follow below.

Selection: To select objects in the graphics overlay.

Closed shape: To draw geometric shapes that have closed borders. The default use of this button is to draw the shape type displayed on the button's icon, which will be the last shape chosen. Clicking the button's drop-down menu symbol (▾) displays a list of closed shape alternatives (see right). Following is a description of each closed shape tool.



Rectangle: To draw shapes of type *rectangle*. The rectangle is defined by two, diagonally opposite corners. Move the mouse pointer to the location of the first corner. Press and hold down the left mouse button down to mark that corner. While holding the left button down, move the mouse pointer to the location for the second corner. A preview of the rectangle is shown as the mouse pointer is moved from the first to the second corner. Releasing the button marks the second corner and displays the final rectangle.

Circle by 3 points: To draw shapes of type *circle* using three points on its circumference. This method is based on the geometric property that a circle is uniquely defined by any three points on its circumference. Each time the left button of the mouse is clicked, a point is defined. A preview of the circle is shown as the mouse pointer is moved after the second point is defined. When the third point is defined by a click, the final circle is displayed.

Circle by Bounding Box: To draw shapes of type *circle* based on the bounding box of the circle. A circle is uniquely defined by the smallest square that can completely contain the circle. Move the mouse pointer to the location of the square's upper left corner. Click and hold down the left mouse button to mark that corner. While holding down the left button, drag the mouse pointer away from the upper left corner. The square and circle are previewed as the mouse is moved. When the circle is the desired size, release the left button to display the final circle.

Circle by Diameter: To draw shapes of type *circle* based on the circle's diameter. A circle is uniquely defined by the end points of any of its diameter lines. Move the mouse pointer to the location of the first end point. Click and hold down the left mouse button to mark that end point. While holding the left button down, move the mouse pointer to the second end point. The diameter and circle are previewed as the mouse pointer is moved. Release the left mouse button to mark the second end point and display the final circle.

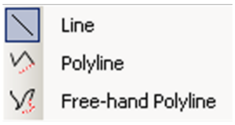
Circle by Radius: To draw shapes of type *circle* based on its radius. A circle is uniquely defined by the end points of any of its radius lines. Move the mouse pointer to the location of the circle's center point. Click and hold the left mouse button down to mark the radius' first end point (i.e., circle's center). While holding down the left button, move the mouse pointer to the second radius end point (at the circle's circumference). The radius and circle are previewed as the mouse pointer is moved. Release the left mouse button to mark the radius' second end point and display the final circle.

Ellipse: To draw shapes of type *ellipse*. An ellipse is uniquely defined by its bounding rectangle and, therefore, by any pair of the bounding rectangle's diagonally opposite corners. Move the mouse pointer to the location of the rectangle's first corner. Press and hold down the left mouse button to mark that corner. While holding the left button down, move the mouse pointer to the location for the second corner. A preview of the ellipse and its bounding rectangle are shown as the mouse pointer is moved away from the first corner. Releasing the left button defines the second corner and displays the final ellipse.

Polygon: To draw shapes of type *polygon*. A polygon is a closed shape comprised of three or more straight line segments (i.e., sides) such that each endpoint of a line segment intersects exactly one other line segment endpoint. Move the mouse pointer to the location of the first endpoint and click the left mouse button to define the first endpoint. Move the mouse pointer to the location of the second endpoint and click the left mouse button to define the first line segment (i.e., side). Repeat this line segment drawing process as needed to complete the polygon. A preview of the polygon begins as the second line segment is drawn. A double click of the left mouse button completes the polygon drawing. (Note that if you allow any side in a polygon drawing to intersect any other side more than twice, you will have drawn multiple, connected polygons, although they will still be treated as a single shape.)

Free-hand Polygon: To draw shapes of type *polygon*. Move the mouse pointer to a location on the perimeter of the polygon. Press and hold down the left mouse button to define the first endpoint of the polygon. While holding down the mouse button, move the mouse pointer to draw the perimeter of the polygon. Releasing the left mouse button will define the last endpoint and cause a straight line to be drawn connecting the last and first endpoints. While the drawn polygon may appear to have curved sides, each curve is comprised of many small, straight line segments, connected end-to-end.

Open shape: To draw open shapes, including straight lines and lines of arbitrary geometry. The default use of this button is to draw the line type displayed on the button's icon, which will be the last type



chosen. Clicking the button's drop-down menu symbol (▾) displays a list of open shape alternatives (see adjacent). Following is a description of each open shape tool.

Line: To draw shapes of type *line*. A line is defined uniquely by its two end points. Move the mouse pointer to the location of the first endpoint. Press and hold down the left mouse button to mark that end point. While holding down the left mouse button, move the mouse pointer to the location for the second endpoint. A preview of the line is shown as the mouse pointer is moved from the first to the second endpoint. Releasing the button marks the second endpoint and displays the final line.

Polyline: To draw shapes of type *polyline*. A polyline is a set of straight line segments connected end-to-end without connecting the first and last endpoints. The first click of the mouse marks the start of the first line segment. Thereafter, as the mouse is moved, each click of the mouse marks the end of the current segment and the start of a new segment. A double click marks the end of the last segment and displays the final polyline.

Free-hand Polyline: To draw shapes of type *free-hand polyline*. Move the mouse pointer to the location of the starting endpoint. Press and hold down the left mouse button to define the first endpoint and continue to hold down the mouse button while moving the pointer to draw the line. When the line is complete, release the mouse button to define the final endpoint of the line. The line is actually a sequence of small, straight line segments connected end to end.

T **Text (Developer Task only):** To add text on an image. Select the **Text** entry in the Shape dropdown list and then click in the image. The Enter Text window opens. Enter the desired text in the empty box at the top of the Enter Text window. Alignment of the text can be changed using the Alignment dropdown menu. The text is then moved into the image depending on the options selected and the location of the mouse cursor. The Rotation Angle counter can be varied ±180º. If the Use default value box is checked, the font of the entered text will be Arial and the size and attributes of the font will be those last selected by a user. To change the font type, size, and attributes, clear the Use default value checkbox and click the Font button to access a font selection dropdown list. Check or clear the Auto Scale box to define if the text follows a zoom action or not. If auto Scale is set to False, the size of the text is not changed when zooming in the image. Click the Apply button to view the effects of the changed settings. After the Enter Text dialog box has been closed, the displayed text can be moved by first selecting the text (i.e., click the Select button on the main toolbar and then click the text) and then drag the text to the desired location. The Edit Text dialog box can be reopened by selecting text and then right clicking to access the contextual menu. From that menu, choose the Edit Text entry.

Angle: To draw shapes of type *angle*. An angle is defined by a polyline composed of two straight line segments whose connecting point is the angle's vertex. Move the mouse pointer to the start of the first line segment. Left click to define the first endpoint. Move the pointer to the vertex point and left click. Then move the pointer to the end of the second line segment and left click to define the final endpoint. The straight line segments are previewed as they are drawn.

Caliper: To compute the distance between two points in the image. Draw a straight line connecting the two points by moving the mouse pointer to the first point and press and hold down the left mouse button to define this endpoint. While holding down the mouse button, move the pointer to the second point and release the mouse button to define the second endpoint. The line is previewed as it is drawn.

Delete selection: To delete the selected objects or shapes. Using the **Selection** tool and select all objects you want to delete. Then click the **Delete selection** button. The selected objects are deleted and the unselected objects remain.

Keep selection: Tool to keep the selected objects or shapes. Using the **Selection** tool and select all objects you want to keep. Then click the **Keep selection** button. The selected objects remain while the unselected objects are deleted.

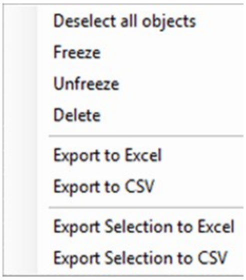
Important Note: When the **Selection** tool is the active tool, the object or shape selected by a left-mouse click is the one located under the mouse pointer. The Active ObjectSet is the ObjectSet or the ShapeSet whose the selected object or shape is belonging. For the multi-selection, note you cannot select both shapes and objects belonging from different ObjectSets and ShapeSet. If you select a shape while objects were previously selected then the shape is selected and all the previously selected objects become unselected.

Contextual Menus of a Measurements Grid

Measurements grids provide various contextual pop-up menus. The entries in these menus are options for displaying or exporting grid data. There are specific menus for row, column, and cell operations and these are activated by right-mouse clicking, respectively a row index in the *Keys* column, a column heading, or an individual grid cell.

Row Contextual Menu

Access this menu by right-mouse clicking on any cell in the Keys column (first column). If no rows are selected when this menu is activated, only Freeze, Unfreeze, Export to Excel, and Export to CSV are enabled. If one or more rows are selected when this menu is activated, all entries in the menu are enabled.



Deselect all objects: Unselects all rows (i.e., objects).

Freeze: Scrolling is enabled only for the rows below the row whose *Keys* index was right clicked. All other rows are frozen (i.e., always in view) until an **Unfreeze** is executed or another row **Freeze** operation is applied to a different row. A row **Freeze** operation will not change an existing column **Freeze** setting.

Unfreeze: Scrolling is re-enabled for all rows.

Delete: Deletes all selected rows.

Export to Excel: Exports the entire grid to Excel.

Export to CSV: Exports the entire grid to a CSV file.

Export Selection to Excel: Exports selected rows to Excel.

Export Selection to CSV: Exports selected rows to a CSV file.

Column Contextual Menu

Access this menu by right-mouse clicking on any column heading (first row). Some menu entries are disabled depending on which column(s) are selected. If a single column is selected that has scalar values, all entries are enabled except for Scatter Plot. If two columns are selected that have scalar values, all entries are enabled except for the first four. If more than two columns are selected or if a selected column includes vector values, the first five entries are not enabled. If no columns are selected, then only Freeze, Unfreeze, Export to Excel, and Export to CSV are enabled.

Following is a brief description of each of the operations listed in the contextual menu for columns (attribute measurements).

Expand/Collapse: If a measurement is a vector with multiple entries, the Expand option displays each individual entry of the vector in a single column, with an index ranging from 0 to the dimension of the vector, and the Collapse option restores the original display in one single column.

Ascending sort: Sorts the rows such that measurement values in a selected column are increasing from top to bottom. This operation is only available if exactly one column with scalar values is selected.

Descending sort: Sorts the rows such that measurement values in a selected column are decreasing from top to bottom. This operation is only available if exactly one column with scalar values is selected.

Default sort: Sorts the rows such that *Keys* index values (leftmost column) are increasing from top to bottom. This operation is only available if exactly one column with scalar values is selected.

Histogram: Computes and displays the histogram of the measurement values in the selected column. This operation is only available if exactly one column with scalar values is selected.

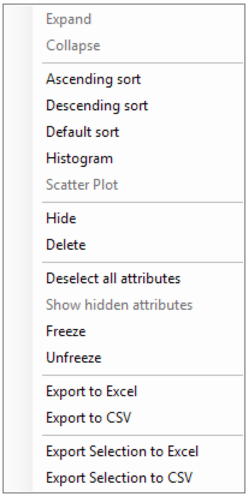
Scatter Plot: Computes and displays the scatter plot of the measurement values in the two selected columns. This operation is only available if exactly two columns with scalar values are selected.

Hide: Hide all selected columns.

Delete: Delete all selected columns.

Deselect all attributes: Unselect all columns (i.e., attributes).

Freeze: Scrolling is enabled only for the columns to the right of the column whose heading cell was right clicked. All other columns are frozen (i.e., always in view) until an **Unfreeze** is executed or another column **Freeze** operation is applied to a different column. A column **Freeze** operation will not change an existing row *Freeze* setting.



Unfreeze: Scrolling is re-enabled for all columns.

Export to Excel: Exports the entire grid to Excel.

Export to CSV: Exports the entire grid to a CSV file.

Export Selection to Excel: Exports selected columns to Excel.

Export Selection to CSV: Exports selected columns to a CSV file.

Cell Contextual Menu

Access this menu by right clicking any data cell in a grid. There are only two operations in this menu and they are always available. The cell that is right clicked becomes the reference cell that determines which rows and columns can be scrolled and which are fixed. Following are descriptions for the two menu entries.

Freeze view: Scrolling is enabled for those rows below the reference cell and those columns to the right of the reference cell. All other rows and columns are frozen (i.e., always in view). If any **Freeze** operation is in effect when the **Freeze view** operation is selected, the existing *Freeze* settings are replaced by the new *Freeze view* settings.

Unfreeze view: Scrolling is re-enabled for all rows and columns.

2.9.6. Report Generation Task

The **Report Generation** Task provides an easy way to create a project report that can contain images, ObjectSets, Measurements grids, and histograms. It also includes tools to copy Aphelion images and ObjectSets to the Windows clipboard for easy insertion into other documents. This task provides a contextual toolbar, the Visualization window, and the Measurement grids. The **Contextual Toolbar** operations are described below.

The report process uses an Excel macro, written in VBA (Visual Basic for Applications), that imports selected project data from Aphelion into an Excel Workbook that contains multiple worksheets of project imagery and data. The macro can be found in the Excel file located at:

FOR	PATH
Dev	*<Aphelion Dev Installation Path>\Aphelion 4.x.y\Dev\bin\<x64> or <win32>\release\report_template.xls*
Lab	*<Aphelion Lab Installation Path>\Aphelion 4.x.y\Dev\bin\<x64> or <win32>\release \report_template.xls*

This macro can be edited by an Administrator to adapt it to the project's specific needs. It is a good idea to back up the file before modifying the macro. Note that knowledge of programming Excel macros in Visual Basic is required.

Report Generator: To generate a project report. Clicking this button causes the Excel macro in *report_template.xls* to execute. This operation is enabled only if there is an Active Image, an Active ObjectSet, and at least one measurements grid computed. Refer to *APPENDIX B* for a detailed description of the Excel macro, including information on edits that can be made to the report format and layout.

Note: Macro execution must be enabled in Excel. To do this, open Excel and click on **Tools→Macro→Security**. The medium security level will allow the Aphelion macro to execute.

Copy whole image to clipboard: To export the whole Active Image to the Windows clipboard.

Copy whole image with ObjectSet to clipboard: To export the whole Active Image and its Active ObjectSet displayed in the graphics overlay to the clipboard.

Copy displayed part of image to clipboard: To export the visible portion of the Active Image to the clipboard. Zoom and pan the image as needed before using this operation.

Copy the displayed part of image with ObjectSet to clipboard: To export the visible portion of the Active Image and the visible portion of its ObjectSet displayed in the graphics overlay to the clipboard. Zoom and pan the image as needed before using this operation.

After exporting the Active Image with or without its Active ObjectSet to the Windows clipboard, the standard Windows tools can be used to paste the clipboard's contents into a document created with Microsoft Word, Excel, PowerPoint, or other compatible applications.

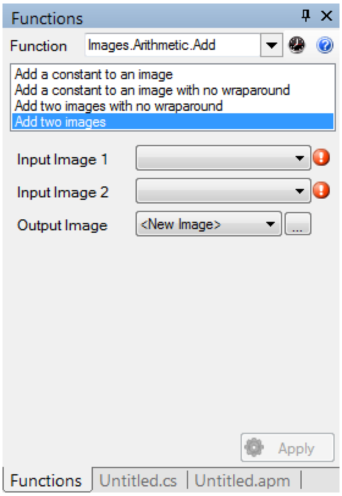
2.9.7. Developer Task (Dev version only)

The **Developer** Task provides a wide range of functions that can be applied to images and ObjectSets. It also provides four scripting languages that enable easy development of image processing applications in the Aphelion environment. Once the **Developer** Task is selected, window D (see *Figure 2.8-1*) is initialized with four, stacked windows. Each of these windows and its contextual toolbar are accessible by clicking on the window's tab.

Note: To add a user function to the pull-down list, open the Aphelion User Guide (Help→Aphelion Help menu), go to the section *C# Programming language* and refer to *Adding a custom function to the Aphelion GUI in C#.*

Functions Panel

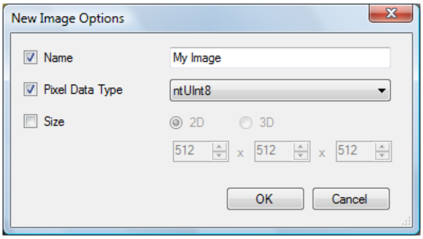
The selection and set up of an image processing or image analysis function is performed in window D after clicking its Functions tab. First, choose the function to be performed by selecting it from the **Process** menu or the Function drop-down list located in window D. When a function is selected using the **Process** menu, then that function will automatically appear as the displayed entry in *Function* name box. Click on the **Recently used function** icon to quickly select a function that has been previously used. To learn more about the selected function, click on the Help icon .



When a selected function has overloads (i.e., it has functional variations based on the type and number of its parameters), any of its overloads is selectable by clicking on one of the entry name located under the *Function* name box. When a function has no overloads, the list contains only one entry. The first overload displayed for a function is the most common one (e.g., for *Image→Arithmetic→Add*, the first overload is the sum of two input images that generates an output image without controlling the sign or the overflow).

The Functions panel has one or more *Image* selection boxes, each with a drop-down list button. The drop-down lists include all currently open images. However, a drop-down list for *Output Image* boxes includes the entry <*New Image*>. If *<New Image>* is chosen, the adjacent button ... becomes active. Click on this button to specify the new image's *Name*, *Pixel Data Type*, and *Size* (see below).

To enter an image name, click on the *Name* checkbox and enter a name for the *Output Image*. The name entered cannot be the name of an open image (i.e., available via a tab in the Visualization windows).


New Image Options

- [✓] Name: My Image
- [✓] Pixel Data Type: ntUInt8
- [ ] Size: [✓ 2D] 3D
  512 x 512 x 512

To have the output image converted to a data type different from the input image(s), pull-down the *Pixel Data Type* menu to choose the desired data type (e.g., 8-bit unsigned, 16-bit signed, float 64). If necessary, choose between the 2D and 3D image alternative and then set the correct pixel counts for the image dimensions. There is usually no need to alter the image size.

Note: The 3rd dimension is only available if the optional 3D Image Processing extension is installed and licensed.

The icon is displayed when a parameter value is invalid or the selected function cannot be applied to it. Hovering the cursor over the icon will display a warning message as a tooltip.

The ... icon identifies a parameter that has sub-parameters for which a graphical tool is provided to specify their values (e.g., defining a convolution kernel). Clicking on the icon displays the parameter's graphical tool (e.g., see *APPENDIX F - Kernel and Structuring Element Editor (Dev version only)*. After the sub-parameter values have been set, click the **OK** button to save them and close the graphical tool. Alternatively, click the **Cancel** button to close the graphical tool without saving the sub-parameter values.

After all of the selected function's parameters are specified, click the Apply button to execute the function.

Macro Programming Windows

Window D's four stacked windows includes, along with the Functions panel, three windows that host programming environments for macro scripting languages, one window each for BasicScript (a Visual Basic compatible language) and Python, and one window shared by Visual Basic and Visual C#. A BasicScript macro file has the extension .apm (for **Ap**helion **m**acro). A Python macro file has the extension .py, a Visual C# macro file has a .cs extension, and a Visual Basic macro file has a .vb extension. These extensions can be seen in the file names on their respective window tabs.

An Aphelion Dev work session can be recorded in a macro file using the BasicScript language. All functions invoked and ran in window D are recorded in the macro, none of the interactive actions such as drawing in images, zooming, etc. are recorded. Since a recorded macro file can be edited, debugged, and played using the contextual toolbar, it provides a convenient way for developing image-based applications. One can construct a macro using the record capability or by typing code directly. While Python, Visual C#, and Visual Basic code can be written and executed in the Aphelion Dev environment, these languages did not include a recording capability at the time this document was printed.

The **Macro Programming Contextual Toolbar** includes the following functions:

▶	Run the macro
⏸	Pause the execution
■	Stop the execution

Step through the code
Step over the code
Go to the end of the code
Enable/Disable break point
Add watch point to enable the display of the variable value during the macro running
Stack icon. Displays the stack of all the entry points
Start Recording
Stop Recording
Cancel Recording
Find a text (string) in the macro
Undo
Redo
Increase indentation
Decrease indentation
Comment selected lines
Uncomment selected lines
Show/hide carriage return

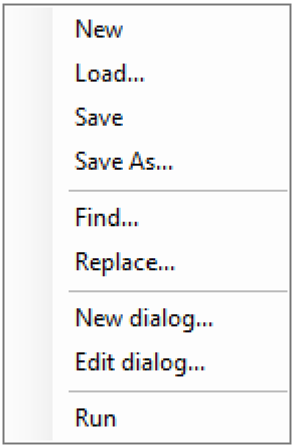
To record a macro, select the BasicScript Macro tab in window D (if the window is empty, the tab name will be *Untitled.apm*). Then click on the **Start Recording** icon ( ) on the **Contextual Toolbar**. To stop the recording, click on the **Stop Recording** icon ( ). To abort a recording, click on the **Abort Recording** icon ( ).

Important note: Recording and debugging a macro is only available in BasicScript.

Tip: A macro will be easier to view if its window is undocked and then expanded to display full lines of code. To undock the Macro window, double-click on the associated tab or window banner (in this case, all the stacked windows are undocked). To redock the window, double-click on its banner. Use the icons in the contextual toolbar to execute the code. For further details about window management, refer to section *2.8.1 Workspace and windows* of this document.

Contextual menus are provided for each Macro Programming interface in window D. Each menu provides operations specific to one of the languages. The menus are activated by right-mouse clicking in the selected Macro window.

BasicScript Contextual Menu



Create a new macro
Open an existing macro
Save a macro
Save macro in a user-specified file

Find a string of characters
Replace a string of characters

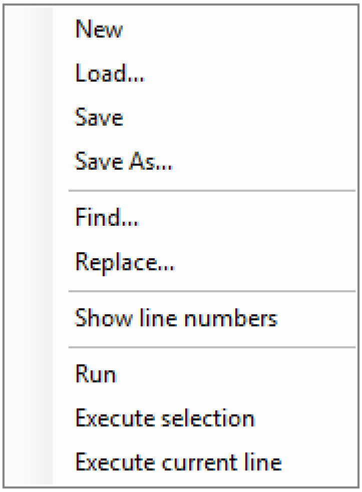
Create a new dialog structure
Edit an existing dialog structure

Run the current macro

The dialog editor available in BasicScript can be invoked from the menu above after clicking on the *New dialog* menu entry. It helps the user create dialogs in a macro. To get help on a specific BasicScript function, press the **F1** key.

Note: All macros developed with Aphelion version 3.2 can be executed in Aphelion Dev version 4.x after minor edits and modifications. See *APPENDIX E - Differences in programming syntax between Aphelion 3.2 and Aphelion 4.x (Dev version only)* for the list of BasicScript functions that were available in version 3.2, but are not available in version 4.x. Macros available in version 3.2 of Aphelion can be found in the Macros32-40 and Examples32-40 folders. Please also refer to *7.1. Aphelion Macros*.

Python (*.py) Contextual Menu



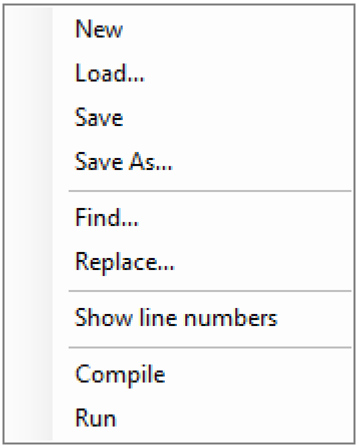
Create a new macro
Open an existing macro
Save a macro
Save a macro in a user-specified file

Find a string of characters
Replace a string of characters

Show line numbers

Execute selected lines of code
Execute the selected lines
Execute the current line

Visual C# (*.cs) and Visual Basic (*.vb) Contextual Menu



Create a new macro
Open an existing macro
Save a macro
Save a macro in a user-specified file

Find a string of characters
Replace a string of characters

Display the line numbers

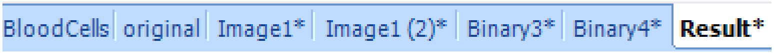
Compile the current macro
Run the current macro

During macro execution, images are not automatically displayed in the Visualization window, except in BasicScript; a specific function has to be called in the macro code to perform that task. The display function alternatives are:

LANGUAGE	IMAGE DISPLAY FUNCTION
BasicScript:	AphImgView
Python:	AphImages.Add
Visual C#:	app.Images.Add
Visual Basic	app.Images.Add

Note: To run a macro in an undocked window, the window D banner must be highlighted to activate the Macro Programming Contextual Toolbar.

When executing a macro that generates successive images, the images will be displayed as stacked images in window A, each with a tab containing its image's name. In the figure below, the set of tabs named BloodCells, original, Image1*, Imag1 (2)*, Binary3*, Binary4*, and Result* were created by running a macro.



The presence of an asterisk appended to an image name means the image has not been saved since it was last modified or is a new image.

Immediate and Output Windows

When the **Developer** Task is selected, window B contains several stacked windows that display thumbnail and measurement grids for images and ObjectSets, as well as various data output windows. The **Developer** tabs for window B typically include the names Image Gallery, ObjectSet Gallery, Output, BasicScript Command, Python Command, and various measurement grid names. Images and ObjectSets generated by functions and macros appear in their respective Galleries.

The Output window displays measurements and other information generated by functions and macros. For example, if a Python macro executes a **print** function, the output of that **print** function is displayed in the Output window. Similarly, for a BasicScript macro, the output of the **AphInfoWrite** function is displayed in the Output window.

The two remaining windows, BasicScript Command and Python Command, can be used to execute individual commands. When a macro command is typed or copied into one of these windows, it is immediately executed after the **Enter** key is pressed. For example, typing a print function in the Python Command window, followed by pressing the **Enter** key, causes the print results to be immediately displayed in the Output window. Similarly, typing an **AphInfoWrite** function in the BasicScript Command window, followed by pressing the **Enter** key, causes the function's results to immediately display in the Output window.

The BasicScript Command window has two features that are not available in the Python Command window:

- The BasicScript Command window enables the user to execute individual BasicScript command lines. A command can be typed into the command line dialog box (located at the bottom of the BasicScript Command window) or a command line can be copied or cut from the BasicScript Macro window and pasted into the command line dialog box. The upper section of the BasicScript Command window is the command line history list which displays an ordered list of the previously executed commands. When the **Enter** key is pressed, the command in the dialog box is (1) immediately executed, (2) inserted as the last line of the history list, and (3) cleared from the dialog box. When the dialog box has been cleared, the user can easily copy a command line from the history list by using the up and down arrow keys

- The dialog box has an auto-complete feature that speeds up typing of a function's name. When the first character of the function name is typed, a list appears containing all functions names beginning with the typed character. As one types additional characters, the list is accordingly reduced to qualifying names. If the user highlights a name on the list, that name is automatically copied into the dialog box. At this point, the user enters the appropriate function parameters into the dialog box and depresses the **Enter** key.

3. TUTORIAL

This chapter should be read by new Aphelion users and by users of Aphelion version 3.2 who wish to get started quickly with Aphelion version 4. Reading through the tutorial material will quickly acquaint you with the Aphelion product's all new graphical user interface and many of its powerful new features. Performing the steps listed in the tutorial sections will provide a good general overview of Aphelion 4.x methods and capabilities, helping assure you will quickly become productive with Aphelion 4.x. A file containing Frequently Asked Questions on Aphelion is also available in the Help folder of the DVD.

3.1. Multimedia demonstrations

If you are new to Aphelion, you will find it helpful to play the multimedia demonstrations available on the Aphelion DVD in the Multimedia Demos folder. The demos show typical usage of the Aphelion software, from basic settings to report generation. The demos can be found in the *Multimedia Demos* folder as .avi files compatible with Windows Media Player software.

3.2. Walk-through of an image processing and image analysis application

If you are a new user and want to try out some of the basic tasks in Aphelion, such as opening an image and running a function, you can follow the tutorial below.

Note: We recommend that you read this section before using Aphelion on a regular basis, and refer to the online Aphelion Imaging Software Suite User Guide as needed.

1. **Select a task**
   Each task has a specific work environment and purpose. The **Task Bar**, located by default along the left edge of the Aphelion graphical user interface, is used to select the task to be performed. The user controls an Aphelion project by successively selecting the tasks, as needed, from **Image Acquisition** to **Developer**. For simple projects, the linear sequence of the tasks **Image Acquisition** through **Report Generation** may be sufficient. For more complex projects, the **Developer** Task (only available in the Dev version) provides an extensive set of additional image and ObjectSet functions, as well as macro programming environments.

2. **Open or acquire an image to process**
   There are two ways to select an image: 1) open an image stored on media or 2) capture an image with an attached imager (e.g., camera). For this walk-through, we'll choose an example image from the hard drive. On the **Main toolbar**, click the button. Navigate to the folder:

   <Aphelion Installation Path>\Aphelion 4.x.y\Dev\Images

Note: If Aphelion is not installed in this default path, navigate to the Aphelion installation folder called *Images*.

   From the Open dialog box, double-click *BloodCells_1.tif*. The image opens in the Visualization window.

3. **Adjust the display parameters for the image**
   The following buttons in the **Main Toolbar** are used to change the zooming factor and the image area to be displayed in the Visualization window. Experiment with these operations to learn how they work. Refer back to the zoom buttons discussion in section *2.8.4 Main Toolbar*.

4. **Extract blobs in an image**

Select the **Object Extraction** Task by clicking the button on the **Task Bar**. In window D, the Object Extraction window, click the **Back** button until you are at the first panel (**Back** button becomes grayed-out). In the first panel, select the *Green* channel, then click **Next**. In the second panel, unselect *Apply Filter* and then select *Threshold*, *Automatic threshold computation*, and *Select dark objects*.

As you made changes to these parameters, their effects could be seen immediately in the Preview window. Remember that you can also change what image area is previewed by modifying the cyan rectangle in the Visualization window. Click **Next**.

At the third panel, first click Apply to see the object extraction results in the Visualization window. Visually inspect the image in the Visualization window. Be sure you see the entire image by clicking on the icon or zooming out so the image frame is visible. Determine if any objects have holes and if any objects are touching (clustered).

- If no objects are touching and none have holes, clear the first three checkboxes.
- If no objects are touching and some have holes, select only the *Fill holes* option.
- If some objects are touching, but none have holes, then select the *Split touching objects (before hole filling)* option.
- If some objects are touching and some have holes, then select the *Fill holes* and the *Split touching objects (after hole filling)* options.

For the *BloodCells_1.tif* image, select the options for *Fill holes* and *Split touching objects (after hole filling)*.

When an object is touching the image frame, an object processing function will assume where the frame touches the object is actually part of the object's boundary. Such partial objects will bias computation of individual measurements. In such cases, the *Remove objects touching image frame* option should be selected. Be aware that there can be cases where all objects are touching the image frame. In such cases, the function would eliminate all of the objects. Thus, the use of this function requires thoughtful consideration of the image contents and the analysis goals. For our example, this option is appropriate and should be selected.

The last checkbox is *Filter objects on*. For this tutorial, we will not use this function.

Finally click Apply to apply all of the selected extraction processes to the entire image. All of the extracted objects are now outlined in the Visualization window. Sometimes it will be necessary to experiment with the object extraction parameters to achieve an acceptable extraction.

5. **Edit the result of the Object Extraction Task**
   After completion of the **Object Extraction** Task, it may be useful to edit its results. To begin editing, select the **Object Editing** Task by clicking on the button. Note that the object outlines are now filled in, forming a red mask over the underlying image pixels. Those pixels that are not part of an object are still visible.
   To become familiar with the drawing tools (found on the task contextual toolbar on the right edge of the Workspace), do the following:

   - Find an object with a hole and fill in the hole.
   - Join two neighboring objects to make a single object.
   - Split a large object into two separate objects.
   - Smooth out the edges of an object.

   Try some of your own ideas using , , , , .

   You need to select a drawing tool, **Draw** ( ) or **Erase** ( ), before selecting the shape you will draw.

Remember, the **Undo** button only cancels the last editing action, and the **Redo** button cancels only the last undo action (see the topic *Object Editing Contextual Toolbar* in section *2.9.4 Object Editing Task*).

6. **Compute measurements**

   Select the **Measurements** Task by clicking on the task button.

   Select the *Object Measurements* operation by clicking the button on the **Contextual toolbar**: Individual measurements for each object will be computed and displayed in the Object Measurements grid. Next, select the *Object Global Measurements* operation by clicking the button on the **Contextual toolbar**. The Object Global Measurements grid will become active and the global measurements will be computed and displayed in this grid. Examine these two grids to become familiar with their contents.

Tip: If the image includes objects touching its frame and both global and individual measurements are computed, then perform as follows: 1) Select the **Object Extraction** Task; 2) Uncheck the Remove objects touching image frame option and click the Apply button; 3) Select the **Measurements** Task; 4) Compute the Object Global Measurements; 5) Select the **Object Extraction** Task; 6) Select the *Remove objects touching image frame* option and click the Apply button; 7) Select the **Measurements** Task; 8) Compute the Object Measurements.

Next, you will use the four shape tools (closed shape, open shape, caliper, and angle), found on the **Contextual toolbar**, to draw shapes in the graphics overlay that provide measurements that can help achieve your analysis goals. Those interactive shapes will be displayed in green in the graphics overlay.

Draw at least one of each shape type, and then select the *Interactive Shape Measurements* operation by clicking the button on the **Contextual Toolbar**. Individual measurements for each shape will be computed and displayed in the Interactive Shape Measurements grid. Next, select the *Interactive Shape Global Measurements* operation by clicking the button on the **Contextual Toolbar**. The Interactive Shape Global Measurements grid will become active and these global measurements will be computed and displayed in this grid. Examine these two grids to understand how using the interactive shapes tools can provide useful, supplemental information.

Note: While interactive shapes are drawn in the graphics overlay, they are special purpose objects that are not fully compatible with objects created with the **Object Extraction, Object Editing, and Developer** Tasks. These shape objects cannot be combined in any way with the other object types. The only ObjectSet functions that can be applied to shape objects are the *ObjectSet→Utility* functions.

7. **Generate the analysis report**
   First, be sure the computer you are using has Microsoft Excel installed. Next, be sure you have both an image and either an ObjectSet or set of interactive shapes open and at least one grid computed. Select the **Report Generation** Task by clicking the button on the **Task Bar**. Next, execute the *Generate Report* operation by clicking the button on the **Contextual toolbar**. If the macro security warning message appears, select the *Enable Macros* button in that message. The report is then quickly generated as an Excel compatible file. During the generation process, you will have the opportunity to specify up to nine object attributes to export and what attribute should be used in a histogram chart.

   The **Contextual toolbar** provides four additional buttons that enable you to copy image, ObjectSet, and interactive shape data to the Windows clipboard. View the pop-up tooltip for these buttons by hovering the mouse pointer over each of the buttons. The copied data can be pasted into the report just generated or into a separate document (e.g., Word). Experiment with these tools to customize your report.

Remember, you can customize the Excel macro and the Excel template used by the **Report Generator**.

8. **Save images and ObjectSets**
   Before closing a window containing images, ObjectSets, or interactive shapes that have not been saved since being created or modified, Aphelion Lab and Dev ask if the files should be saved. Select the save options you prefer. Remember, you may save an image, ObjectSet, and interactive shapes at any time using the *Save* operations provided by the **Files** menu.

3.3. *Interactive Analysis Tools* *(Dev version only)*

3.3.1. Intensity Profile

1. **Draw a line and display its intensity profile**
   Select the **Developer** Task. Enable the **Profile** tool by clicking on the **Profile** button ( ) on the **Main toolbar**. Next, click the drop-down menu button on the **Shape** tool ( ) to select a shape drawing tool. Since an intensity profile is always taken along a straight line, select *VerticalLine* ( ). Create a vertical line profile by clicking on a point in the image through which you want the vertical profile to be taken. A vertical line is drawn and the intensity profile along that line is then displayed in the Chart window (window E).

2. **Select and move the line**
   Make sure the **Select** tool ( ) is active (button outlined in blue). If it is not, click on it now. Select the line drawn in the previous step and move it. Note that the profile chart is updated in real-time to show the intensity profile along the moving line.

3. **Delete the selected line**
   Using the **Select** tool ( ), click on the line to select it. Next, click **Delete Selection** button ( ). The line is deleted and the profile chart is cleared.

   Note that multiple lines can be drawn. However, the intensity profile will reflect the last line drawn or selected. More than one line can be selected by holding down the **Shift** or **Control (Ctrl)** key and clicking on the lines you want selected. All selected lines can be moved or deleted simultaneously.

3.3.2. Histogram

1. **Draw a shape and display its histogram**
   While in the **Developer** Task, enable the **Histogram** tool by clicking on the button on the **Main toolbar**. Next, click the drop-down menu button on the **Shape** tool to select a shape drawing tool. While any closed shape can be used, choose the **Rectangle** ( ) tool and draw a rectangle over the desired image area. Release the left mouse button to complete drawing of the rectangle. The intensity histogram is displayed immediately in the Chart window.

2. **Select, move, and resize the rectangle**
   Make sure the **Select** tool ( ) is active (button outlined in blue). If it is not, click on it now. Any shape in the Visualization window can now be selected and moved. Select the rectangle drawn in the previous step to cause its handles to appear. With the mouse pointer positioned on any one of the handles, hold down the left mouse button and move the mouse to resize the rectangle. Release the mouse button to complete the resizing. To move the rectangle, select it and move the mouse pointer to the rectangle's interior. Hold down the left mouse button while you drag the rectangle, releasing the mouse button when the rectangle is at the desired location. Note that the histogram is updated in real-time as the rectangle is resized or moved. Note also that the user can turn on or off any colors in the histogram by clicking that color's

label under the histogram or turn on all three colors by clicking anywhere in the Chart window except on the color labels and in the chart's graphing area.

3. **Delete the selected rectangle**
   Using the **Select** tool ( ), click on the rectangle to select it.

   Next, click **Delete Selection** button ( ). The line is deleted and the histogram chart is cleared. Multiple shapes can be selected by holding down the **Shift** or **Control (Ctrl)** keys while clicking on the shapes. Once selected, multiple shapes can be moved as a group or can be deleted by a single click the button.

Note: The conversion of one or more objects of an ObjectSet into multiple ROIs can be done by calling C# code. Take a look at the macro *RoiProcess.apm* in the *Macros32-40* macro folder to find the syntax. This folder can be found at:

<Dev Installation Path>\Aphelion 4.x.y\Dev/Macros32-40 \RoiProcess.apm

3.4. Using Developer functions & macros (Dev version only)

3.4.1. Select and run a function

1. **Accessing the Developer Task**

   Click on the **Developer** Task ( ) on the **Task Bar**. Note that the **Developer** Task can also be accessed by selecting a function from the **Process** menu on the **Menu Bar**.

2. **Select the function to run**
   A function can be selected using either the **Process** menu on the **Menu Bar** or the *Function* drop-down list located near the top of the Functions panel in window D. For example, using the **Process** menu select *Images→Utility→ExtractBand*. Using the *Function* drop-down list, select *Image.Utility.ExtractBand* entry. Try both methods; sometimes one will be more convenient to use than the other.

3. **View the online Help for the selected function**
   The *ExtractBand* is selected if its name appears in the Function box at the top of window D. Click the **F1** key on the keyboard while the Functions window is active to display the Help topic for the selected function *ExtractBand*.

4. **Specify the function parameters, including the input and output images**
   If you haven't already done so, open the *BloodCells_1.tif* image found at *<Dev Installation Path>\Aphelion 4.x.y\Dev\Images\*. Then, click the **Input image** box in the Functions panel and select *BloodCells_1* from the drop-down list. Then click the **Output image** box and select *<New image>*. In this case, Aphelion Dev will automatically define the data type and the size of the output image depending on the function executed. Finally, change the band value from 0 to 1 (for an RGB color image: 0 = Red, 1 = Green, 2 = Blue).

5. **Run the function**
   Click Apply to run the function. Observe that the extracted green band appears as a new image in the Image Gallery in window B with the name Image n (where n is an index value for multiple uses of the name Image.

Note: The Apply button will be disabled (Apply) if any parameter is not correctly specified. The ❗ icon is displayed at the right of a parameter that is not properly specified. Hover the mouse pointer over the icon and a tooltip will pop-up explaining the problem.

3.5. Load and run a BasicScript macro

1. **Load a BasicScript macro**
   Aphelion Dev supports programming of macros in four different languages: BasicScript (unique macro language used in Aphelion 3.2), Python, Visual Basic, and Visual C#. Click on the window D tab Untitled.apm. Load the BasicScript macro *Ceramic.apm* by right-clicking in the BasicScript Macro window and selecting **Load…** from the contextual menu. Windows File Explorer will open. Navigate to:

   <Dev Installation Path>\Aphelion 4.x.y\Dev\ Macros32-40\BasicScript\

   and select the *Ceramic.apm* file. The file opens in the Macro window and the file name appears in the window's banner.

Note: The Macro Programming panel can be undocked as a window by double-clicking its tab. This window becomes a floating window. It can be moved to any location on the Windows desktop and resized so the macro code can be more easily read and edited. To redock the panel, double-click on its banner.

---

2. **Run the macro**
   Click ▶ to run the *Ceramic.apm* macro that is open in the Macro window. You will see the macro processing as the intermediate images appear in the Visualization window. The final image shows the fiber ends outlined in red. You can also see that the macro excluded fibers touching the window's frame. Each of the intermediate images can be viewed in the Visualization window by clicking on its tab at the bottom of the window. The intermediate images are also viewable in the Image Gallery as thumbnails. Any image can become the Active Image in the Visualization window by just double-clicking on its thumbnail.

   Run the macro again, but use the **Step Into** button ( ) to see the macro executed line-by-line. In this case, the macro can be stopped before reaching its end by clicking the **Stop** button (■).

Note: If the Macro Programming window is undocked or not active, double-click the **Run** button (▶) instead of a single click to run the macro. (The first click selects window D.)

3. **Record a new macro**
   Only BasicScript macros can be recorded. The process below records a simple macro. Begin by selecting the BasicScript panel in window D by clicking on its tab (labeled <macro_name>*.apm*). If macro code already exists in this panel, it must be removed first. To do this, right click in the panel's programming area (white) and select *New* from the drop-down menu. If the existing code has not been saved since it was last modified, a window opens asking if the macro should be saved. Once you have answered the save macro question, the macro disappears from the programming area. Next, click on the **Start Recording** button ( ) on the Contextual toolbar. Now perform the following steps:

   a. Click on **Close All** from the **Windows** menu to close all the loaded images. This will assure that the first *<New Image>* created will be called Image 1.
   b. Load (open) the *BloodCells_1.tif* image.
   c. Click on window D's Functions tab.
   d. Select the function: *Images→Utility→ExtractBand*.
   e. Set *Input Image* to: *BloodCells_1*.
   f. Set *Output Image* to: *<New Image>* to avoid overwriting of the original *BloodCells_1* image.
   g. Set *Band* to: 0 to extract the red channel that corresponds to band 0 of the original RGB image.
   h. Click Apply (Note name of resulting image is *Image 1*.)
   i. Set *Output* Image to: *<New Image>*.
   j. Set *Band* to: 1 to extract the green channel.
   k. Click Apply (Note name of resulting image is, *Image 2*.)

l. Select the function: *Images→Arithmetic→Subtract*.

m. Set *Input Image 1* to: *Image 1* (result of step g).

n. Set *Input Image 2* to: *Image 2* (result of step j).

o. Set *Output Image* to: <*New Image*>.

p. Click Apply. (Note name of resulting image is, *Image 3*.)

q. Click on the BasicScript tab to switch to the macro code.

r. Click on the **Stop Recording** button ( ).

Now, run the macro you just created by clicking on the **Run** button ( ).

Your macro extracted the red and green bands from the *BloodCells_1* image and labeled them *Image 1* and *Image 2*, respectively. Then it subtracted the second image (i.e., the green band) from the first image (i.e., the red band).

Note: Steps a. and b. in the above sequence, which included opening the image *BloodCells_1*, were not recorded in the macro. This is because these two steps used GUI operations rather than **Developer** functions that are callable from BasicScript. To record the opening of an image in a macro, follow the instructions below: 1) Select function: *Images→IO→Read*; 2) Set image to <New Image>; 3) Set Filename to *<Dev Installation Path>\Aphelion 4.x.y\Dev \Images\BloodCells_1.tif*. 4) Click Apply to load the image.

Note that the *BloodCells_1* image will be called *Image 1* in Aphelion and the red and green channels will be respectively called *Image 2* and i*mage 3*.

3.6. Basic knowledge in image processing & analysis

3.6.1. Processing Images

This section describes the basics of how to process and enhance images.

Filtering to remove gray-level noise

Gray-level noise is characterized by varying pixel values. This noise can be random or regular. Any filtering function will affect pixel values and will usually remove any type of noise in an image.

Linear or Convolution Filters

Linear filters, such as a convolution are the most basic of image processing tools. Depending on the kernel passed to *Images→ Filtering→Convolve*, you can apply a low-pass filter or a high-pass filter to an image. For removing random gray-level noise, use one of the low-pass filters available in Aphelion **Developer** Task, such as *Images→Filtering→Box*, *Images→Filtering→Gaussian*, or select the *Convolve* function and choose its overload that provides a predefined kernel, then select the *Low Pass 5x5* predefined kernel from its Kernel drop-down menu.

Tip: To select a function overload, scroll through the list of available overloads that are displayed in the section below the function name.

To enhance object boundaries in an image, use a high-pass filter.

Non-linear Filters

Non-linear filters are filters whose output is not proportional to the input. For example, smoothing by averaging is linear, while median smoothing is not. An example of a non-linear filter in Aphelion is *Images→Filtering→Median*. This function replaces each pixel's value with the median value (i.e., middle value of rank-ordered values) of its neighboring pixels. As with the linear low-pass filters, a larger neighborhood results in a more uniform output image (large areas with quite similar pixel intensity values).

Frequency Filters

Frequency filters are useful when you have predictable gray-scale noise (e.g., non-random noise generated by the sensor, such as noise due to the alternating current). You can run *Images→Frequency→FFT* to view an image's power spectrum (also an image). If you see well-defined clusters that are not near the center of the power spectrum image, these may be caused by repetitive noise or by a repetitive pattern such as a grid. Such artifacts often can be removed with a frequency filtering function such as *Images→ Frequency→Filter*.

After filtering the power spectrum image, removing high frequency pixels, use the *Images→Frequency→InverseFFT* function to recreate the original gray-scale image without the undesired pixels.

To view any of the four bands of a frequency image use the function *Images→Math→RealPart | ImaginaryPart | Magnitude | Phase*.

Filtering to remove spatial noise

Some noise is spatial rather than gray-level. This noise is characterized by roughness along the boundaries of objects, or holes within objects. Aphelion's morphological functions can be very effective at cleaning up this type of noise. For instance, the function *Images→Morphology→OpeningClosing→Open* can be applied to a binary, gray-scale, or color image in order to remove boundary roughness from bright objects. The function *Images→Morphology →OpeningClosing→Close* can be used to fill in holes in objects.

Other Image Processing functions

The above functions provide an introduction to just a few image processing groupings in Dev's function library. Following are additional examples of image processing function groups in Dev:

- *Arithmetic/Logic (Add, Multiply, And);*
- *EdgeDetection (SobelEdges);*

- *Geometry (Rotate, Scale);*
- *Utility (Copy, ExtractBand).*

3.6.2. Extracting and Analyzing Objects from Images

This section describes how to use Aphelion software to measure objects in images.

Global Measurements

Global measurements are either computed on the entire image or limited by a mask. The mask used is the aggregation of the object surfaces generated by the **Object Extraction** and **Object Editing** Tasks, or by specific functions in the **Developer** Task. For instance, the measurement called FractionOfSurface computed in the **Measurements** Task is a global measurement. It is the surface area of the mask divided by the surface area of the entire image. Given a binary image (i.e., each pixel is either black, having a value of zero or is red, having a value of 1, meaning each pixel is in, respectively, the image background or its foreground), the area of the image is defined as the number of foreground pixels in the image multiplied by the pixel area in real-world units. Other global measurements are: intensity mean of all the image pixels (computed by *Images→Measurements→Moments*), intercepts, object count, etc.

A binary image can be generated with a simple threshold operation (e.g., *Images→ Segmentation→Threshold*).

Note: A simple *Threshold* operation generates a binary image. The **Object Extraction** Task described above automatically generates an ObjectSet, bypassing the step of generating a binary image. But it is always possible to perform the conversion step by step, as described in the example below.

Then, one can compute the total area of the objects as the number of pixels in the foreground of the binary image. After the Threshold function just referenced, apply the function *Images→Measurements →Area* and select the binary image as the input image.

The volume of a gray-scale image is defined as the sum of the pixel intensity values for all pixels in the image. The function that performs this measurement is *Images→Measurements→Volume*.

Segmentation

One is often interested in individual objects in an image, rather than characteristics of the overall image. Aphelion allows you to identify individual objects using either the processes found in the **Object Extraction** and **Object Editing** Tasks, or the segmentation functions (i.e., *Images→Segmentation* group) found in the **Developer** Task.

Bitmap is the most common object *spatial attribute* value. A Bitmap is a set of connected pixels (i.e., each pixel is a neighbor of at least one pixel in the same Bitmap). With the **Developer** Task, the most basic way to generate Bitmaps is to threshold the image using:
ObjectSets→Bitmaps→Generation→Threshold
This threshold function is manual in that one must specify the threshold interval (i.e., upper and lower limits). The resulting Bitmaps (i.e., ObjectSet) can then be displayed in the graphics overlay by double-clicking the left mouse button while the mouse pointer hovers over the ObjectSet thumbnail in the ObjectSet Gallery, or right clicking the thumbnail, followed by clicking *Show→Bitmap*.
There are also a number of automatic segmentation operators, such as *EntropyThreshold* in the same function group.

If the image is already segmented as a binary image, then it can be converted into a set of Bitmaps using the *ObjectSets→Bitmaps→ Generation→Clusters* function.

An object type is defined by its spatial attribute. In addition to the most common spatial attribute, Bitmap, the **Developer** Task provides other spatial attributes. The following list gives example functions for extracting some of the other object types:

- Chains (*ObjectSets→Chains→Generation→EdgelsToChains*);
- Edgels (*ObjectSets→Edgels→Generation→EdgesToEdgel*);
- Lines (*ObjectSets→Lines→Generation→RegionLineFit*);

- Polygons (*ObjectSets→Polygons→Generation→ConvexHull*).

Object Measurements

Object measurements are functions that calculate the value of object attributes (e.g., length, area, and angle). The attributes possible depend on the object type (e.g., Line, Bitmap, and Chain). Typically, measurements are computed in order to quantify, filter, or classify objects found in an image. Aphelion contains powerful functions for computing and visualizing object measurements.

Note: The terms *Bitmap* and *Region* both name the same entity and may be used interchangeably in this text and in the Aphelion software. *Region* was used exclusively in Aphelion versions prior to Aphelion 4.x.

ObjectSets

An ObjectSet is a group of objects derived from a specific source image and saved together in a special file format (.aso or .tks). This file includes the spatial attribute(s) (i.e., the spatial representation) and the numeric attributes (measurements available as scalars or vectors) for each object. These data are structured as a database. Numeric attributes are displayed in a grid in Aphelion. Each row corresponds to a specific object in the ObjectSet and each column contains the measurement values for a specific attribute of the objects in the ObjectSet. Deleting a row in the grid results automatically in the deletion of all data (including the spatial attribute) associated with the corresponding object. Using the **Object Editing** Task to add an object to an ObjectSet results in also adding a new row in the grid for the corresponding ObjectSet.

There are a number of ObjectSet utility functions, including these examples:

- *ObjectSets→Utility→Append|Copy|Merge|ToImage;*
- *ObjectSets→Filtering→Filter.*

You can load and save ObjectSets using the entries in the **File** menu.

Tip: While version 4.x of Aphelion has its own ObjectSet file format called AphelionSerialObjects (.aso), its ObjectSet access functions are backwards compatible with the Aphelion 3.2 file format (.tks). The .aso format supports new attribute types and very large images since image coordinates are now defined as 32 bit variables. For these and other reasons, the .aso format should be used instead of the old .tks format.

Example: Producing an Aphelion ObjectSet using the Developer Task

This example describes how to use the **Developer** Task to produce object measurements in a grid when starting from an image.

1. Open the image for which you want to compute an Object Measurements grid.
2. Using the **Process** menu, select the function:

   Images→Utility→ExtractBand

   - From the *Input Image* pull-down list, select the image you loaded in step 1 above.
   - From the *Output Image* pull-down list, select *<New Image>*.
   - In the *Band* box, choose the color band from which you wish to extract objects (For BGR images: 0 = Blue, 1 = Green, 2 = Red).
   - Click Apply to execute the function.
3. Use the **Process** menu to select the following function:

   ObjectSet→Bitmaps→Generation→Threshold

   - From the *Input Image* pull-down list, select the color band image you created in step 2 above (e.g., Image 1).
   - From the *Output ObjectSet* drop-down list, select *<New ObjectSet>*.
   - Slide the Low and High threshold sliders until the desired objects are outlined in red in the graphics overlay. The preview of the threshold is only available if the image is gray-scale.

- Select *8 connected* from the Graph drop-down list (see *Connectivity* in the section *2.7 Terminology* to learn more about 4- and 8-connected).
- Click Apply to execute the function.

4. A new ObjectSet is created and appears in the ObjectSet Gallery. Right-click on the ObjectSet thumbnail in the ObjectSet Gallery to open a drop-down menu. Choose either *Show/Hide→Bitmap* to display the ObjectSet in the graphics overlay of the Active Image or *Show Grid* to display a grid in window B that lists values for each object's area and its bounding rectangle coordinates (see section *2.8.10 ObjectSet Gallery Window*.

Grids

Results from an object analysis (i.e., forming an ObjectSet and computing attribute measurements for that ObjectSet) are displayed in an Aphelion grid. A grid is displayed in window B after the **Measurements** Task computes the attribute values for an ObjectSet's objects. If measurements have not yet been computed for an ObjectSet, entering the **Developer** Task and clicking the *Show Grid* function (accessible by right clicking the ObjectSet thumbnail in the ObjectSet Gallery) displays the ObjectSet's default attributes for each object, namely, Area (i.e., pixel count) and the coordinates for its minimum bounding rectangle.

Tip: An ObjectSet can be converted into an image using the *ObjectSets→Utility→ToImage* function. This function assigns an object's scalar value of a user selected numeric attribute to all pixels of that object. As pixel values outside objects are not modified by the function, intensity values of pixels not contained in an object can be set to zero if the output image is set to zero before running the function.

Other Object Processing Functions

The above sections are just an introduction to the object function libraries in Aphelion. To view the highest grouping level of ObjectSet functions, go to the **Menu Bar** and click on **Process→ObjectSets**. Click through each entry in the menu to become more familiar with the functions available to extract, display, and compute measurements for objects. Also see section *5.2 ObjectSets*.

4. LIST OF MEASUREMENTS

This chapter defines global and object measurements.

4.1. *Global Measurements*

Global measurements are measurements computed on an entire image. As such, these measurements are applied to all pixels in an image or to the aggregate of all pixels in an ObjectSet. The results are displayed in the Global measurements grid. This grid has only one row and it corresponds to an entire image or ObjectSet, and one column for each measurement calculated for that image or ObjectSet.

Note: Pixel intensity values are computed and displayed based on the number of intensity channels in the image: one value for gray-scale images and three values for color images (e.g., red, green, blue channels). For color images, channel results are specific to the channel. For example, Pixel Intensity Minimum will display as [I_{Red}, I_{Blue}, I_{Green}], where I_{Red} is the lowest intensity value found in the red channel and is independent of the blue and green channels, and similarly for I_{Blue} and I_{Green}.

4.1.1. List of Global Measurements

Statistical Measurements (based on pixel intensities)

Kurtosis: An image’s centered moment of order 4 of pixel intensities.

Maximum: An image’s highest pixel intensity value.

Mean: An image’s mean pixel intensity value.

Minimum: An image’s lowest pixel intensity value.

Skewness: An image’s centered moment of order 3 of pixel intensities.

Standard Deviation: An image’s standard deviation of pixel intensities.

Area Measurement

FractionOfSurfaceArea: Aggregate area of the objects in an ObjectSet divided by the area of the image from which the ObjectSet was extracted. This measurement is sometimes referred to as Specific Area.

4.2. *Object Measurements*

Object measurements are measurements computed for each object in an ObjectSet, in contrast to global measurements which are computed for the whole image (i.e., all pixels). Object measurements that can be computed depend on the object type and on the list of measurements enabled by the Administrator (see section *2.9.1 Administrator Settings Tasks*).

4.2.1. Object types

Aphelion can generate the following object types:

Surface Object Types:

Bitmap: An object having irregular shape (e.g., a blob) generated by the **Object Extraction** or the **Developer** task.

Ellipse and **Circle**: Object with an elliptic shape.

Polygon: Object with a polygonal contour (i.e., comprised of closed, connected line segments). A polygon can have a regular shape (e.g., square, parallelogram) or be irregular.

Rectangle: Object with a rectangular shape that also qualifies as a polygon.

Line Object Types:

Chain: A set of two or more line segments (i.e., straight lines) connected end-to-end.

Line: A single line segment connecting two pixels as the extremities of the line segment.

Point: Line of length equal to one pixel.

4.2.2. List of Object Measurements

When computed, the following measurements are displayed either in the Interactive Measurements grid or Objects Measurements grid for all objects in the active ObjectSet. Please see Section *2.9.5 Measurements Task* for an explanation of this feature.

Statistical and Texture Measurements

Statistical measurements can be applied to all object types.

HaralickContrast: Contrast of the object in four directions. The mean value of the contrast is also computed.

HaralickCorrelation: Correlation of the object in four directions. The mean value of the correlation is also computed.

HaralickEnergy: Energy of the object in four directions. The mean value of the energy is also computed.

HaralickEntropy: Entropy of the object in four directions. The mean value of the entropy is also computed.

HaralickIDMoment: Inverse difference moment of the object in four directions. The mean value of the inverse difference moment is also computed.

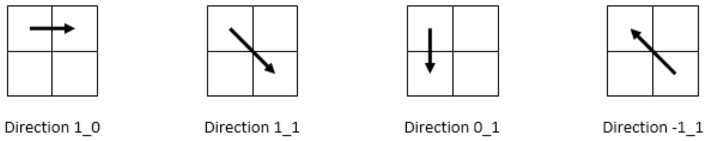


Figure 4.2-1: Coding of the directions for the Haralick's texture features.

$$\text{Energy} = \sum_{i=0}^{\text{Max}} \sum_{j=0}^{\text{Max}} p^2(i, j)$$

$$\text{Inverse Difference Moment} = \sum_{i=0}^{\text{Max}} \sum_{\substack{j=0 \\ j\neq i}}^{\text{Max}} \frac{p(i, j)}{(i-j)^2}$$

$$\text{Entropy} = -\sum_{i=0}^{\text{Max}} \sum_{j=0}^{\text{Max}} p(i, j) \times \text{Log}(p(i, j))$$

$$\text{Correlatio n} = \sum_{i=0}^{\text{Max}} \sum_{j=0}^{\text{Max}} \frac{(i-\bar{i}) \times (j-\bar{j}) \times p(i, j)}{\sigma_i \times \sigma_j}$$

$$\text{Contrast} = \sum_{i=0}^{\text{Max}} \sum_{j=0}^{\text{Max}} (i-j)^2 \times p(i, j)$$

Figure 4.2-2: Haralick's texture features calculated from the co-occurrence matrix.

Kurtosis: An object’s fourth standardized moment of pixel intensities.

Maximum: An object’s highest pixel intensity value.

Mean: An object’s mean pixel intensity value.

Median: An object’s median pixel intensity value.

Minimum: An object’s lowest pixel intensity value.

Skewness: An object’s third standardized moment of pixel intensities.

StandardDeviation: An object’s standard deviation of pixel intensities.

Size and Shape Measurements

These measurements can be applied to any object type except Points:

Centroid: The average location of all the pixels in an object, expressed as an x,y coordinate pair (i.e., center of mass of the object). For some objects, the *Centroid* can lie outside the object.

Height: The difference between the uppermost Y coordinate of an object and its lowermost Y coordinate, plus the Y pixel size. This value is expressed in calibrated units.

LogOfHeightToWidth: Log_{10} (Height/Width).

MajorAxis: The angle in radians from the X axis of the major axis of inertia. This object attribute provides the main orientation of the object relative to the X axis. The range of the Major Axis values lies between $-\pi/2$ and $+\pi/2$.

NumberOfBlobs: The number of connected components in an object using 4-connectivity.

PixelCount: Number of pixels comprising an object.

WeightedCentroid: The average of the intensity-weighted locations of all pixels in an object, expressed as an x,y coordinate pair. Thus, x_{WC} = sum of p_i times x_i divided by number of pixels, and similarly for

y_{WC}. For some objects, the *WeightedCentroid* can lie outside the object.

Width: The difference between the rightmost X coordinate of an object and its leftmost X coordinate, plus the X pixel size. This value is expressed in calibrated units.

The following measurements apply only to line objects (i.e., objects of type Line and objects of type Chain):

Angle: Angle (in radians) between the two first lines. The second point of the chain is the vertex of the angle. The range of the Angle values lies between $-\pi$ and $+\pi$.

Important notes: a) In Aphelion, clockwise angle values are positive since the Y axis is pointing down. b) All angular measurements are expressed in radian units (e.g., $\pi/2$ radians) unless specified otherwise in the text (90 degrees).

Curvature: Mean of the inverse of the curvature radii for all the points on the chain, divided by the object's *PixelCount*. A curvature radius value equals the length of the line that connects a point on a chain to the X,Y axes origin (point 0,0). The *Curvature* attribute computes a value using all possible radii for a chain.

Length: Number of pixels comprising a line object in 4-connectivity expressed in calibrated units.

The following measurements apply only to objects of type Line (i.e., straight line comprised of a single line segment):

MaxAbscissa: Distance between the image origin (0,0) and the furthest extremity of a Line object type. See Max T in *Figure 4.2-3*. This distance is expressed in calibrated units.

MiddlePoint: Middle pixel of a Line object type, defined in pixels.

MinAbscissa: Distance between the image origin (0,0) and the closest extremity of a Line object type. See Min T in *Figure 4.2-3*. This distance is expressed in calibrated units.

Rho: Perpendicular distance between a Line object type and the image origin (0,0), expressed in calibrated units.

Theta: Angle formed with the X axis by a line drawn from the image origin (0,0) perpendicular to a Line object type (see *Figure 4.2-3*), expressed in radians. The range of the Theta values lies between -π and +π.

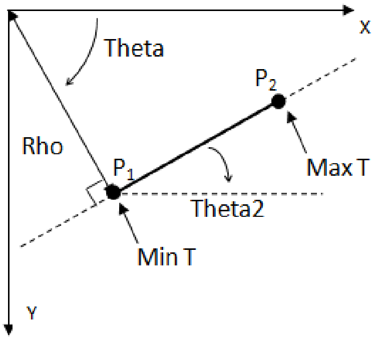


Figure 4.2-3: Attributes of type Line object (not type Chain).

Theta2: Angle between a Line object type and the X axis (see *Figure 4.2-3*), expressed in radians. The range of the Theta2 values lies between -π and +π.

The following measurements can be applied to any 2D object (i.e. Bitmap, Rectangle, Ellipse, and Rectangle):

Area: The area of an object, expressed in calibrated units.

BRFillRatio: Ratio between the *Area* of an object and the *Area* of its bounding rectangle. The bounding rectangle has the same orientation as the X,Y coordinate system of the image.

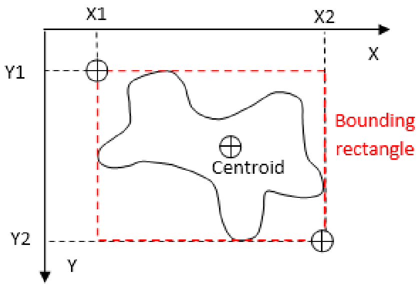


Figure 4.2-4: Bounding rectangle and centroid

Circularity: For a given object, this attribute is equal to $4\pi.Area/CroftonPerimeter^2$. It is equal to 1 for a circle and close to zero for a thin and long object.

Compactness: An object attribute that is equal to $16.Area/Perimeter^2$. It is equal to 1 for a square object oriented along the X,Y axes and approaches zero for irregular or long and thin objects.

ConvexArea: Area of the *convex hull* of an object. *ConvexArea* is always greater than or equal to *Area*. This measure is expressed in calibrated units.

Convexity: This attribute is equal to the area of an object divided by the area of its convex hull (see following *Help*).

Help: A *convex hull* is the smallest shape that can completely contain an object such that for any straight line segment connecting two points on the convex hull's boundary, the entire line segment is contained inside the convex hull.

ConvexMinAngle: The minimum of the angles formed by the adjacent pairs of line segments that comprise the polygonal boundary of an object, given in radians.

ConvexPerimeter: Perimeter of the convex hull of an object using the *Perimeter* measure and expressed in calibrated units.

CroftonPerimeter: An estimated value of an object's perimeter based on a more complex analysis than 4-connectivity. This analysis results in a more accurate perimeter estimate for most cases. This measure is expressed in calibrated units.

Help: *CroftonPerimeter* gives a more accurate estimate of an object's Euclidean (i.e., true) perimeter and is less sensitive than *Perimeter* to the object's orientation.

Elongation: The absolute value of the difference between the inertias of the major and the minor axes, divided by the sum of these inertias. The minor axis is defined as the perpendicular axis to the major axis. This measure is zero for a circle and approaches 1 for a long and narrow ellipse.

EquivalentDiameter: Gives the diameter of the circle whose area is equal to the area of the object, expressed in calibrated units.

EquivalentDiameterP: Gives the diameter of a circle whose perimeter equals the perimeter of the object, expressed in calibrated units.

Help: The Feret diameter is an important concept in image analysis. It is a line segment that is formed by the orthogonal projections of an object's boundary pixels onto a line of specified angle. *Figure 4.2-5* provides a graphical depiction of a Feret diameter.

Feret.Diameters: The set of projection lengths (i.e., Feret diameters) derived from an object, where the angle of each projection is $n.\partial\theta$, with n ranging from 0 to 89 and $\partial\theta$ is equal to $\pi/90$. The Feret diameters are expressed in calibrated units.

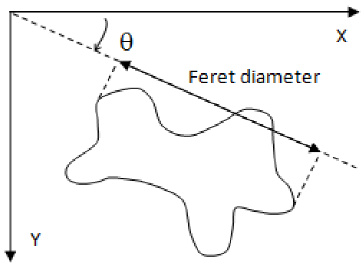


Figure 4.2-5: Feret Diameter for angle

Feret.Elongation1: This attribute is the ratio computed as the Feret diameter that is perpendicular with the *Feret.Max* diameter divided by the *Feret.Max* diameter.

Feret.Elongation2: This attribute is the ratio computed as the *Feret.Min* diameter divided by the Feret diameter that is perpendicular with the *Feret.Min* diameter.

Feret.Max: Maximum Feret Diameter in an object's set of Feret diameters.

Feret.Max.Theta: The angle of the Maximum Feret diameter (*Feret.Max*) for an object, expressed in radians. The range of the Feret.Max.Theta values lies between $-\pi/2$ and $+\pi/2$.

Feret.Mean: Mean of the Feret diameters of an object.

Feret.Min: Minimum Feret diameter in an object's set of Feret diameters.

Feret.Min.Theta: The angle of the Minimum Feret diameter (*Feret.Min*) for an object, expressed in radians. The range of the Feret.Min.Theta values lies between $-\pi/2$ and $+\pi/2$.

Feret.MinMaxRatio: Ratio between the minimum and the maximum Feret diameters for an object. This ratio is equal to 1 for a circle, is less than 1 for an elongated object, and equals zero for a straight line.

Feret.PentlandSphericity: Equal to $4.Area/(\pi.Feret.Max^2)$ for an object. This attribute is similar to the *Circularity* attribute with the *Feret.Max* replacing the *CroftonPerimeter*.

FractionOfSurfaceArea: The number of pixels in an object (*PixelCount*) divided by the number of pixels in the image region from which the ObjectSet was extracted.

HolesArea: Area of the holes in an object. A hole is one or more connected background pixels fully contained within an object.

HolesTotalArea: Total area of all the holes in all objects. A hole is one or more connected background pixels fully contained within an object.

Intercepts: Number of transitions from the background to the object in the 0°, 45°, 90° and 135° directions.

Key: The Key of an object is the index number of this object in its ObjectSet. Clicking on a Key in a measurements grid causes the associated object to be highlighted in the Visualization Window. Conversely, clicking on an object in the Visualization Window causes the measurement grid row associated with that object's Key to be highlighted.

Help: MBR stands for *Minimum Bounding Rectangle*. It is the smallest rectangle whose area wholly contains an object (i.e., minimizing the following formula):

$$\text{Feret.Diameters}(\theta) \times \text{Feret.Diameters}(\theta+\pi/2).$$

The MBR is usually not parallel to the X,Y image axes.

MBR.Area: *Area* of the MBR, expressed in calibrated units.

MBR.Center: Coordinates of the center of the MBR.

MBR.FillRatio: Ratio between the area of an object (*Area)* and the area of its MBR (*MBR.Area*).

Tip: *MBR.FillRatio* is an object shape attribute independent of the object's orientation. It is not the case of the attribute *BRFillRatio*. So it is better to use the attribute *MBR.FillRatio* instead of *BRFillRatio* to discriminate objects based on their shape.

MBR.Height: Length of the smallest side of the Minimum Bounding Rectangle (see *Figure 4.2-6*) of an object, expressed in calibrated units.

MBR.HeightWidthRatio: Ratio between MBR.Height and MBR.Width.

MBR.Perimeter: *Perimeter* of the MBR, expressed in calibrated units.

MBR.Polygon: List of an MBR's corner points as x,y coordinates.

MBR.Theta: Angle of the largest side of the MBR (labeled as MBR.Width in *Figure 4.2-6*). This angle is expressed in radians. The range of the MBR.Theta values lies between -π/2 and +π/2.

MBR.Width: Length of the largest side of the Minimum Bounding Rectangle (see *Figure 4.2-6*) of an object, expressed in calibrated units.

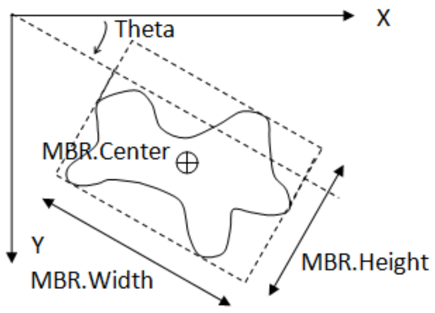


Figure 4.2-6: Minimum bounding rectangle measurements

NumberOfHoles: Number of holes in an object. A hole is one or more connected background pixels fully contained within an object.

Perimeter: An estimated value of an object's perimeter based on the number of 4-connectivity adjacent pixels along the object's boundary. This measure is expressed in calibrated units.

PerimeterRatio: Ratio between the perimeter of an object and the perimeter of its bounding rectangle, where the latter is oriented along the X,Y axes. The perimeter measure used for this ratio is *Perimeter* as described above.

PerimeterVariation: The sum of the direction changes between boundary pixels, where a 45-degree change equals 1, a 90-degree change 2, and a 135-degree change 3.

Projection.Count: A vector whose scalar elements are the number of pixels in each stripe of a set of contiguous stripes partitioning an object and having a given width and direction (angle measured from x axis). (*Figure 4.2-7*).

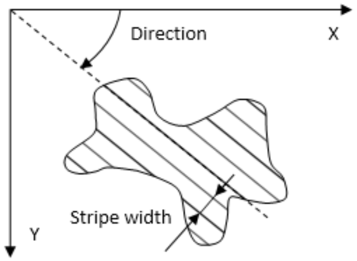


Figure 4.2-7: Direction and stripe width used to compute the Projection attributes Count, Mean, Width.

Projection.Mean: A vector whose scalar elements are the mean pixel value for each stripe of a set of contiguous stripes partitioning an object and having a given width and direction (angle measured from x axis). (*Figure 4.2-7*).

Projection.Width: A vector whose scalar elements are the lengths of each stripe of a set of contiguous stripes partitioning an object and having a given width and direction (angle measured from x axis). (*Figure 4.2-7*).

SymmetryMeanDifference: Mean of the absolute values of the length difference between the centroid and two opposite boundary points of an object (see *Figure 4.2-8*). This shape attribute is given in calibrated units. With *N* equal to the number of boundary points divided by two, the measurement's formula is:

Note: *SymmetryMeanDifference* is not computed when the centroid is outside the object, as one of the extremities may be undefined.

$$\frac{1}{N}\sum_{i=0}^{N-1}\left|\rho(i)-\rho'(i)\right|$$

With $\rho(i)=\left\|\overrightarrow{CX}\right\|$ and

$$\rho'(i)=\left\|\overrightarrow{CX'}\right\|$$

and $Angle\left(\overrightarrow{CX},\overrightarrow{CX'}\right)=\pi$

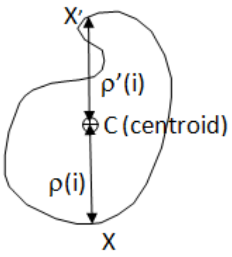


Figure 4.2-8: Symmetry mean difference

Volume: Sum of pixel intensity values in an object.

The following measurements can only be applied to objects of type Polygon:

PolygonArea: *Area* of a polygon, expressed in calibrated units.

5. LIST OF FUNCTIONS (Dev and SDK only)

The set of Aphelion functions for images and ObjectSets are described below.

5.1. *Images*

5.1.1. Arithmetic

Abs: Computes the absolute value of pixel values of an image

Add: Adds two images together

Blend: Performs an alpha blending combination of two images

Divide: Divides two images together

Invert: Inverts the pixel values of an image

LinearScale: Maps image values using a linear ramp function

Maximum: Returns the pixel-by-pixel maximum of two images

Minimum: Returns the pixel-by-pixel minimum of two images

Multiply: Multiplies two images together

Subtract: Subtracts two images

5.1.2. Color

Utility

ExcessRGB: Computes a set of excess color images from RGB images

ExpandRGB: Converts a one band image with a look up table into an color image (RGB image)

5.1.3. Edge Detection

CannyDericheEdges: Performs a Canny-Deriche edge detection on the input image

EdgesThin: Performs edge thinning by suppression of non-maximal magnitude edge values in a local neighborhood

MorphoGradient: Performs morphological edge detection by subtracting the eroded image from the dilated image

PrewittEdges: Produces edges at every pixel where the gradient magnitude is at least as great as the *magnitudeThreshold*

RidgeValleyEdges: Detects ridge & valley edges for a given input image

RobertsEdges: Applies two 2x2 kernels over the input image: along the NW-SE diagonal & along the SW-NE diagonal

SobelEdges: Produces edges at every pixel where the gradient magnitude is at least as great as the *magnitudeThreshold*

ZeroCrossing: Computes an edge image based on the zero-crossings of the non-directional second derivative of the image intensity values

5.1.4. Enhancement

EqualizeHistogram: Equalizes the histogram of an image

ShadingCorrection: Performs the shading correction of the input image based on a flat field image

5.1.5. Filtering

Convolve: Convolves an image using a user-defined M*N kernel

BoxFilter: Performs box filtering on an image using an arbitrary window size

Gaussian: Convolves an image using a kernel generated with a Gaussian shape

LowStop: Performs a fast low-stop filter by subtracting the results of a fast low-pass filter from the original image

Median: Performs median filtering using a user-defined M*N window

Mode: Performs mode filtering on an image

Nagao: Performs the edge-enhancement image smoothing algorithm proposed by Nagao & Matsuyama

RankValue: Provides a median-like filter where an element can be selected from the sorted list of pixel values

Weymouth: Performs image enhancement proposed by Weymouth and Overton

5.1.6. Frequency

FFT: Performs fast Fourier transform on an input image

InverseFFT: Performs fast inverse Fourier transform on an input image

5.1.7. Geometry

AffineMap: Performs an affine transform of the input image

Rotate: Performs an affine rotation of the input image about its midpoint

Scale: Performs an affine scaling of the input image

Shear: Performs an affine shear transform of the input image

Translate: Performs an affine translation of the input image

5.1.8. Logic

And: Performs logical AND of two images

BitAnd: Performs a pixel-by-pixel bitwise AND of an image

BitDifference: Finds the bit difference of 2 images

BitNot: Takes the bitwise complement of the pixel values in an image

BitOr: Performs a bitwise OR of an image

Difference: Performs logical difference

Not: Performs logical NOT of an image

Or: Performs logical OR of two images

Xor: Performs logical exclusive OR of two images

5.1.9. Matching

Correlate: Computes correlation of a template pattern with an input image

5.1.10. Math

ComplexFacet: Copies the selected band of a frequency image (real, imaginary, magnitude, or phase part) into a gray-scale image

Exp: Performs the Exponential function on every pixel

Exp10: Performs the Base-10 Exponential function on every pixel

ImaginaryPart: Copies the imaginary part of a frequency image into a gray-scale image

Log: Performs the Logarithm function on every pixel

Log10: Performs the Base-10 Logarithm function on every pixel

Magnitude: Copies the magnitude part (band) of a frequency image into a gray-scale image

MatrixMultiply: Multiplies a three-band image by a 3x3 user-specified matrix on a pixel basis

Phase: Copies the phase part (band) of a frequency image into a gray-scale image

RealPart: Copies the real part of a complex image into a gray-scale image

Sqr: Computes the Square of every pixel

Sqrt: Computes the Square root of every pixel

5.1.11. Measurements

Area: Computes the area of a binary image

Compare: Compares each pixel in an image with the corresponding pixel in a second image, returning the quantity of such pixel pairs that have equal intensity

Correlation: Computes the correlation index between 2 images

Distance: Computes the distance between 2 points in an image

Euler: Computes the Euler number (connectivity number) of a binary image

Histogram: Computes the intensity histogram of all pixels in the image

Intercepts: Computes the intercepts of a binary image

Moments: Computes the first & second order moments

ObjectCount: Returns the number of 4-connected objects/bitmaps in the input image

Range: Computes the minimum and maximum values of the image pixels for a given band

Variance: Computes the image variance

Volume: Computes the sum of all pixel intensities in an image

5.1.12. Morphology

Basic

ConstrainedThicken: Performs a constrained thickening which specified pixels cannot thicken

ConstrainedThin: Performs a constrained thinning which specified pixels cannot thin

Dilate: Performs morphological dilation of the input image

Erode: Performs morphological erosion of the input image

HitOrMiss: Performs a hit-or-miss transform on an input image

Thicken: Performs a thickening by a user-defined neighborhood configuration on a binary image

Thin: Performs thinning by a neighborhood configuration on a binary image

Distance

Distance: Computes for every non-zero pixel of an image the distance to the nearest zero pixel. The computed distance is a true Euclidean distance for the first overload, or a graph distance depending on the selected graph for the second overload

Enhancement

Contrast: Performs a morphological contrast on the input image

Features

LocalMaxima: Computes the binary image containing the local maximum pixels of a gray-scale image

LocalMinima: Computes the binary image containing the local minimum pixels of a gray-scale image

RegionalMaxima: Computes the binary image containing the regional maximum pixels or the h-maxima pixels of a gray-scale image depending on the specified overload

RegionalMinima: Computes the binary image containing the regional minimum pixels or the h-minima pixels of a gray-scale image depending on the specified overload

UltimateErodedSet: Computes the ultimate eroded sets, using the Euclidean or graph distance depending on the specified overload

Filtering

AutoMedian: Performs an automedian filter on the input image

Geodesy

BorderKill: Removes all connected components connected to the image borders

BorderKillAndHoleFill: Performs both a BorderKill and HoleFill

Dilate: Performs a geodesic dilation using a specified structuring element

Distance: Computes for every non-zero pixels the distance of the shortest path, inside the given mask, to the nearest pixel whose value is zero. The computed distance is a Euclidean distance when the first overload is selected or a graph distance depending on the specified graph

HoleFill: Puts all zero point sets surrounded by non-zero points to non-zero points

Reconstruct: Performs a reconstruction of an input image inside a reference image (mask)

Opening/Closing

AreaClose: Performs a closing based on the surface area of structures present

AreaOpen: Performs an opening based on the surface area of structures present

Close: Performs morphological closing of the input image

DilateReconsClose: Dilates the input image & reconstructs the resulting image

ErodeReconsOpen: Erodes the input image & reconstructs the resulting image

Open: Performs morphological opening of the input image

InfimumClose: Performs the intersection of closings by segments in all directions

SupremumOpen: Performs the combination of openings by segments in all directions

Segmentation

BlackTophat: Performs a Top Hat over the black structures (clusters of low intensity pixels) of the input image using the supplied structuring element

CatchmentBasins: Computes the catchment basins of the input image

SeededCatchmentBasins: Reconstructs the catchment basins of the input image from seeds

SeededWatershed: Reconstructs the catchment basins with the 1-pixel width boundaries (watershed) of the input image from seeds

SplitConvex: Uses the watershed operator to split connected blobs into convex blobs

Watershed: Computes the catchment basins with the 1-pixel width boundaries (watershed) of the input image

WhiteTophat: Performs a Top Hat over the white structures (clusters of high intensity pixels) of the input image using the supplied structuring element

Skeleton

ConnectedSkeleton: Computes a connected skeleton of a binary image

MinimalSkeleton: Computes the minimal skeleton of a binary image

OpenSkeleton: Computes the opening skeleton of a binary image

ThickenSkeleton: Computes a skeleton by thickening the input binary or gray-scale image

ThinSkeleton: Computes a skeleton by thinning the input binary or gray-scale image

5.1.13. Segmentation

AdaptivePercentileThreshold: Automatically computes the couple of thresholds such as the number of pixels between the two thresholds-total number of pixel ratio of the input image is equal to the defined percentile. The first overload returns the couple of computed thresholds while the second overload produces the output binary image computed by the Threshold function using the computed thresholds

Clusters: Produces a region-label image from a binary image based on a specified graph

EntropyThreshold: Automatically computes the threshold using the entropy of the gray-scale histogram of the input image. The first overload returns the threshold interval defined by the computed threshold and the maximum value of the input image, and the second overload produces the output binary image computed by the Threshold function using the threshold interval

HysteresisThreshold: Reconstructs the binary image produced by the Threshold function using the user-defined hysteresis thresholds from the binary image produced by the Threshold function using the user-defined seed thresholds

MaximumContrastThreshold: Automatically computes of the thresholds using the maximization of the contrast of the input image and produces a N-level image, where N is equal to the user-defined threshold count plus one

OtsuThreshold: Automatically computes the threshold using the Otsu's method (minimization of the intra-class variance). The first overload returns the threshold interval defined by the computed threshold and the maximum value of the input image, and the second overload produces the output binary image computed by the Threshold function using the threshold interval. A parameter enables the user to select either dark or bright objects in the image.

RegionGrow: Performs region growing on a gray-scale input image and produces a region-label image

SeededRegionGrow: Performs a region growing algorithm on a gray-scale image from a region-label seed image and produces a region-label image

Threshold: Produces a binary image such as output pixel values are equal to 1 between the lower and upper bound, else equal to zero

5.1.14. Utility

Clear: Set all pixels in an image to 0

Clip: Clips the pixel values of an image to a specified range

Copy: Copies the input image to a destination image

CopyBand: Inserts a single band image (gray-scale image) into a multi-band image as its N^{th} band

ExtractBand: Extracts the Nth band of a multi-band image to produce a single band image (gray-scale image)

Mask: Masks an image using another image as the mask

SubCopy: Copies the portion of the input image inside the region of interest into an output image resized to the dimensions of the area being copied from the input image. One of the function overloads supports multiple ROIs

5.1.15. Texture

LawsTexture: Computes the 9 Laws texture images for a given input image. The 9 single-band images are copied into a multi-band image

5.1.16. IO (Input/Output)

Import: Reads an arbitrary binary file into Aphelion image format

Read: Reads an image into an Aphelion Image

Write: Writes an Aphelion Image to a file in a specified image file format

5.2. ObjectSets

5.2.1. Display

Draw: Draws one spatial attribute of an ObjectSet in the overlay of an image

Erase: Removes one spatial attribute overlay for one ObjectSet from image display

ShowGrid: Opens the grid containing the ObjectSet

5.2.2. Chains

Generation

EdgelsToChains: Groups edgels into chains in an iterative process

5.2.3. Edgels

Grouping

EdgelNeighbors: Computes the set of near-by for each edgel in an ObjectSet

Generation

EdgesToEdgels: Converts edge data from an image format to edgel Objects

5.2.4. Filtering

Filter: Removes Objects from an ObjectSet falling outside the specified range for a numeric attribute

5.2.5. Geometry

AffineMap: Performs an affine transform of the input ObjectSet. The affine transform is defined by a 3x3 matrix. The result is defined as another spatial attribute of the input ObjectSet or in another ObjectSet depending on the selected overload

Rotate: Performs an affine rotation of the input ObjectSet about its midpoint. The result is defined as another spatial attribute of the input ObjectSet or in another ObjectSet depending on the chosen overload

Scale: Performs an affine scaling of the input ObjectSet. The result is defined as another spatial attribute of the input ObjectSet or in another ObjectSet depending on the specified overload

Translate: Performs an affine translation of the input ObjectSet. The result is defined as another spatial attribute of the input ObjectSet or in another ObjectSet depending on the specified overload

5.2.6. Lines

Generation

RegionLineFit: Fits a straight line for each region object of the input ObjectSet. The result is defined as another spatial attribute of the input ObjectSet or in another ObjectSet depending on the specified overload

ChainLineFit: Fits a straight line for each chain object of the input ObjectSet. The result is defined as another spatial attribute of the input ObjectSet or in another ObjectSet depending on the specified overload

5.2.7. Measurements

ChainMeasurements: Computes the attributes of each chain object in a given ObjectSet (curvature, length, and angle)

FeretShapeMeasurements: Computes the Feret attributes (minimum and maximum bounding box and all associated orientations) of each bitmap in an ObjectSet. The minimum bounding box (MBR) and its attributes are computed by this operator

Histogram: Computes a 1D histogram for a chosen measurement

Moments: Computes the first order moments of a chosen measurement (mean, standard-deviation, skewness, and Kurtosis)

Statistics: Computes the statistics (minimum, maximum average intensity, etc.) of a selected attribute for each object in an ObjetSet

StandardShapeMeasurements: Computes the shape attributes for each object in an ObjectSet

TextureMeasurements: Computes the texture attributes defined by Haralick (Energy, Entropy, Contrast, Inverse Difference Moment, Correlation) for each object in an ObjectSet

Note: The number of measurements computed by the functions above depends on the measurements selected in the Setting Task.

5.2.8. Polygons

Generation

ConvexHull: Computes the convex hull spatial attribute of each bitmap object in an ObjectSet

5.2.9. Bitmaps

Logic

Overlap: Computes the set of bitmaps which overlap for each bitmap in an ObjectSet

Morphology

Close: Performs a morphological closing on each bitmap in an ObjectSet

Dilate: Performs a morphological dilation on each bitmap in an ObjectSet

Erode: Performs a morphological erosion on each bitmap in an ObjectSet

HoleFill: Fills holes in bitmaps, creating a new bitmap attribute containing the filled bitmaps

Open: Performs a morphological opening on each bitmap in an ObjectSet

Generation

AdaptativePercentileThreshold: Produces a set of bitmaps from the output of the *Images.Segmentation.AdaptativePercentileThreshold* function

Clusters: Performs connected components on the input image and converts these components into bitmap objects

EntropyThreshold: Produces a set of bitmaps from the output of the *Images.Segmentation.EntropyThreshold* function

HysteresisThreshold: Produces a set of bitmaps from the output of the *Images.Segmentation.HysteresisThreshold* function

Labels: Converts each label of a label image into a bitmap object

MaximumContrastThreshold: Produces a set of bitmaps from the output of the *Images.Segmentation.MaximumContrastThreshold* function

OtsuThreshold: Produces a set of bitmaps from the output of the *Images.Segmentation.OtsuThreshold* function

RegionGrow: Produces a set of regions from the output of the *Images.Segmentation.RegionGrow* function

SeededRegionGrow: Produces a set of regions from the output of the *Images.Segmentation.SeededRegionGrow* function

Threshold: Produces a set of bitmaps from the output of the *Images.Segmentation.Threshold* function

ZeroCrossing: Produces a set of bitmaps from the output of the *Images.Segmentation.ZeroCrossing* function

5.2.10. Utility

Append: Combines two ObjectSets by adding the second set to the first

Copy: Duplicates an ObjectSet

Merge: Merges two ObjectSets into one

ToImage: Performs a raster conversion of a spatial attribute and writes the result into the output image. A data attribute is used to set the gray-scale value inside each object in the output image

6. OPTIONAL EXTENSIONS (Dev version only)

Aphelion Extensions are optional programs that extend the capabilities of the Dev product. These extensions include both standalone programs and programs that integrate seamlessly with Dev. Following are brief descriptions of fourteen optional extensions available when the Dev product was released. Check our web site for the availability of these and new extensions added in the future (*www.adcis.net* or *www.AmerinexImaging.com*).

6.1. 3D Image Processing and 3D Image Visualization Extensions

Aphelion Dev provides a rich array of functions for 2D images. But, Dev's capabilities are easily extended to include 3D images through use of Aphelion's optional 3D Image Processing extension. Most of the functions in Aphelion's libraries are available for the 3D space. With the extension, 3D data (X, Y, Z) can be processed and measurements can be computed. For example, functions such as filtering, labeling, and watershed are available in the extension and accessible through Dev's GUI. To work on a 3D image, just enter its name in the selected function's Input Image dialog box.

The 3D Image Display extension is also an add-on program for Dev. It provides tools to display 3D images, both before and after a function is executed. 3D images can be displayed in various modes (e.g., 2D sections, isosurface, and volume). LUT and transparency controls are also available.

An AVICreator sub-extension can record the motion of a 3D object, and generate an AVI file to be used in a presentation or a multimedia document.

6.2. Classification Extension

This new extension replaces the Recognition Toolkit module available for Aphelion 3.2.

The Aphelion™ Imaging Software Suite includes four tools for developing classification applications for automatically identifying the type or class of objects found in a set of images. The four tools are ClassifierBuilder (the classification application front-end which is a stand-alone application) and three classifier plug-ins: Fuzzy Logic, Neural Network, and Random Forest.

Once a classifier has been defined in ClassifierBuilder, it can be called in Aphelion Dev using the function **Classify**.

6.3. Color Extension

This extension includes a set of functions to convert color images from one color space to a different color space. Color spaces supported include RGB, HIS (Hue, Saturation and Intensity), LAB, YIQ, YUV, and more. Color segmentation functions such as region growing, morphological partitioning, color cube, and color sphere are also included in this extension.

6.4. Hardware interfaces

Optional software drivers are available to interface Aphelion to acquisition devices that do not meet the IEEE1394 IIDC and DirectShow protocols. These provide the capability to control an image acquisition device from within the Aphelion GUI. Video images can be captured within the Aphelion environment if the acquisition device is supported in Aphelion Dev. Contact an Aphelion representative to find out if a specific acquisition device is compliant with an Aphelion device interface.

6.5. *Image Registration*

This application was originally developed to register retinal images of different modalities, such as color and infrared images. This registration process has been adapted to work effectively with any image domain. The process is based on an algorithm called the Price method. It takes two images as input and generates their registration matrix.

6.6. *Kriging Extension*

This extension aids in the removal of acquisition noise in an image. The extension uses filtering techniques based on Kriging Analysis and Geostatistics. The underlying technique is based on the computation of experimental variograms, and the approximation of these experimental variograms using a linear combination of functions. The extension performs computation of the variograms and their approximation by functions, then utilizes user interaction to match a model on the real data, and generates a filtered image.

6.7. *Neural Network Toolkit*

The Aphelion Neural Network Toolkit enhances Aphelion Dev with an optional tool to automatically classify objects of interest based on a supervised classification. Objects are assigned to categories or classes based on a Neural Network algorithm. Probability and information based classifiers can be generated automatically, freeing the user from the necessity of specifying complex rules for object recognition and classification.

Classifier Builder, delivered with Aphelion Dev, provides efficient tool to manually classify objects and generate a training database used as the input of any Aphelion classifier including Neural Network Toolkit.

6.8. MultiFocus Extension

This new extension replaces the MultiFocus module available for Aphelion 3.2. It provides tools to generate a focused image from a stack of unfocused and registered images. Two methods are available in this extension: sharpness analysis and residue processing.

6.9. Stage Controller interface

The Stage Controller Interface provides convenient control of precision, optical and electronic microscope systems. For example, with this extension, users can fully control the motion of a microscope's stage in the X, Y, and Z directions.

6.10. Stereology Module

The Stereology Module implements long-standing, accepted stereology techniques that utilize a test grid overlaid on a slide image to characterize its microscopic structures. For example, with this module, accurate estimates of biomarker parameters can be computed, including volume fraction and length density. This module's techniques can be easily applied to virtual slides acquired using a high throughput imaging system (e.g., Aperio ScanScope® scanning system).

6.11. Virtual Image Capture

Virtual Image Capture is a stand-alone application to automatically generate a set of images by scanning a sample using an optical microscope equipped with a precision motorized stage* and a digital camera or a scanning electron microscope*. This application controls the stage and the camera, as well as the beam, the magnification, and other parameters that need to be set for a scanning electron microscope. The main control parameters are: scan mode, image size, overlap size, and scanning area definition. The images captured by Virtual Image Capture are saved together as a project, along with all scanning information, stage positions, and pixel size in real units.

* Contact your Aphelion representative to verify that a specific stage controller and/or scanning electron microscope are supported by this application.

6.12. Virtual Image Stitcher

Virtual Image Stitcher is a stand-alone application used to reconstruct a larger image from a set of smaller overlapping images. By combining several smaller images, Virtual Image Stitcher builds a seamless composite image that can be larger than the available memory space of the computer. This new composite image can then be exported as a whole image (depending on the memory space of the computer) or as tiles. Virtual Image Stitcher includes a powerful and intuitive viewer that enables the user to browse virtually any size image.

For microscopy applications, Virtual Image Stitcher aids in the formation of an image from a set of high resolution subimages in order to have one single, high resolution image of the entire sample. Virtual Image Stitcher enables accomplishment of this process without the microscope having an expensive, automatic stage.

The stitching position of each individual image is saved in a project folder generated by Virtual Image Capture. Images captured by third party system can also be loaded in Virtual Image Stitcher.

6.13. VisionTutor

The VisionTutor Computer Vision Course is a combined theoretical and laboratory package intended for an introductory course in Image Processing and Image Understanding. The format of the course makes it possible for colleges, universities, research labs and in-house trainers to present computer vision techniques in a manner that will prepare students and end-users to both understand and apply the algorithms and techniques of this field. The format of this course is unique, as it consists of computer-based text, graphics, figures and images. Image Processing operators are illustrated in a stand-alone application that calls for the operators theoretically presented in the Lecture guide and applied on a set of typical images.

7. MACRO SCRIPTING (Dev version only)

Aphelion Dev supports four macro languages: BasicScript, Python, Visual C#, and Visual Basic. The first two of these languages each have dedicated coding environments as tabbed panels in window D. The latter two share the same panel. The tabbed panels enable coding of macros without leaving the Dev user interface. Coding for macros can be simultaneously open in the three coding panels.

The following table gives the main differences between the four scripting languages. The + sign ratings are an indication of the relative strength a language has with respect to a listed capability. The ratings are based on the collective experiences of the Aphelion engineering team.

Table 6.13-1: Main differences between macro languages supported in Dev

Language Capability	**BasicScript**	**Python**	**Visual Basic & C#**
Debug tools	Yes	No	No
Recording	Yes	No	No
Dialog generation	Yes^{+}	Yes^{+++}	Yes^{+++}
Interactive execution	Yes^{+}	Yes^{+++}	No
Reusability	No	Yes^{+++}	Yes^{+++}
Application automation	No	Yes^{+++}	Yes^{+++}
3rd party components	No	Yes^{+++}	Yes^{++}

Dev provides a macro recording feature that enables a user to easily record sequences of functions without coding.

Executing and Debugging a Macro

A full BasicScript execution and debug environment is included within Dev. User controls are provided for executing macros, stepping through macro instructions, monitoring variable values, and more. These facilitate macro debugging and can be used to create a demo where the steps of the demo are recorded in a macro and executed one at a time.

For more information on this topic, open the online Help system and type *Macros* in the Index tab.

Creating Dialogs

BasicScript includes a complete dialog building environment. You can create a dialog and add buttons, type-in areas, pull-down lists, etc. This is useful for creating applications, which present a limited number of options to the user, such as three filtering options, followed by measurement options. For more information, open the online Help system and type *Dialog Editor* in the Index tab.

7.1. Aphelion Macros

The following macros are provided in the standard Aphelion distribution. We recommend that you run each macro and view the results. For an annotated version of Ceramic.apm, see the section *7.1.3. Macro Example (Ceramic.apm)*. There are three folders containing macros in the Aphelion distribution:

- **Macros 32-40**: BasicScript macros available in Aphelion version 3.2 (Macros folder) and ported in the 4.x version;
- **Examples 32-40**: BasicScript macros available in Aphelion 3.2 (Examples folder) and ported in the 4.x version;
- **Macros**: BasicScript, Python and C# macros.

7.1.1. Application Macros

The application macros listed below are stored in the *Macros32-40* folder. To load an application macro, select the **BasicScript** tab, right-click and select *Load* from the drop-down menu. Then choose a macro from the *Macros32-40* folder.

Table 7.1-1: List of macros available in the *<Aphelion Installation Path>\Macros32-40* folder

Macro Name	Purpose Demonstrated
BasicExamples.apm	A variety of useful programming examples using BasicScript
Blood.apm	Basic object creation & measurement computation
Ceramic.apm	Basic image processing & object extraction techniques
Circuit.apm	Morphological operators to detect a defect
Color.apm	Color processing
Confocal.apm	RGB to HSI conversion
DemoDialog.apm	Dialog presenting a menu of applications demos
DemoLoop.apm	Loops to apply many operators to all images in a directory
FeretAccess.apm	Accesses the Feret diameters in an ObjectSet

Fourier.apm	Demonstrates various frequency domain filters
Grille.apm	ObjectSet measurements and morphological operators
ImgRegister.apm	Registers an input image to a reference image
ImmunoMarker.apm	Uses color information to detect cells
Muscle.apm	Morphological operators to detect cancerous cells based on their gray-scale intensities
Road.apm	Computes boundary chains and demonstrates edgels, chains and lines
RoiProcess.apm	Demonstrates conversion of an ObjectSet of regions to ROIs
SequenceInterface.apm	Uses a dialog for acquiring a sequence of images
WCCO.apm	Watersheds used to segment a multi-phase image

7.1.2. Example Macros

The example macros are stored in the *Examples32-40* folder. To load an example macro, select the BasicScript tab, right-click, and select *Load* from the drop-down menu.

Table 7.1-2: List of macros available in the *<Aphelion Installation Path>\Examples32-40* folder

Macro Name	Purpose
3DSlicer	Display a 2D section selected from a set of 2D sections that compose a 3D image Note that this macro can be run only if the 3D Image Processing extension is licensed
BrowseDirectory.apm	Browses a folder containing images from BasicScript
ColorImageAccess.apm	Shows how to access pixels in a color image from BasicScript.
ComplexImageAccess.apm	Shows how to access pixels in a complex image from BasicScript.
Dialog.apm	Displays a dialog box to run an application
EdgeImageAccess.apm	Shows how to access pixels in an Edge image from BasicScript.
ExportToExcel.apm	Starts Excel and loads an image histogram into a worksheet
ExportToNotepad.apm	Starts Notepad and loads the target text into a Notepad text document.
ExportToWord.apm	Starts Word and loads an image and some text into a Word document.
FileAccess.apm	Demonstrates reading and writing a file from BasicScript

GlobalMeasurements.apm	Computes the default global image measurements
HistogramAccess.apm	Accesses a histogram's values from BasicScript
ImageAccess.apm	Reads and writes pixel values in an image loaded in memory from BasicScript
ImageRefresh.apm	Turns updating of the display OFF and ON as a means for saving processor time
ImgBoxes.apm	Turns regions into their minimum bounding boxes and filters them
ImgChains.apm	Starts with a gray-scale image and extract chains
ImgHistogramChart.apm	Uses chart functions to display an image histogram
ImgLines.apm	Starts with a gray-scale image and extract lines
ImgRead.apm	Loads an image. This macro can be used as a prefix in another macro to load an image from storage
LineROI.apm	Creates processing regions of interest (ROIs) which are derived from a line ObjectSet
Lines2Regions.apm	Converts line objects into bitmaps for measurement purposes

ObjectAccess.apm	Accesses attribute values from an ObjectSet using BasicScript
ProcessDialog.apm	Demonstrates how to keep a simple dialog box open so that multiple buttons can be pressed to perform image processing operations
RegionMorphology.apm	Morphology performed on an ObjectSet of bitmaps
RegionRotate.apm	Rotation of an ObjectSet
Regions2Lines.apm	Fits lines onto objects of an ObjectSet
ShowImageSequence.apm	Shows how to display any sequence of 2D images

7.1.3. Macro Example (Ceramic.apm)

This macro example is an annotated version of the macro Ceramic.apm, one of the application macros distributed with Aphelion (available in the *<Aphelion Installation Path>\Macros32-40*). In this macro, the goal is to:

- Extract all fibers (dark, circular objects);
- Measure the surface, elongation and compactness of each fiber;
- Provide the fiber position in pixel coordinates.

The image, Ceramic.tif, was obtained with a scanning electronic microscope. The pixel values range from 1 to 255. In this macro, the fibers are segmented from the background using a threshold. Morphological functions are used to clean the image. Local analysis is performed to get measurements on each fiber. From the measurement data, the fibers are filtered.

This macro also shows how to use BasicScript to create dialog boxes.

Note: In this example, the macro code appears in boxes, with the annotations in regular text. For more information about a specific BasicScript command, refer to the online Macro Reference Guide, available from the Help menu in Aphelion.

In BasicScript, all comments made in a macro must start with a quote. An alternative is to use the rem command.

```
'------------------------------------------
'* Ceramic.apm                            *
'------------------------------------------
'* Originator: GG                         *
'* Date of creation: 02/17/10             *
'* Date of last modifications: 03/01/10   *
'------------------------------------------
```

Each subroutine must have a name. In Aphelion, the name by default is main, but any other name can be given to a subroutine. A subroutine can call another one, if the other subroutine is placed in the same file.

The main program is the subroutine named main. To be run, a macro should include a subroutine named main.

```
Sub main
'------------------------------------------
'* Declarations
'------------------------------------------
'The following declarations define two integer
variables
breaktime and bt1000.
Dim breaktime As Integer
Dim bt1000 As Integer
```

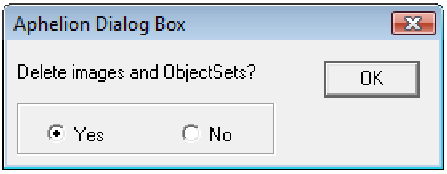
The following is the definition of a dialog box to specify whether all images are freed after you run the macro. The values 180 and 48 correspond to the size of the dialog box. The string "*Aphelion Dialog Box*" is the text which appears in the dialog box banner.

```
Begin Dialog YesNoDialog,,180,48,_
"Aphelion Dialog Box"
    OKButton 132,8,40,14
    GroupBox 4,20,108,24,"",.GroupBox1
    Text 4,8,108,8,"Do you want to free all
images?",_
    .Text1
    OptionGroup .OptionGroup1
    OptionButton 16,32,32,8,"YES",.OptionButton1
    OptionButton 72,32,32,8,"NO",.OptionButton2
End Dialog
```

The variable named *ABox* is defined as a dialog box of type *YesNoDialog*.

```
Dim ABox As YesNoDialog
```

When this dialog variable is used in a dialog call, it will appear in Aphelion as:



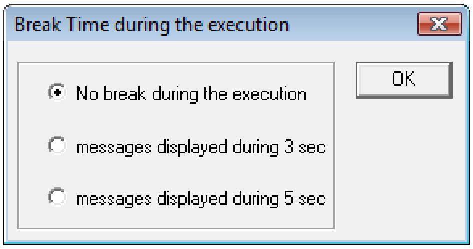
The following is the definition of a dialog box to specify break time. The variables *OptionButton1*, *2*, *3* will contain the value 0 or 1, depending on the selection. Other numerical values correspond to the size and position of the **OK** and option buttons.

```
Begin Dialog BreakTimeDialog ,,189,76,"Break
Time_
during the execution"
    OKButton 144,8,40,14
    GroupBox 4,4,132,68,"",.GroupBox1
    OptionGroup .OptionGroup1
    OptionButton 16,16,108,8,_
    "No break during the execution",.OptionButton1
    OptionButton 16,36,116,8,_
    "messages displayed during 3
sec.",.OptionButton2
    OptionButton 16,56,116,8,_
    "messages displayed during 5
sec.",.OptionButton3
End Dialog
```

The variable named *Breakbox* is defined as a dialog box of type *BreakTimeDialog*.

```
Dim BreakBox As BreakTimeDialog
```

When this dialog variable is used in a dialog call, it will appear in Aphelion as:


Break Time during the execution

[✓ No break during the execution]

messages displayed during 3 sec

messages displayed during 5 sec

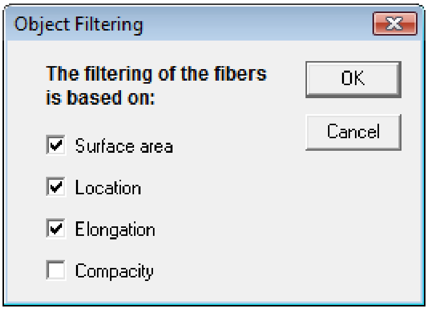
Following is a definition of a dialog box to specify which filters are applied on the measurements. All numerical values correspond to positions in pixels and sizes.

```
Begin Dialog FilterBox,,172,100,"Object
Filtering"
    OKButton 124,8,40,14
    CancelButton 124,28,40,14
    Text 16,8,92,20,_
    "The filtering of the fibers is based on:",_
    .Text1,"Arial",9,ebBold
    CheckBox 16,36,68,8,"Surface area",.surface
    CheckBox 16,52,68,8,"Location",.location
    CheckBox 16,68,68,8,"Elongation",.elongation
    CheckBox 16,84,68,8,"Compacity",.compacity
End Dialog
```

The variable named *FBox* is defined as a dialog box of type *FilterBox*.

```
Dim FBox As FilterBox
```

When this dialog variable is used in a dialog call, it will be displayed in Aphelion as:


Object Filtering

The filtering of the fibers is based on:

[✓] Surface area
[✓] Location
[✓] Elongation
[ ] Compacity

Aphelion supports both 4- and 8-connectivity. The two following variables are used for the labeling operation, to specify the number of neighbors of each pixel.

```
graph4C = AphNgbGraph("2D 4-connected")
graph8C = AphNgbGraph("2D 8-connected")
```

After all the declarations, the code begins.

```
'-----------------------------------------
'* Main program                          *
'-----------------------------------------
```

First, the dialog box to select the time between two operators (break time) is displayed, using the *Dialog* function and the *BreakBox* variable. This dialog returns two values, -1 if the **OK** button is pressed, or 0 if the **CANCEL** button is selected. As there is no **CANCEL** button, the dialog can only return -1. When the **OK** button is selected, the variable *BreakBox.OptionGroup1* tells which option has been selected. This variable can take three values, 0, 1 and 2.

```
response% = Dialog(BreakBox)
```

The *Select Case* command allows you to test a variable, and to branch to any line depending on the value. This is useful for handling the different button selections from a dialog.

```
Select Case response%
    Case -1' OK button
       If BreakBox.OptionGroup1 = 0 Then
breaktime=0
       Else
          If BreakBox.OptionGroup1=1 Then
             breaktime = 3
          Else breaktime = 5
          End If
       End If
    Case 0  ' Cancel button
```

```
      MsgBox "Cancel button doesn't exist!"
End Select
```

All time values in BasicScript are given in milliseconds. Thus the number of seconds specified by the variable *BreakBox.OptionGroup1* is multiplied by 1000.

```
bt1000 = breaktime * 1000
```

Load the image Ceramic.tif from disk and create a new image in Aphelion, using the *AphImgNew* operator. The full path to the image file has to be specified.

```
original = AphImgNew()
AphImgRead original,_
"c:\Aphelion 4.2.0\Dev\Images\ Ceramic.tif"
```

The *Msg.Open* command will display a string in a pop-up box. The *Sleep* command creates a pause depending on the time lapse defined. After opening a message window, you must close that window using *Msg.Close*.

```
Msg.Open "Composite material in a ceramics
matrix",_
breaktime, False, False
Sleep bt1000
Msg.Close
```

The following are a number of messages to guide the user. Usually, the call to *AphImgThreshold* has three parameters, the two images and the bounds. If one of the parameters such as threshold bounds has to be user specified while running the macro, then the parameter should be replaced by *Aph4MissingParameter()*. A default value can be specified by adding *AphThreshold (0,100)* between the two empty parentheses. In that case, as one of the three parameters has been omitted, Aphelion will prompt the user to enter those values.

```
Msg.Open "The gray-level histogram of this image_
Includes two modes", breaktime, False, False
Sleep bt1000
Msg.Close
Msg.Open "A threshold will segment the two
phases",_
breaktime, False, False
Sleep bt1000
Msg.Close
Msg.Open "Specify Low as 1 and High as 117", 0,
False,_
False
Sleep 3000
```

The following lines show a threshold of the original image into a new image named *Binary1*. This name will appear in the window banner, and will also allow the user to work later on this image.

```
b1 = AphImgNew("Binary1")
AphImgThreshold original,b1, AphThreshold(0,117)
Msg.Close
Msg.Open "Interactive gray-level Thresholding",_
breaktime, False, False
Sleep bt1000
Msg.Close
```

The binary image resulting from the threshold includes the fibers and also parts of the background (the darkest pixels). However, the fibers are much larger than the background noise, so you can apply a morphological dilation to the original gray-scale image (with a structuring element whose size is greater than the size of the objects in the background, but smaller than the size of the fibers) to remove the darkest pixels.

The following lines perform a gray-scale dilation of size 3, based on a square-structuring element. The dilation is performed on the original image, and the output image is named *Image1*.

```
i1 = AphImgNew("Image1")
AphImgDilate original,i1,AphSElement("Square",3)
```

A new threshold is performed to extract the core of the fibers. The two bounds are set to 0 and 117. The threshold is performed on the result of the previous dilation, and the output image is named *Binary2*. *Binary2* contains the seed points for the fibers, but not for the background noise.

```
b2 = AphImgNew("Binary2")
AphImgThreshold i1,b2,AphThreshold(0,117)
```

The following call to *AphImgReconstruct* performs binary reconstruction of the first binary image, using seeds detected after the second threshold. The output result will contain all blobs from the first threshold image which contain one or more seed pixels in the second threshold image. The binary reconstruction is based on the 4-connectivity, which means that every pixel in the binary image has four neighbors, in the 0, 90, 180 and 270 degree direction.

```
b3 = AphImgNew("Binary3")
AphImgReconstruct b1,b2,b3,graph4C
```

The following call to *AphImgFree* deletes the first two binary images, *Binary1* and *Binary2*, which are not needed anymore.

```
AphImgFree b1
AphImgFree b2
```

Some of the fibers in the remaining binary image are connected. The use of binary segmentation operators based upon mathematical morphology techniques will segment all the fibers. The following lines call *AphImgClustersSplitConvex*, an operator based on the watershed algorithm.

```
b4 = AphImgNew("Binary4")
AphImgClustersSplitConvex b3, b4, 10, graph4C
```

Now *AphImgHoleFill* is used to fill the holes within blobs (fibers) in the previous binary image.

```
AphImgHoleFill b4, b4, graph8C
```

AphImgBorderKill will eliminate the objects intersecting the border of the image (partial fibers). Notice that both input and output images are identical. This feature is supported by Aphelion for all the operators which have input and output images of the same class and the same type. The operator is based on 4-connectivity.

```
AphImgBorderKill b4, b4, graph4C
```

The following *AphImgClustersToLabels* function takes a binary image as input, and generates a labeled image. In the labeled image, each object has a different gray-level value. All the pixels of the same connected object have the same value. The labels go from 1 to the number of objects. The first object is the first one hit while scanning the image from top to bottom and left to right. The connectivity is very important while performing a labeling.

```
l1 = AphImgNew("Label")
AphImgClustersToLabels b4, l1,_
AphNgbGraph("2D 4-connected")
AphImgFree b4
```

After a labeling, you can change the image representation from the pixel space to a symbolic representation. The ObjectSet named *FIBERS* is created after calling the Aphelion *AphImgLabelsObj* function. The *FIBERS* ObjectSet has a spatial representation of the fiber regions, and a few basic attributes (scalar attributes), such as the number of pixels in each bitmap. One way to examine this data is to use the grid feature available in Aphelion. A grid is similar to a spreadsheet, where each row corresponds to a label or object, and each column is an attribute or measurement.

```
AphImgLabelsObj l1, AphObjNew("FIBERS")
```

After creating the *FIBERS* objects, it is possible to analyze each bitmap, and compute their convex hull from the region representation. This is the best way to compute the convex hull of objects, from their boundary representation.

```
AphRegionConvexHullPolygons AphObj("FIBERS")
```

The *FIBERS* ObjectSet can be viewed in the grid (scalar attributes) or overlaid on top of the image (as a filled bitmap, or as the boundary of each object). The operator *AphObjDraw* overlays the *FIBERS* ObjectSet, seen as polygons, on top of the image named original.

```
AphObjDraw original, AphObj("FIBERS"), "POLYGON"
```

The ObjectSet contains extensive information about the fibers. You can filter the objects based on their associated measurements, or use *AphObjComputeMeasurements* to compute more attributes of the fibers.

PIXELCOUNT	Number of pixels within each object
BITMAP	Coordinates of the bitmap bounding box (upper left and lower right corners)
POLYGON	Coordinates of the convex hull bounding box (upper left and lower right corners)

The FBox dialog box lets the user select which filter to apply. There are four types of filters available: surface, location, elongation, compacity.

```
response% = Dialog(FBox)
Select Case response%
   Case -1' OK button
      If FBox.surface Then
         i1 = AphImgNew("Surface")
         AphImgCopy original, i1
```

The following line filters the ObjectSet *FIBERS* into a new ObjectSet named *SURFACE*. The attribute involved in the filtering is *PIXELCOUNT*. The lower and upper bounds are set respectively to 1000 and 1400.

```
         AphObjFilter AphObj("FIBERS"),_
         AphObjNew("SURFACE"), "PIXELCOUNT",_
         1000, 1400
```

AphObjDraw will overlay the resulting ObjectSet on top of the original image.

```
         AphObjDraw i1, AphObj("SURFACE"),_
         "POLYGON"
      End If
```

The following lines filter based upon the position of the objects in the image. As the lower labels are in the upper part of the image, it is sufficient to filter on the *INDEX* variable. The two bounds are set to 1 and 10. A new ObjectSet named *LOCATION* is created.

```
      If FBox.location Then
         i2 = AphImgNew("Location")
         AphImgCopy original, i2
         AphObjFilter AphObj("FIBERS"),_
         AphObjNew("LOCATION"), "TOKEN_INDEX", 1,
10
         AphObjDraw i2, AphObj("LOCATION"),
"POLYGON"
         Msg.Open "Selection of the 10 first
fibers",_
         breaktime, False, False
         Sleep 2000
         Msg.Close
      End If
```

The following lines filter based on the elongation of the objects. The elongation has to be measured on the regions, and not the boundaries. The two bounds of the filter are set to 0 and 0.05. A new ObjectSet named ELONGATION is created.

```
      If FBox.elongation Then
         i3 = AphImgNew("Elongation")
         AphImgCopy original, i3
         AphRegionShape AphObj("FIBERS"), "REGION"
         AphObjFilter AphObj("FIBERS"),_
         AphObjNew("ELONGATION"), "ELONGATION",_
         0, 0.05
         AphObjDraw i3, AphObj("ELONGATION"),_
         "POLYGON"
         Msg.Open "Fibers with elongation between
0 and 0.05",_
         breaktime, False, False
         Sleep 2000
         Msg.Close
      End If
```

The following lines filter based on the compacity of the objects. The first call to *AphObjAttributeRatio* generates a new attribute named

A1 which is equal to the *PERIMETER* divided by the *AREA* of each object. The second call defines a new attribute *A2* equal to *A1* divided by the *PERIMETER*, or the *AREA* divided by the square of the *PERIMETER*. The compacity is defined from this ratio. The two bounds of the filter are set to 0.045 and 0.06. This filter shows how new attributes can be defined from already computed ones. For further information on creating new attributes from BasicScript see the *Obj** macro functions. The final result is displayed on top of the original image.

```
       If FBox.compacity Then
          i4 = AphImgNew("Compacity")
          AphImgCopy original, i4
          AphRegionShape AphObj("FIBERS"), "REGION"
          AphObjAttributeRatio AphObj("FIBERS"),_
          "PIXEL_COUNT", "PERIMETER", "A1"
          AphObjAttributeRatio AphObj("FIBERS"),
"A1",_
          "PERIMETER", "A2"
          AphObjFilter AphObj("FIBERS"),_
          AphObjNew("COMPACITY"), "A2", 0.045, 0.06
          AphObjDraw i4, AphObj("COMPACITY"),
"POLYGON"
          Msg.Open "Fibers whose shape is close to
a circle",_
          breaktime, False, False
          Sleep 2000
          Msg.Close
       End If
    Case 0' Cancel button
    MsgBox "Cancel button doesn't exist!"
End Select
```

The following dialog box asks whether to delete all the images and objects created during this macro. The first choice is to delete all images, and the value of the .OptionGroup1 is then 0. The variable response% is the returned value of the dialog box.

```
response% = Dialog(ABox)
If ABox.OptionGroup1 = 0 Then
    AphImgFree original
    If (i1<>0) Then
       AphImgFree i1
       AphObjFree AphObj("SURFACE")
    End If
    If (i2<>0) Then
       AphImgFree i2
       AphObjFree AphObj("LOCATION")
    End If
    If (i3<>0) Then
       AphImgFree i3
       AphObjFree AphObj("ELONGATION")
       AphImgFree i3
    End If
    If (i4<>0) Then
       AphImgFree i4
       AphObjFree AphObj("COMPACITY")
    End If
    AphObjFree AphObj("FIBERS")
End If
```

This is the end of the macro and the subroutine main. The call to the function *End Sub* ends the subroutine.

```
End Sub
```

7.2. C# Programming Language

See *APPENDIX G - C# Programming Language* section for detailed information about the C# programming language in Aphelion.

7.3. Python

See examples provided in the folder "<Aphelion Dev installation folder>\Macros\Python".

8. PROGRAMMING EXAMPLES (Dev and SDK only)

Users of Aphelion SDK should review the examples provided in the folder called *Examples* in the SDK folder. Aphelion DEV users need to install the Aphelion SDK if they wish to review those examples.

The examples provided as Microsoft Visual Studio solutions show how to develop an application that uses Aphelion's libraries. The examples include the following support:

- 32 and 64 bit architectures;
- Native and Managed code.

Note: We recommend reviewing the example called Pads. It is easy to understand and illustrates some basic image processing. Section *8.3 Pads Example (Visual C# code)* describes the managed code version of the Pads example.

There are three subfolders in *Examples*, one contains the programming examples as Native code, the second contains the examples as Managed Code, and the third contains temporary files ".obj" generated by the compiler. This latter folder does not exist immediately after the first installation of SDK.

8.1. Native Examples

The Native folder (Win32) includes two C++ examples, named Pads and Clocks. Visual C++ 2013 or newer is needed to compile these two examples. However, the executable versions of these files are provided for those who do not have the compiler. Both 32-bit and 64-bit versions of the executable files are included.

The *Pads* example illustrates how to use basic image processing and morphology to detect malformed solder pads (i.e., defects) on a circuit board. This is done by extracting the small, circular pads as objects, computing their measurements, and then filtering the pads based on their shape.

The *Clocks* example demonstrates how to differentiate between six clocks in an image and locate the hands of the clocks. This example uses irregular regions of interest, derived from an edge detection function, to isolate each individual clock, and then extracts the dark objects on the clocks corresponding to the clock hands.

8.2. .NET Examples

The Managed Code folder (dotnet) includes two subfolders, one called *Applications* and the other called *Help.Examples*. *Help.Examples* is a C# project that calls for any Aphelion function. We recommend using this C# project to get the syntax of each Aphelion image and ObjectSet function. This project is the foundation for generating the Aphelion Online Help examples.

The *Applications* folder includes four subfolders corresponding to four distinct applications: *Ceramic*, *Clocks*, *Pads*, and *WCCo*. For each application, individual projects (*.csproj) are provided for both 32-bit and 64 bit systems. Visual Studio .Net or Visual C# Express (freeware product) are required to open and execute these projects. .Net components can also be called in Python, Visual C++, and Visual Basic projects.

Note: SharpDevelop is a free, open source programming environment that can be used to open and manage Aphelion C# projects.

The *Ceramic* example demonstrates the use of a threshold function, morphological operations, and ObjectSet management to extract dark, circular objects from an image. The *WCCo* example uses an image that contains carbide objects in a cobalt matrix, and illustrates how to extract multiple phases from an image. The *Pads* and *Clocks* applications were explained in the prior section.

Each example project includes a graphical user interface for entering parameters values required for proper execution of each phase of the project.

8.3. Pads Example (Visual C# code)

Following is a detailed description of the image processing code used in the *Pads* example.

```
// Define the image folder to be able to load the
image in the GUI and process it

String imagePath = Application.ExecutablePath.
ToLower().Replace("pads.exe",
@"..\..\..\Images");

Adcis.Images.IO.Globals.ImageFolders.Add(imagePat
h);

// Create the input image and read the image file
Pads.tif from the disk. Other image formats are
supported in Aphelion (bmp, jpg, etc.). The image
is automatically loaded when starting the
application

m_imageInput = new Adcis.Images.Image();
m_imageInput.Read("Pads.tif");

// Keep the default resolution as pixels. If a
resolution is already defined for the image, then
there is no need to include the line below

m_imageInput.Resolution = null;
```

```
// Display the input image in the image window
that has been defined in the form

m_imageView.Data.Add(m_imageInput);

// Write some text in the text box located at the
bottom of the GUI to guide the user

richTextBoxComment.Text = "Click on Detect, Pads
to process the image";
richTextBoxComment.Refresh();

// User has selected the Execute entry in the
menu

private void padsToolStripMenuItem_Click(object
sender, EventArgs e)
{
    [...]

    // Top Hat transform to detect the pads even
    on a non uniform background. The input image
    is m_imageinput, the output image is called
    imageTophat and the structuring element
    involved in the process is a square of size
    15.

    // Create a new image

    Adcis.Images.Image imageTophat =
    new Adcis.Images.Image();
    Adcis.Images.Morphology.Segmentation.Functions
    .WhiteTophat(m_imageInput, imageTophat,
    PredefinedStructuringElementEnum.pseSquare, 15);
```

```
// Threshold the result of the tophat
transform between 101 and 232

Adcis.Images.Image imageThreshold =
new Adcis.Images.Image();
Adcis.Images.Segmentation.Functions.Threshold
(imageTophat, imageThreshold,
new Adcis.Math.IntervalD(101, 232));

// To remove small artifacts in the image,
perform an opening followed by a
reconstruction

Adcis.Images.Morphology.OpeningClosing.
Functions.ErodeReconsOpen(imageThreshold,
imageThreshold, new Adcis.Geometry.
StructuringElement(Adcis.Geometry.
PredefinedStructuringElementEnum.pseSquare,
7), PredefinedNgbGraphEnum.pngSquare8);

// Convert the binary image into an Aphelion
ObjectSet

Adcis.ObjectSets.ObjectSet osPads =
new Adcis.ObjectSets.ObjectSet();
Adcis.ObjectSets.Regions.Segmentation.
Functions.Clusters(imageThreshold, osPads,
PredefinedNgbGraphEnum.pngSquare4);

// Display the detected pads as an ObjectSet
in the graphics overlay

m_imageView.Data.Add(osPads.Attributes
["REGION"]);
```

```
// Compute the maximum Feret diameter (using
directions every PI / 180) to detect
irregular and faulty pads

Adcis.ObjectSets.Measurements.Functions.Feret
ShapeMeasurements(osPads.Attributes["REGION"],
new FeretShapeMeasurementEnum[]
{ FeretShapeMeasurementEnum.fsmFeretMax },
Math.PI / 180);
// Extract irregular pads defined as pads with
a maximal Feret >= 14.5

Adcis.ObjectSets.ObjectSet osIrregular =
new Adcis.ObjectSets.ObjectSet();
Adcis.ObjectSets.Filtering.Functions.
Filter(osPads, osIrregular,
osPads.Attributes["REGION.Feret.Max"], 14.5,
1000);

// Change the fill style and color to display
the irregular pads

(osIrregular.Attributes["REGION"] as
Adcis.Drawing.IDrawable).StandardProperties.
FillStyle = Adcis.Drawing.FillStyleEnum.fsSolid;

(osIrregular.Attributes["REGION"] as
Adcis.Drawing.IDrawable).StandardProperties.
LineColor = Color.Green;
(osIrregular.Attributes["REGION"] as
Adcis.Drawing.IDrawable).StandardProperties.
FillColor = Color.Green;
```

```
// Display the irregular pads in the graphics
overlay

m_imageView.Data.Add(osIrregular.
Attributes["REGION"]);
richTextBoxComment.Text = "Irregular pads are
colored in green";
richTextBoxComment.Refresh();
```

9. .NET COMPONENTS (Dev and SDK only)

The Aphelion Dev and SDK products both include a large set of .Net components called Aphelion Imaging Toolkits (TKs). Each TK contains a set of related functions and the TKs collectively contain all Aphelion image processing functions Additional components are available to display images and ObjectSets before and after processing, and to interact with those objects in the Graphical User Interface.

- The Imaging Toolkits include all image and ObjectSet processing functions. There is an Aphelion .Net component for each group of Aphelion functions, such as *Filtering*, *Morphology*, and *EdgeDetection*.
- Other .Net components include the display of images and ObjectSets, interactive drawing in images, and object management.
- A complete list of the imaging groups can be found in the second-level dropdown menu for the **Process** entry on Dev's **Menu Bar**.

Note: The Aphelion 4.x online documentation includes a detailed description of each property and method of the .Net components.

APPENDICES

APPENDIX A APHELION DVD ARCHITECTURE

The Aphelion DVD root directory contains seven folders, one each for the Aphelion main software products called *AphelionDev*, *AphelionLab*, and *AphelionSDK*, one for the device drivers (*Dongle and acquisition device drivers*), one for the extensions, one for the documentation (*Help*), and an additional *Multimedia Demos* folder. Each product folder contains an installation program. The Aphelion Dev product has separate installation programs for 32-bit and 64-bit architectures. A different installation code is required to install each product.

For each product installed, the subfolders present on the disk are:

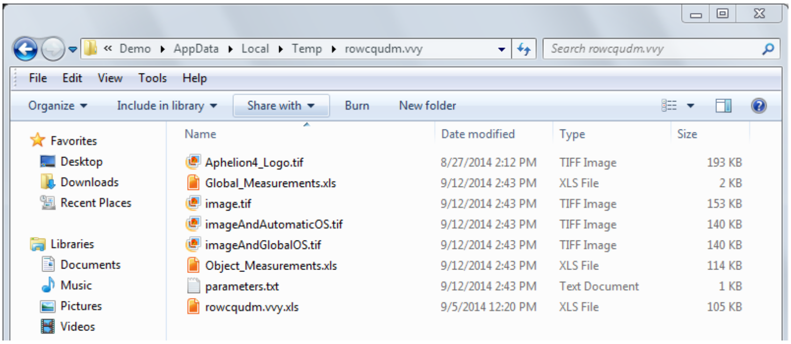
FILE TYPE	DESCRIPTION	DEV	LAB	SDK
bin	DLLs & Dev executable files	✓	✓	
Chart	Chart models	✓		
drivers	Acquisition device drivers		✓	
Examples	Programming examples	✓		✓
Examples 32-40	Version 3.2 examples ported to version 4.0	✓		
Help	Online documentation	✓	✓	✓
Images	Sample images for tutorials, testing, and developing macros and applications	✓	✓	✓
Include	Include files for Native C++ programming			✓
Lib	Libraries for 32-bit and 64-bit architectures			✓
Macros	Macro examples	✓		
Macros 32-40	Version 3.2 macros ported to version 4.0	✓		
Redistribution Files	Microsoft products required to install and run Aphelion software	✓	✓	✓

APPENDIX B REPORT GENERATION

B. 1. How to modify report-template

When generating a report, Aphelion looks for the different elements composing a report (images, objectsets, measurements, etc.) before creating a temporary folder containing the files needed to generate the report. This folder is located in *C:\Users\<Username>\Appdata \Local\Temp*.

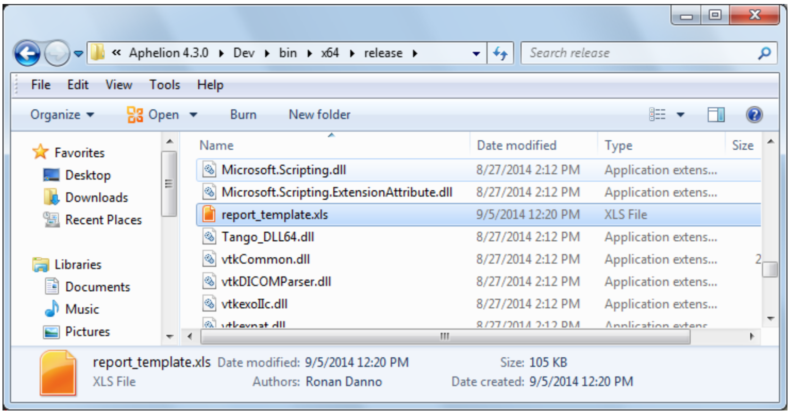
Aphelion uses Visual Basic macros to automatically generate the report. In the temporary folder, those macros are located in an xls file with a randomly generated name (e.g.: da0leqfm.jy5.xls). This file is a copy of the report_template.xls file and is only affects the report in the temporary folder.


« Demo ▸ AppData ▸ Local ▸ Temp ▸ rowcqudm.vvy
Search rowcqudm.vvy
File Edit View Tools Help
Organize ▾ Include in library ▾ Share with ▾ Burn New folder
Favorites
Desktop
Downloads
Recent Places
Libraries
Documents
Music
Pictures
Videos

Name	Date modified	Type	Size
Aphelion4_Logo.tif	8/27/2014 2:12 PM	TIFF Image	193 KB
Global_Measurements.xls	9/12/2014 2:43 PM	XLS File	2 KB
image.tif	9/12/2014 2:43 PM	TIFF Image	153 KB
imageAndAutomaticOS.tif	9/12/2014 2:43 PM	TIFF Image	140 KB
imageAndGlobalOS.tif	9/12/2014 2:43 PM	TIFF Image	140 KB
Object_Measurements.xls	9/12/2014 2:43 PM	XLS File	114 KB
parameters.txt	9/12/2014 2:43 PM	Text Document	1 KB
rowcqudm.vvy.xls	9/5/2014 12:20 PM	XLS File	105 KB

The generated xls file contains three different sheets, one for histogram values, one for object measurements, and one for the main report ready for printing.

To permanently customize the reports you are generating, modify the report_template.xls file located in the *<Aphelion Installation Path>\Aphelion 4.x.y\Dev\bin\<x64> or <win32>\release* folder.


« Aphelion 4.3.0 ▸ Dev ▸ bin ▸ x64 ▸ release ▸
Search release
File Edit View Tools Help
Organize Open Burn New folder
Favorites
Desktop
Downloads
Recent Places
Libraries
Documents
Music
Pictures

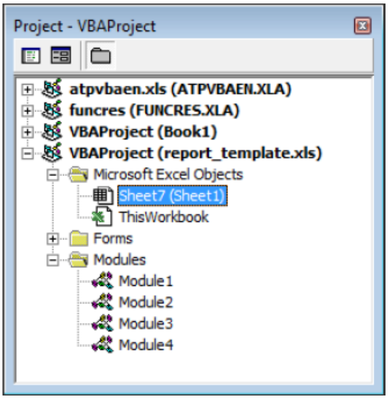
Name	Date modified	Type	Size
Microsoft.Scripting.dll	8/27/2014 2:12 PM	Application extens...	
Microsoft.Scripting.ExtensionAttribute.dll	8/27/2014 2:12 PM	Application extens...	
report_template.xls	9/5/2014 12:20 PM	XLS File	
Tango_DLL64.dll	8/27/2014 2:12 PM	Application extens...	
vtkCommon.dll	8/27/2014 2:12 PM	Application extens...	2
vtkDICOMParser.dll	8/27/2014 2:12 PM	Application extens...	
vtkexoIIc.dll	8/27/2014 2:12 PM	Application extens...	

report_template.xls Date modified: 9/5/2014 12:20 PM Size: 105 KB
XLS File Authors: Ronan Danno Date created: 9/5/2014 12:20 PM

Once the file is opened, select the **Tools→Macro→VB Editor** menu, or press **ALT + F11**.

Once the editor is opened, files that need to be edited are located in the *"module"* folder.

Modules to be modified are modules 1, 2 and 3. All the files share a similar architecture, but ***Module1*** is used for object measurement reports (classic report), ***Module2*** for interactive measurement reports (measurement of interactive shapes defined in the measurement section of Aphelion) and ***Module3*** is used for global measurement reports (measures computed on the whole image).



Major changes will require some Visual Basic coding knowledge. Click on the following link to access the online Visual Basic documentation: *http://msdn.microsoft.com/en-us/library/2x7h1hfk.aspx.*

B. 2. Examples of small changes

B. 2. a. Change the Aphelion logo

Open the module to be edited (it can either be modules 1, 2 and 3), and look for the comment “Add the Aphelion Logo”. Open ActiveSheet.Pictures.Insert(“CurrentFilename.tif”) and replace the current filename by the one you want to add:

```
' Add the Aphelion logo
'
    ActiveSheet.Pictures.Insert("Aphelion4_Logo.tif").Select
```

B. 2. b. Change the report title

Open the module to be edited (it can either be modules 1, 2 and 3), and look for the comment “Add the Aphelion Logo”. On the line ActiveCell.FormulaR1C1 = "CurrentTitle" replace the current title by the new one.

```
' Add the Aphelion logo
'
    ActiveSheet.Pictures.Insert("Aphelion4_Logo.tif").Select
    Range("C1").Select
    ActiveCell.FormulaR1C1 = "Object Measurement Report"
'
```

B. 2. c. Change displayed images

Open the module to be edited (it can either be modules 1, 2 and 3), and look for the comment “Insert images in the report”. Filenames, positions, and size can be modified, and images can be discarded.

```
' Insert images in the report
'
    ActiveSheet.Pictures.Insert("image.tif").Select
    SetImageSizeAndPosition ActiveSheet, 5, 1

    ActiveSheet.Pictures.Insert("imageAndAutomaticOS.tif").Select
    SetImageSizeAndPosition ActiveSheet, 5, 6
'
```

B. 2. d. Print report

Open the module to be edited (it can either be modules 1, 2 and 3). To automatically print the report into a PDF file, add the following code at the end of the macro. Use the method called PrintOut to generate a PDF file or just print to a printer. Look for the active printer available on the PC and add the code below accordingly:

```
'Print file
'
'Check if printer is ready
Application.ActivePrinter = "MyPDFPrinter"
    ActiveWindow.SelectedSheets.PrintOut ActivePrinter:="MyPDFPrinter"
```

B. 2. e. Add header/footer

Open the module to be edited (it can either be modules 1, 2 and 3), and look for the comment “Set the margins” at the end of the macro. Uncomment and change the needed lines, hand refer to the example below:

```
' Set the margins
'
    On Error Resume Next
    With ActiveSheet.PageSetup
'         .LeftHeader = ""
'         .CenterHeader = ""
'         .RightHeader = ""
'         .LeftFooter = ""
'         .CenterFooter = ""
'         .RightFooter = ""
```

```
' Set the margins
'
    On Error Resume Next
    With ActiveSheet.PageSetup
        .LeftHeader = "Company name"
        .CenterHeader = "Page &P of &N"
        .RightHeader = "Printed &D &T"
'         .LeftFooter = ""
'         .CenterFooter = ""
'         .RightFooter = ""
```

To add a header/footer on every worksheet, add a loop in the code like so:

```
' Set the margins
'
Dim ws As Worksheet
    For Each ws In ActiveWorkbook.Worksheets
        With ws.PageSetup
            .LeftHeader = "Company name"
            .CenterHeader = "Page &P of &N"
            .RightHeader = "Printed &D &T"
            .LeftFooter = "Path : " & ActiveWorkbook.Path
            .CenterFooter = "Workbook name &F"
            .RightFooter = "Sheet name &A"
        End With
    Next ws
    Set ws = Nothing
```

B. 2. f. Change histogram to scatter plot

Removing the histogram and replacing it with another diagram requires some Visual Basic knowledge. To avoid programming, the straightforward solution is to record a macro and add it to existing macros (e.g., modules 1, 2 and 3).

Start the recorder, selecting **Tools→Macro→Record new Macro** and enter the name of the new macro (e.g., “AddScattPlot”).

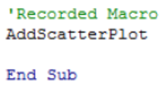
The basic concept is to edit the report sheet, substitute the 1D histogram by a 2D histogram or scatter plot, and at the same time record the modifications in a Visual Basic file containing the macro that automatically makes those changes.

Once the changes are completed, stop the recording process from the **Tools→Macro→Stop recording** menu.

Then open the Visual Basic Editor (**Alt + F11**) to locate the new file under the *“Modules”* folder. Move into the file to be modified (e.g., module 1, 2, or 3) and add the name of the recorded macro (“AddScattPlot”) at the end of the file (before the “End Sub” line).

Refer to appendix 1 for Visual Basic code that generates a scatter plot.

Note that any other macro that may be recorded by the user to perform a specific task can be added in the existing macro.



```
'Recorded Macro
AddScatterPlot

End Sub
```

B. 3. Example

The following code generates a scatter plot of the area versus the perimeter on 84 objects of an objectset.

```
Sub Macro10()
'
' Macro used to generate a scatter plot of the
Area versus the perimeter
'
  ActiveSheet.Shapes.AddChart.Select
  ActiveChart.ChartType = xlXYScatter
  ActiveChart.SetSourceData Source:=Range( _
      "'Object Measurements'!$B$2:$B$85,'Object
      Measurements'!$D$2:$D$85")
  ActiveChart.SetElement
      (msoElementChartTitleAboveChart)
  Selection.Caption = "Area vs Perimeter"
  ActiveChart.Legend.Select
  Selection.Delete
  ActiveChart.SetElement (msoElementPrimary
      CategoryAxisTitleAdjacentToAxis)
  Selection.Caption = "Area"
  ActiveChart.SetElement (msoElementPrimary
      ValueAxisTitleRotated)
  Selection.Caption = "Perimeter"
End Sub
```

APPENDIX C TROUBLESHOOTING

C. 1. Troubleshooting for the dongle

Verify that the dongle is actually connected to a USB port. If it is not, then go back to Step 4 of section *2.4.4 Installing the USB Dongle* of this guide. If it is connected, then unplug it, wait 10 seconds, and then reconnect the dongle. If the wizard Found New Hardware opens, go back to Step 5 of section *2.4.4 Installing the USB Dongle* of this guide. Otherwise, perform the following steps:

1. Right-click on the icon **My Computer** (for XP) or **Computer** (for Vista and 7) on your desktop or on the **Start** menu.
2. Select **Properties** from the drop-down menu.
3. For XP, select the **Hardware** tab, then the **Device Manager** button. For Vista and 7, select **Device Manager** Task (in upper left of the window).
4. A window opens showing a list of all devices connected to your computer. Double click on the **Universal Serial Bus controllers** so that all connected USB devices are displayed.
5. If a version of the *CBUSB device* other than *2.0* appears, it should be uninstalled. Do this by right-clicking on the incorrect version and selecting the option *Uninstall* on the menu that appears. Then install the *CBUSB 2.0* version.
6. After the uninstall is completed, remove the dongle from the USB port.
7. Wait ten seconds, then re-connect the dongle to the USB port.
8. After the Found New Hardware wizard opens, continue at step 5 of the section *2.4.4 Installing the USB Dongle* of this guide.
9. If the installation fails again, contact the Aphelion Support team (see section *2.3 Product Support*).

C. 2. Troubleshooting for the camera interface

C. 2. a. Checking the installation of the driver

To check that the CMU 1394 Digital Camera Device driver was installed properly, please follow the instructions below.

1. Check that the FireWire cable is properly connected to both the camera and the computer.
2. Open the **Windows Device Manager**.
3. Open the *Imaging devices* item and select the *CMU 1394 Digital Camera Device* driver you installed.

Note: If this driver is not listed, confirm that the computer was rebooted as described above. If the computer was rebooted and the driver does not appear in the *Imaging devices* list, then please attempt the driver installation again or contact your local systems support technician.

4. Right click the *CMU 1394 Digital Camera Device* driver to access its contextual menu, click the **Properties** item, and then click the **Driver** tab.
5. Check the version of the driver. It should be version *6.4.5.133* or higher. If the version of the driver is not a version cited above, please contact the Aphelion Support Team (see section *2.3 Product Support*).

C. 2. b. Checking camera operation

The stand-alone program ***1394CameraDemo.exe*** can be used to check camera operation. This program is located in the installation folder whose default path is:

<Aphelion Installation Path>\Dev\Drivers\IEEE1934\.

Use this program now to check that the camera and its drivers are properly installed.

After the 1394 Camera Demo interface is open, click **Camera** on the **Menu** bar. Then click **Check Link** to see if the camera is recognized. If it is not recognized, then please correct any of the following conditions that are not met.

1. Confirm that the camera is properly connected to the computer with a FireWire cable.
2. Confirm that the driver *CMU 1394 Digital Camera Device* is properly installed by checking its properties in the Windows Device Manager interface.
3. Confirm that the camera is compatible with the IIDC version 1.30 or version 1.31 specifications.

If these three conditions are all met and the camera still isn't recognized by the 1394 Camera Demo, or if you are unable to correct any of these conditions, then please contact the Aphelion Technical Support (see section *2.3 Product Support*), describing the problem you are experiencing as fully as possible, including which of the conditions are or are not being met.

APPENDIX D MAIN DIFFERENCES BETWEEN APHELION DEVELOPER 3.2 AND APHELION DEV 4.X

The main differences between Aphelion 4.x and its predecessor, Aphelion 3.2, are an all new graphical user interface (GUI), and a new architecture based on the Microsoft .Net Framework. The use of .Net components results in a more open and flexible product, benefiting from the new programming concepts. Note that all Aphelion functions have been totally rewritten in Visual Studio .Net and ported into both 32-bit and 64-bit environments.

The all new GUI was totally redesigned to offer a new *look and feel*, and a more structured operation paradigm based on tasks that make Aphelion 4.x especially easy to use. The tasks divide the image analysis process into its logical components, from image acquisition through report generation. The tasks are all accessible with one click from an always visible task bar.

Following the trend of “everything digital,” Aphelion 4.x now includes two image acquisition drivers: IEEE1394 and DirectShow. Aphelion 4.x does not support frame grabber products. Gigabit Ethernet cameras can be interfaced to Aphelion 4.x provided the manufacturer included support for DirectShow.

Since Aphelion 3.2 only supported 32-bit architectures, it could not process images larger than about 300MB. Aphelion Dev and SDK support both 32-bit and 64-bit architectures. With the 64-bit version, Aphelion 4.x imposes virtually no limit on the size of images that it can process.

Aphelion 3.2 only supported one macro language, BasicScript. While Aphelion 4.x continues support for BasicScript, it supports three additional macro languages: Python, Visual C# and Visual Basic. BasicScript macro recording is still provided, but the three new languages do not yet support macro recording. (Check Aphelion Readme files for information on macro recording support.)

Following is a brief description of the new features.

D. 1. Tasks

Aphelion Lab and Dev structure image processing and analysis work in terms of tasks. Each task corresponds to a coherent subset of the work steps necessary in performing image analysis. Lab and Dev share the same tasks, but Dev adds an additional task, **Developer**, that greatly expands the number and capability of the image processing and analysis functions and adds tools for developing deployable applications. We recommend that new users and users of Lab follow the sequence of tasks as laid out in Aphelion's GUI. With the Dev product, the complexity and goals of the image analysis project will dictate if and when the expanded capabilities of the **Developer** task are needed.

D. 2. Administrator rights

Aphelion 4.x provides a password protected mode for administrator users who have full control of all system configuration settings. The non-administrator user typically captures images using a system configured by an administrator and processes the images using preconfigured software procedures. All system calibration settings, camera selection, and other advanced parameters are typically not made available in the non-administrator user mode.

D. 3. Acquisition Task

In Aphelion 3.2, the image acquisition functions were spread out in the menus. In Aphelion 4.x, all image acquisition functions are together in a single task. **Calibration** tools and associated resolutions for imaging devices are now easily accessed in the GUI.

D. 4. Object Extraction Task

Many images can be easily segmented (e.g., using a threshold). For such images, the **Object Extraction** task will usually be sufficient. However, for more complex images the added capabilities included in Aphelion Dev's **Developer** task may be necessary. Note that an ObjectSet created with the **Object Extraction** task can be input for an ObjectSet function in the **Developer** task.

D. 5. Object Editing Task

This task enables a user to alter an ObjectSet generated by either the **Object Extraction** task or **Developer** task. Alteration of an ObjectSet includes using the mouse to draw new objects or modify existing objects, as well as delete objects.

D. 6. Measurement Task

The number of object measurements has been dramatically increased in Aphelion 4.x. The current release of Aphelion 4.x provides its users with about 50 measurements functions. The set of interactive measurement tools available in version 3.2 has also been enriched in version 4.x, with many, new tools such as angle measurement and caliper.

D. 7. Report Generation Task

This task provides a new capability that was not available in Aphelion 3.2. It generates analysis reports that use a Microsoft Excel template. The content and format of the report can be configured by the user who has some Visual Basic for Applications knowledge. This task also includes functions to copy images and ObjectSets to the Windows clipboard, from where a user can easily import them into other applications (e.g., Word, Excel).

D. 8. Developer Task

D. 8. a. Function Selection

In Aphelion 3.2, all functions were selected in the Operator Dialog Box by clicking on a unique icon. Aphelion 4.x makes all functions accessible from the **Process** menu on the **Menu Bar**. Version 3.2's small icons containing images no longer exist. With version 4.x, images entered as either input or output parameters of a function are automatically displayed in the Visualization window.

Most of the functions available in Aphelion 3.2 are still available in Aphelion 4.x. While a few uncommon functions have not been ported to version 4.x, many new functions have been added.

D. 8. b. Macro Languages

As mentioned in many places in this User Guide, four macro languages are now supported: BasicScript, Python, Visual C#, and Visual Basic. Thus Aphelion users of the previous version can still develop and use BasicScript macros. However, new users may opt for a more modern macro language.

D. 8. c. BasicScript Command window

The History tab of the Information/History/Message window in version 3.2 of Aphelion is no longer available in version 4.x. It has been replaced by the BasicScript Command window (see *Immediate and Output Windows*). A BasicScript command can be written, edited, and run from this window.

D. 9. Aphelion 4.x User Interface

The MDI (Multiple Document Interface) available in Aphelion version 3.2 has been replaced by a fixed (i.e., non-floating) window (Visualization window) in which one or four images can be displayed. However, while all other Aphelion 4.x windows have a default, fixed location in the GUI, any of them can be easily floated as the user chooses, and just as easily re-fixed in the GUI, using simple mouse clicks. Floating windows can be dragged to any location in the Windows interface. All windows can be easily resized in the standard way for Windows applications.

D. 10. Image and ObjectSet Galleries

Two new features available in Aphelion 4.x are the Image Gallery and the ObjectSet Gallery. Every open image and ObjectSet, whether loaded from disk or created in the GUI, is displayed in its respective gallery as a thumbnail, labeled with its file name. Double-clicking on a thumbnail causes the corresponding image or ObjectSet to be displayed in the Visualization window. This can also be done through a thumbnail's contextual menu.

D. 11. Color space

The Color Space property of a color image could only be changed by calling the AphImgColorConvert image processing function in version 3.2 of Aphelion.

D. 11. a. Default Color Space

The default color space for an image is Blue-Green-Red (BGR) in version 4.x while it was RGB in version 3.2. This change have been made to improve the refresh of images, especially when images are very large.

D. 11. b. Color Space Conversion

Since the color space is one of the image properties, it has been added to the image properties available in the Image Properties window (righ-mouse click on the image thumbnail in the Image Gallery tab). This property can be edited by selecting a color space from the color space list that opens when clicking on the Color Space button in the Image Properties window.

Note: The color space change does not modify the image display in version 4.x. It changed the image display in version 3.2.

The color space of a color image can also be modified in a macro:

BasicScript syntax in version 3.2:

```
AphImgColorConvert AphImg(("Image 1"),
AphImg(("Image 1"), 1
```

BasicScript syntax in version 4.x:

```
Aph4ImgSetColorSpace AphImg("Image 1"), SpaceName
```

D. 11. c. Accessing the band of a color image

The display of a single band of a mult-band image has been improved. Clicking on ◀▶ displays the previous or next band of a multi-band image (i.e. color, frequency, complex, and edge images).

A single-band image can also be extracted from a multi-band image by calling the *Images.Utility.ExtractBand* function in version 4.x instead of the *AphImgSplitBands* function in version 3.2.

A single-band image can also be copied as one of the bands of a multi-band image by calling the *Images.Utility.CopyBand* function in version 4.x instead of the *AphImgJoinBands* function in version 3.2.

D. 12. Running a macro when starting Aphelion Dev

In version 3.2 of Aphelion Dev, the Aphelion.apm macro was automatically run when launching the software.

To automatically run a macro when launching Aphelion Dev 4.x, add its name in the list available in the Startup tab of the **Tools→Options** menu.

D. 13. Structuring Element definition

Predefined structuring elements used by functions in the Image.Morphology group are defined in a different way than in version 3.2. The size of the structuring element now gives the “radius” of the ball while it defined the width in the previous 3.2 version. The relationship between the two sizes is the following:

$S_{v4.x} = (S_{v3.2} – 1) / 2$

APPENDIX E DIFFERENCES IN PROGRAMMING SYNTAX BETWEEN APHELION 3.2 AND APHELION 4.X (Dev version only)

E. 1. Programming syntax

Aphelion 4.x includes namespaces for functions libraries.

In version 3.2 of Aphelion, a function had the following syntax in BasicScript: *AphImgFunction* or *AphObjFunction*. The prefix *Aph* is no longer present in version 4.x, except in BasicScript (to maintain backwards compatibility). Aphelion 4.x syntax is now:

Adcis.Images.Group.SubGroup.FunctionName

Adcis.ObjectSets.Group.SubGroup.FunctionName

For example, erosion functions for images and ObjectSets are defined as:

Adcis.Images.Morphology.Basic.Erode

Adcis.ObjectSets.Bitmaps.Morphology.Erode

Refer to section *5 List of functions (Dev and SDK only)* for the group and subgroup of each Aphelion function.

E. 2. BasicScript Editor

Most of the Aphelion 3.2 features for developing macros, such as:

- Execute code step by step;
- Watch a variable; and
- Set a break point

are still available in Aphelion 4.x.

E. 3. BasicScript syntax

The text below gives the major differences between the two Aphelion versions with respect to programming in BasicScript.

Note: BasicScript functions starting with the Aph4 prefix are functions that are either new in version 4.x, not available in version 3.2 (e.g., *Aph4ImgClip*, *Aph4ObjOtsuThreshold*, *Aph4MissingParameter*), or an alternate version of functions that were available in version 3.2 (e.g., Aph4ImgErode).

Tip: There are new functions available in Aphelion 4.x (e.g., *Otsu Threshold*). Since these functions did not exist in version 3.2, the associated BasicScript function did not exist in the previous version. All new functions in Aphelion 4.x can be called in BasicScript using the following syntax:

Aph4<name of the Function> (e.g., Aph4ImgOtsuThreshold)

E. 3. a. Function to let the user enter function argument values during the macro execution

The function *Aph4MissingParameter* has been introduced to convert a BasicScript macro into an interactive macro in which some parameters can be omitted. In this case, a window pops up to let the user define the missing parameter(s)

E. 3. b. Function to retrieve argument values set during the macro execution

The *AphOperatorArgumentValue* function is replaced by the *Aph4OperatorArgumentValue* function. Below are the differences:

Aph4OperatorArgumentValue does not take the name of the function as the parameter. It is only possible to retrieve parameter values of the last executed version.

The returned value in version 4.x is a .NET object. It was an Active X object in version 3.2 defined as a simple type object (e.g., long, double, string). To access properties of the .NET object, use the *Aph4GetPropertyValue* and *Aph4ToBuiltinType* functions.

Example:

```
Binary = AphImgNew("Image 2")
AphImgShow AphImg("Image 1")
    ' Make sure the thresholded image is visible
AphImgThreshold AphImg("Image 1"), Binary,
Aph4MissingParameter()
AphImgShow Binary
threshold = Aph4OperatorArgumentValue(2)
minimum = Aph4ToBuiltinType(Aph4GetPropertyValue(
Aph4GetPropertyValue(threshold, "Item", 0),
"Minimum"))
    ' Item is the name of the 'this[band]' indexer
of the .NET IntervalD class
maximum = Aph4ToBuiltinType(Aph4GetPropertyValue(
Aph4GetPropertyValue(threshold, "Item", 0),
"Maximum"))
MsgBox "Minimum = " & minimum & ", Maximum = " &
maximum
```

E. 3. c. ObjectSet Attributes

In version 3.2 of Aphelion, some attribute names had an "_" such as *PIXEL_COUNT*. The underscore character has been removed in version 4.x. Although an attribute name with an underscore is valid in version 4.x, we recommend not using the underscore in new macros you develop in version 4.x as these names will disappear in future versions of Aphelion Dev.

Note: By default, in Aphelion 4.x, ObjectSet names are case sensitive.

Spatial attribute name of extracted objects

The spatial attribute default name of an ObjectSet generated by an object extraction applied to an image is “BITMAP” in version 4.x of Aphelion Dev while it was “REGION” in version 3.2. The default name can be altered by editing the ObjectSets > DefaultAttribute in the Advanced tab in the **Tools→Options** menu.

Sub-attribute access

Below is the new BasicScript syntax to access Aphelion version 3.2 pseudo-attributes that are sub-attributes in version 4.x.

Syntax to get the Bitmap object position.

Version 3.2 syntax:

```
AphObjGetAttributeD(AphObj("OS"), index,
"BITMAP.EXTENTS.LL.X")
AphObjGetAttributeD(AphObj("OS"), index,
"BITMAP.EXTENTS.UR.X")
```

Version 4.x syntax:

```
AphObjGetAttributeD(AphObj("OS"), index,
"BITMAP.X1")
AphObjGetAttributeD(AphObj("OS"), index,
"BITMAP.X2")
```

Syntax to get the Line object position.

Version 3.2 syntax

```
AphObjGetAttributeD(AphObj("OS"), index,
"LINE.P1.X")
```

Version 4.x syntax

```
AphObjGetAttributeD(AphObj("OS"), index,
"LINE.X1")
```

Syntax to get overlaping objects.

Version 3.2 syntax

```
AphObjGetAttributeD(AphObj("OS"), index,
"NEIGHBORS.NUMENTRIES")
```

Version 4.x syntax

```
AphObjGetAttributeD(AphObj("OS"), index,
"NEIGHBORS.Length")
```

Syntax to get sub-attributes from a vector attribute.

To get the mean computed on each band of a color image:

```
AphObjGetAttributeD(AphObj("OS"), index,
"Mean[0]")
AphObjGetAttributeD(AphObj("OS"), index,
"Mean[1]")
```

To get the index of each object overlaping the object from the ObjectSet named OS which the index is equal to the index variable:

```
AphObjGetAttributeD(AphObj("OS"), index,
"NEIGHBORS_Overlap[2]")
```

In addition, if the AphObjGetAttributeD function applied to a vector attribute returns the first vector element:

```
AphObjGetAttributeD(AphObj("OS"), index, "Mean")
```

returns the same value as:

```
AphObjGetAttributeD(AphObj("OS"), index, "Mean[0]")
```

E. 3. d. Histogram value access

The value storage of the histogram in a BasicScript array is different between versions 3.2 and 4. For each band, there is the exact number of values equal to the number of the band class. This number may vary between bands.

E. 3. e. Image list access

The *AphImageList* function is now created by '*MacroAddToLib'. It has to be declared before it is used. The associated parameter is a Variant array. Note it was a Long array in version 3.2.

E. 3. f. Value Output

The function *AphInfoWrite* outputs a value to the tabbed Output window in window B.

```
Dim volume() As Double
AphImgVolume im, volume
AphInfoWrite "volume: " & volume(0)
```

E. 3. g. List of non-supported functions in Aphelion 4.x

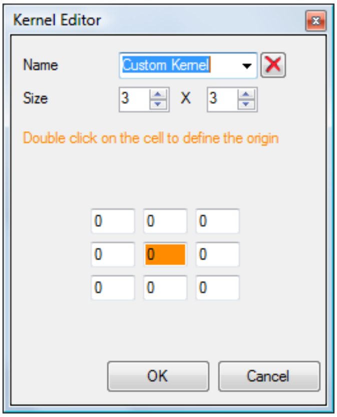
Most of the BasicScript functions available in version 3.2 that were specific to control the user interface no longer exist since the version 4.x GUI has been totally redesigned.

E. 4. Change of Syntax

Functions such as *AphImg*, *AphImgNew*, *AphObj*, *AphObjNew*, *AphNamedKernel*, and *AphNamedSElement* that are returning an object instance do not return an object of type Long. They are now returning an object of type Variant.

APPENDIX F KERNEL AND STRUCTURING ELEMENT EDITOR (Dev version only)

Kernel Editor – The Kernel Editor window is a graphic tool for easily specifying a kernel to be used by the convolution function. First, specify the name of the kernel in the *Name* dialog box. Next, in the *Size* boxes, specify the kernel's dimensions, and then enter kernel values from the keyboard. Use the **Tab key/Ctrl+Tab key** to move forward/backward through the parameter values. Double-clicking on a matrix cell will define this cell as the kernel origin. When done, click on the **OK** button. The name of the Kernel can be specified by the user; its default name is *Custom Kernel*. Clicking ✕ deletes the kernel.


Kernel Editor

Name: Custom Kernel

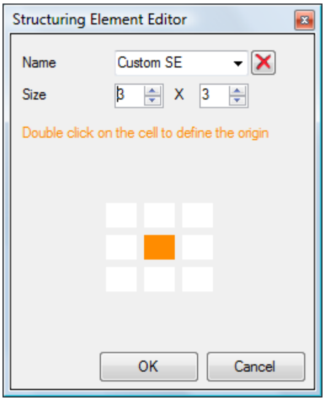
Size: 3 X 3

Double click on the cell to define the origin

0	0	0
0	0	0
0	0	0

Structuring Element Editor – The Structuring Element Editor window is a graphic tool for easily specifying a structuring element to be used by a basic morphology function (e.g., erosion, dilation). First, specify the name of the structuring element in the *Name* dialog box. Next, specify the X and Y sizes, and then enter structuring values left-clicking inside each individual cell (cell values can be ✕ for set and blank for not set). Double-clicking on one cell will define this cell as the

Structuring Element origin. When done, click on the **OK** button. By default, the name of the Structuring Element is *Custom SE*. Clicking ❌ deletes the Structuring Element.



Double click on the cell to define the origin

Tip: Select the overload that does not use predefined Kernels or Structuring Elements to be able to define a custom Kernel or Structuring Element.

APPENDIX G C# PROGRAMMING LANGUAGE (Dev version only)

G. 1. Developing an Aphelion macro in C#

G. 1. a. Introduction

Writing an Aphelion C# macro is almost similar to writing a stand-alone C# executable, referencing Aphelion managed DLLs:

- ✓ Standard C# language is used
- ✓ The macro is compiled before being run

Differences are:

- ✓ Macros do not need to call `Adcis.Library.Initialize()` and `Adcis.Library.Terminate()`
- ✓ Macros have access to Aphelion internal memory. In particular, the Aphelion4.Application static class gives access to its current state. The list below gives a few of its useful members:
  - Aphelion4.Application.Images: collection of images displayed in the Image Gallery
  - Aphelion4.Application.ObjectSets: collection of ObjectSets displayed in the ObjectSet Gallery
  - Aphelion4.Application.ImageView(Image): the ImageView (control handling the display of images, ObjectSets and user-drawn shapes) containing a specific image
  - Aphelion4.Application.MainForm.CurrentImageView: the active ImageView
  - Aphelion4.Application.MainForm.CurrentImageView.Data[0]: the image (or video) in the active ImageView

For example, to add an image to the Aphelion GUI, execute the following in a C# macro:

```
IImage image = …;
Aphelion4.Application.Images.Add(image);
```

This automatically adds the image to the Image Gallery and displays it in the Visualization Window.

G. 1. b. Referencing DLLs in a C# macro

To call functions from external DLLs in a C# macro, the external DLLs must be referenced in the "`APHELION_HEADER`" section at the beginning of the macro, using the `reference` keyword. For example, to reference a new DLL called MyFunctions.dll, add the following line:

```
reference "MyFunctions.dll"
```

Between

```
#if APHELION_HEADER
```

And

```
#endif
```

Note: The most common .NET and Aphelion DLLs are already referenced by default when a new C# macro is created in the C# macro window.

G. 1. c. Developing a C# macro using an external IDE

Developers who are used to developing in C# may notice that the Aphelion integrated C# macro window lacks some advanced functionalities, such as automatic completion (provided by Microsoft® Visual Studio® Intellisense, for example).

However, it is possible to have the benefit of such external Integrated Development Environments (IDEs) while developing an Aphelion macro. To do so:

- ✓ Create a new project
- ✓ Add the necessary `Adcis.*.dll` references (the DLL files loaded by the macro are the ones specified by the "`reference`" lines in the section starting by `#if APHELION_HEADER`).
- ✓ Add `AphelionDev.exe` as a reference
- ✓ Add the macro file to your project (tip: add it as a link to avoid duplicating the macro file).
- ✓ Edit the macro file in the IDE, without removing the `#if APHELION_HEADER` … `#endif` section
- ✓ When editing is completed, save it
- ✓ Come back to Aphelion and click in the C# macro window: Aphelion automatically detects the file alterations and asks whether to reload it. Answer Yes.

G. 2. Adding a custom function to the Aphelion GUI in C#

G. 2. a. General conditions to be met by the macro function

Macro functions can be added to the list of Aphelion functions, and therefore may be called from the GUI, either from the Process menu of by typing its name in the Functions panel (in Developer task only). To achieve this, the following conditions must be met:

- ✓ The function must be a member of a public class;
- ✓ The class owning the function must have a `Adcis.System.Functions.Group` attribute, defining the sub-menu in which it will appear in the `Process` main menu;
- ✓ The function must be declared `public` and `static`;
- ✓ The function must have no return value (type `void`);

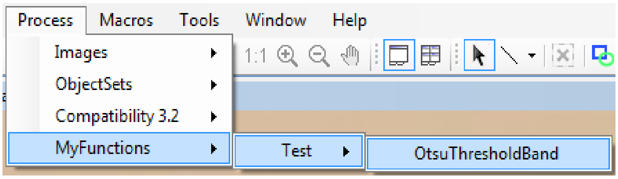
- ✓ The function input parameters must all have one of the types listed below (see *Overloads*);
- ✓ The macro must be compiled without error in Aphelion C# macro window.

Note: If the function misses any of these conditions, it will not be added to the Aphelion functions list. However, no error message will be output.

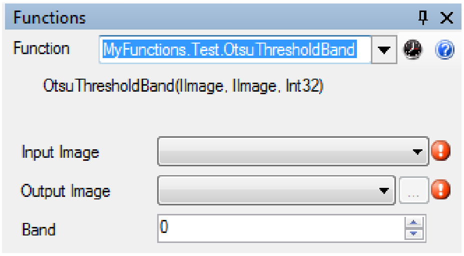
For example, let us look at the following function:

```
namespace MyNamespace
{
   [Adcis.System.Functions.Group("MyFunctions.
    Test")]
   public class Functions
   {
      static public void OtsuThresholdBand(
       IImage inIm, IImage outIm, int band)
      {
         Image bandIm = new Image();
         Adcis.Images.Utility.Functions.
          ExtractBand(inIm, bandIm, band);
         Adcis.Images.Segmentation.Functions.
          OtsuThreshold(bandIm, outIm);
      }
   }
}
```

Once compiled in Aphelion C# macro window, it will be available from the GUI through the menu: **Process→MyFunction →Test→OtsuThresholdBand**, as follows:



It will also be available from the Functions panel, with the name MyFunctions.Test. OtsuThresholdBand (see below for the variable naming conventions):



MyFunctions.Test.OtsuThresholdBand

OtsuThresholdBand(IImage, IImage, Int32)

Input Image

Output Image

Band 0

Note: Several Aphelion custom functions can be defined in a single macro file.

G. 2. b. Overloads

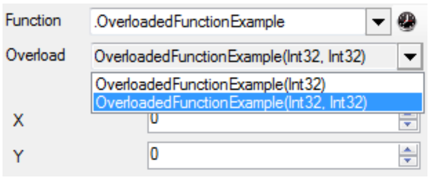
Inside a class, it is possible to have several methods with the same name. They are called overloaded methods, or overloads. When a function has several overloads, the user can select one in the list under the function name.

For example, in the following example:

```
namespace MyNamespace
{
   [Adcis.System.Functions.Group(
    "MyFunctions.Test")]
   public class Functions2
   {
      static public void
       OverloadedFunctionExample(int x)
      {
         // ...
      }
      static public void
       OverloadedFunctionExample(int x, int y)
      {
         // ...
```

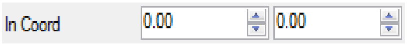
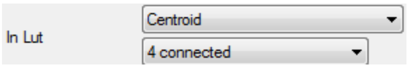
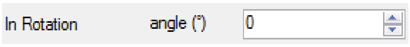
```
            }
        }
}
```

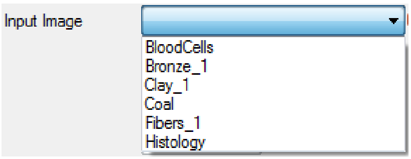
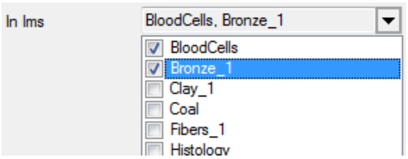
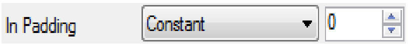
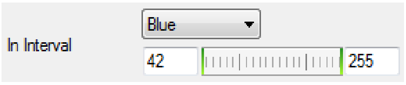
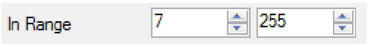
Both overloads will be available in the Functions panel as follows:

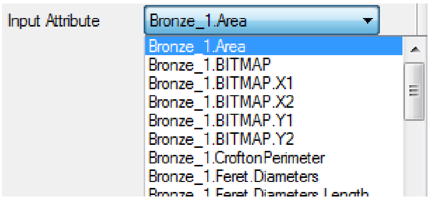
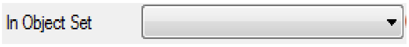
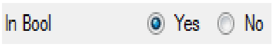
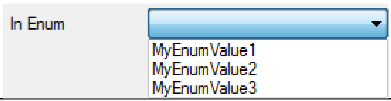
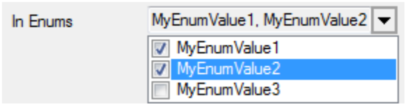
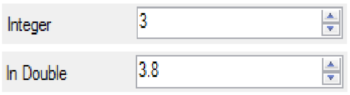

Function
OverloadedFunctionExample
Overload
OverloadedFunctionExample(Int32, Int32)
OverloadedFunctionExample(Int32)
OverloadedFunctionExample(Int32, Int32)
X
0
Y
0

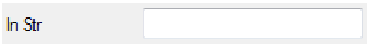
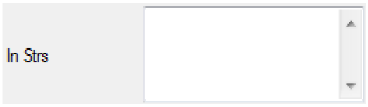
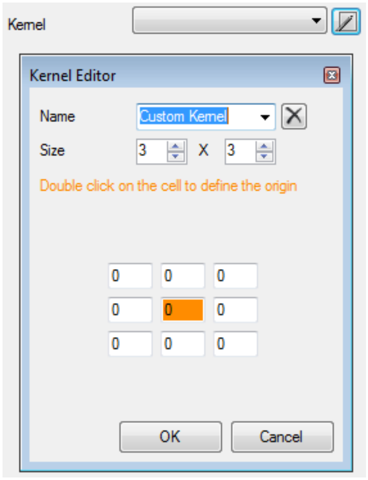
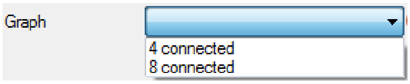
G. 2. c. Function input parameters

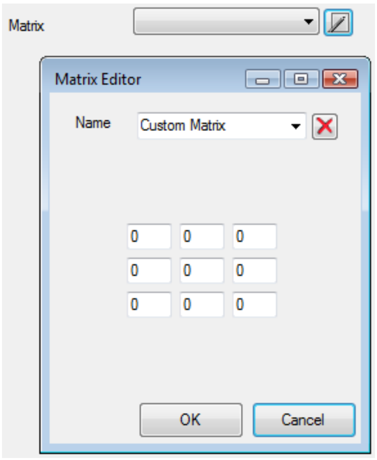
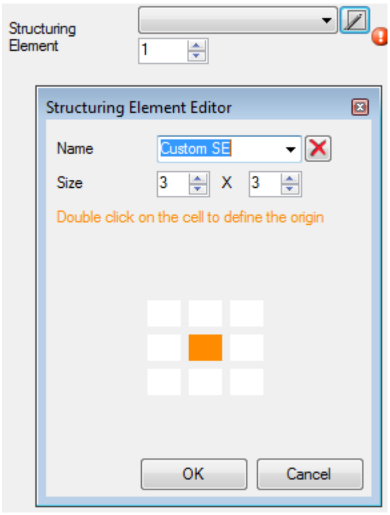
The function input parameters must belong to the following list. When the function is selected in the Aphelion Functions panel, the display is then automatically adapted to let the user specify an object of the required type. Depending on the argument types, different attributes can be used to customize the display (see *Additional attributes to customize the display of a parameter in the functions panel*).

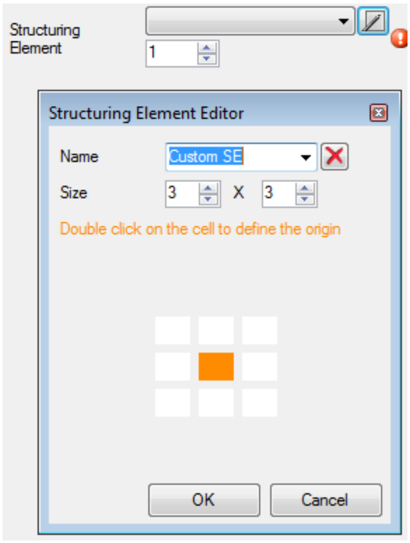
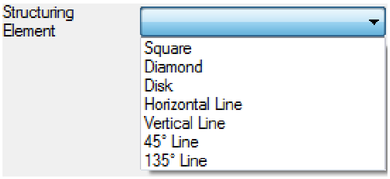
Function argument type	**Definition and Display**
`Adcis.Geometry.ICoordinatesD`	Coordinates of a point in an image 
`Adcis.Geometry.INgbLUT`	Neighborhood configurations used by some Thinning and Skeletonization functions 
`Adcis.Geometry.IRotation`	An angle, displayed in degrees, with value retrieved in radians 

Function argument type	**Definition and Display**
`Adcis.Images.IImage`	Image  Input Image: BloodCells, Bronze_1, Clay_1, Coal, Fibers_1, Histology
`Adcis.Images.IImageCollection`	List of images  In Ims: BloodCells, Bronze_1 — ✓ BloodCells, ✓ Bronze_1, Clay_1, Coal, Fibers_1, Histology
`Adcis.Images.Padding`	Padding value used by some transform functions (e.g., Rotate)  In Padding: Constant, 0
`Adcis.Math.IIntervalD`	Multi-dimensional interval of floating-point values  In Interval: Blue, 42, 255
`Adcis.Math.IRangeD`	One-dimensional interval of floating-point values  In Range: 7, 255

Function argument type	**Definition and Display**
`Adcis.ObjectSets.IAttribute`	Attribute of an ObjectSet  Input Attribute: Bronze_1.Area; Bronze_1.Area, Bronze_1.BITMAP, Bronze_1.BITMAP.X1, Bronze_1.BITMAP.X2, Bronze_1.BITMAP.Y1, Bronze_1.BITMAP.Y2, Bronze_1.CroftonPerimeter, Bronze_1.Feret.Diameters
`Adcis.ObjectSets.IObjectSet`	ObjectSet  In Object Set
`Bool`	Boolean value (true/false)  In Bool [✓ Yes] No
`enum` type (e.g., `Adcis.System.NumericTypeEnum)`	A C# enumeration, defined by the `enum` keyword  In Enum: MyEnumValue1, MyEnumValue2, MyEnumValue3
`Enum type array (e.g., Adcis.System.NumericTypeEnum[])`	A C# array of an enumeration type  In Enums: MyEnumValue1, MyEnumValue2; [✓ MyEnumValue1] [✓ MyEnumValue2] MyEnumValue3
System primitive types: `Byte, SByte, Int16, UInt16, Int32, UInt32, Int64, UInt64, Char, Double, Single`	.NET built-in integer and floating-point types  Integer: 3; In Double: 3.8

Function argument type	Definition and Display
`String`	A character string  In Str
`string[]`	An array of character strings  In Strs
`Adcis.Geometry.IKernel`	A user-defined kernel, used in convolution functions (see *Convolve function* help)  Kernel Kernel Editor Name: Custom Kernel Size: 3 X 3 Double click on the cell to define the origin 0 0 0 0 0 0 0 0 0 OK Cancel
`Adcis.Geometry.PredefinedNgbGraphEnum`	A 2D or 3D predefined connexity graph  Graph 4 connected 8 connected

Function argument type	**Definition and Display**
`Adcis.Math .IMatrixD`	A matrix (currently limited to 3x3 matrices) 
`Adcis.Geometry .IStructuring Element`	A 2D user-defined structuring element  Structuring Element Editor Double click on the cell to define the origin

Function argument type	Definition and Display
`Adcis.Geometry.IStructuringElement`	A 2D user-defined structuring element  Structuring Element Editor Double click on the cell to define the origin
`Adcis.Geometry.PredefinedStructuringElementEnum`	A 2D or 3D predefined structuring element 

Note: the declared input parameters must be **exactly** the ones described above. For example, the following functions will be recognized and added to Aphelion functions list:

```
static public void
Function1(Adcis.Images.IImage inIm) {} // OK
```

```
using Adcis.Images;
static public void Function2(IImage inIm) {}
// OK
```

But the following will not:

```
// Function not recognized: Image is not the
expected IImage type
static public void
 Function3(Adcis.Images.Image inIm) {}
```

G. 2. d. Function output parameters

Basic types output

Function results that need to be output must be passed as output parameters. For most output parameter types, this can be done using the standard C# `out` keyword. In the Functions panel, the parameter display is automatically adapted so that the user cannot enter a value for these parameters.

For example, let us look at the following function:

```
namespace MyNamespace
{
   [Adcis.System.Functions.Group(
    "MyFunctions.Test")]
   public class Functions
   {
     static public void
      OutputParametersExample(IImage image,
      out long width, out long height)
     {
       width = image.Size.X.Value;
       height = image.Size.Y.Value;
     }
   }
}
```

When the user selects it in the Functions panel, the width and height parameters are grayed out, as follows. Their value is then automatically displayed when the function is executed.

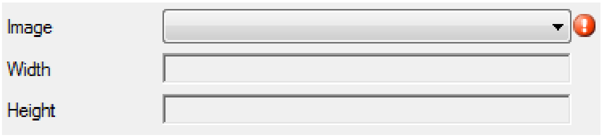


Image and ObjectSet output

When a function outputs an image (Adcis.Images.IImage parameter type) or ObjectSet (Adcis.ObjectSets.ObjectSet parameter type), the `out` keyword is not suitable. The recommended method consists in displaying the list of existing images or ObjectSets (as for input parameters), adding a “New entry” to them.

Two methods can be used to achieve this:

- Have parameter names start with “out” (e.g., “outImage”, “outOS”): see *Implicit attribute definition*);

or

- Add a Adcis.Images.NewImageAllowed attribute to allow the creation of a new image: see *Adcis.Images.NewImageAllowed attribute*;
- Add a Adcis.ObjectSets.NewObjectSetAllowed attribute to allow the creation of a new ObjectSet: see
- *Adcis.ObjectSets.NewObjectSetAllowed* attribute.

G. 2. e. Conventions used for arguments display

Parameter names are displayed in the Functions panel, with the following alterations:

Case decomposition

The first letter of the argument name is automatically converted to upper case. Other upper case letters are automatically preceded by a space. For example, an argument named “`inputArgument`” is displayed “Input Argument”.

Automatic translation

A few argument names are automatically translated to a more complete name in the Functions panel. The most common translations are:

Parameter name	**Display in the Functions panel (in English culture)**
inIm	Input Image
inIm1	Input Image 1
inIm2	Input Image 2
inReferenceIm	Reference Image
seedIm	Seed Image
outIm	Output Image
im	Image
maskIm	Mask Image
inAt	Input Attribute
inAttribute	Input Attribute
outAt	Output Attribute
outAttribute	Output Attribute
inOS	Input ObjectSet
inOS1	Input ObjectSet 1
inOS2	Input ObjectSet 2
ioOS	Input/Output ObjectSet
outOS	Output ObjectSet
p1	Point 1

p2	Point 2
se	Structuring Element
nt	Numeric Type

Note: Argument type is not taken into account for these automatic replacements.

Automatic translation result depends on the current culture.

Implicit attribute definition

For simplification purposes, a few rules are applied, without requiring an explicit attribute (see *Additional attributes to customize the display of a parameter in the functions panel*) definition:

Arguments of type Adcis.ObjectSets.IImage whose name begins with “out” are implicitly considered as having an Adcis.Images.NewImageAllowed attribute, hence displaying a “New Image” item at the beginning of the Image list.

Arguments of type Adcis.ObjectSets.IObjectSet whose name begins with “out” are implicitly considered as having an Adcis.ObjectSets.NewObjectSetAllowed attribute, hence displaying a “New ObjectSet” item at the beginning of the ObjectSet list.

Arguments of types `Int32, UInt64, Int64` are implicitly considered as having an Adcis.System.Number attribute with property `DecimalPlaces=0`, hence no decimals can be entered by default for arguments of these types.

G. 2. f. Additional attributes to customize the display of a parameter in the functions panel

Adcis.System.Functions.ArgumentEditor attribute

Applies to: Any argument

Constructor arguments:

Type	Optional	Value	Comment
Int		Zero-based index of a function argument	
String		See example below	

This attribute is mainly used to specify how an argument should be displayed and edited by the user in the Functions panel. It is particularly suitable for user-defined structures.

Example:

```
public class MyClass
{
   private int m_count;
   private string m_name;

   public int Count
   {
      get { return m_count; }
      set { m_count = value; }
   }

   public string Name
   {
      get { return m_name; }
      set { m_name = value; }
   }

   public override String ToString() { return
    String.Format("Count={0} Name={1}",
    m_count, m_name); }
}

[ArgumentEditor(0, "Adcis.System.
 PropertyGridArgumentEditor, Adcis.UI.1")]
```

```
[Adcis.System.Functions.DefaultValue(0,
 typeof(MyClass), null)]
static public void ArgumentEditorExample(
 MyClass myClassInstance)
{
   // ...
}
```

Adcis.Geometry.AxesProvider attribute

Applies to: `Adcis.Geometry.PredefinedNgbGraphEnum, Adcis.Geometry.PredefinedStructuringElementEnum, Adcis.Geometry.IStructuringElement, Adcis.Geometry.IKernel, Adcis.Math.IMatrixD`

Constructor arguments:

Type	Optional	Value	Comment
Int		Zero-based index of a function argument	
Int		Zero-based index of a function IImage, `IAttribute` or `IObjectSet` argument defining the axes (generally either XY or XYZ) along which the element is defined	

Example:

```
[Adcis.Geometry.AxesProvider(1, 0)]
static public void
 PredefinedStructuringElementExample(IImage
 image, Adcis.Geometry.Predefined
 StructuringElementEnum structuringElement)
```

```
{
    // ...
}
```

Adcis.ObjectSets.Attribute attribute

Applies to: `Adcis.ObjectSets.IAttribute`

Constructor arguments:

Type	Optional	Value	Comment
Int		Zero-based index of the function argument to which this attribute applies	

Optional properties:

Name	Type	Value	Comment
`DataType`	`string`	Full name of a data type	When set, only attributes of the selected data type are listed
`DimensionProviderIndex`	`Int`	Zero-based index of an IImage or IAttribute function argument that determine the dimension (2D or 3D) of the attribute to be selected or created.	

`OwnerObjectSetIndex`	`Int`	Zero-based index of an IObjectSet function argument from which the attribute is selected	
`Scalar`	`bool`	true, false	If true, only scalar attributes are listed
`Spatial`	`bool`	true, false	If true, only spatial (i.e. displayable) attributes are listed
`TypeProviderIndex`	`Int`	Zero-based index of an IImage or IAttribute function argument that determine the data type (e.g., Float, IChainF…) of the attribute to be selected or created.	

Example:

```
[Adcis.ObjectSets.Attribute(1,
 OwnerObjectSetIndex = 0, Spatial = true)]
static public void AttributeExample(
 Adcis.ObjectSets.IObjectSet inObjectSet,
 Adcis.ObjectSets.IAttribute inAttribute)
{
   // ...
}
```

Adcis.Geometry.Coordinates attribute

Applies to: `Adcis.Geometry.ICoordinatesD`

Constructor arguments:

Type	Optional	Value	Comment
Int		Zero-based index of the function argument to which this attribute applies	

Optional properties:

Name	Type	Value	Comment
`AxesProvider ArgumentIndex`	`int`	Zero-based index of an IImage function argument	When set, 2 or 3 coordinates will be displayed, depending on whether the selected image is 2D or 3D

`BoundsProviderArgumentIndex`	`int`	Zero-based index of an IImage function argument	When set, the coordinate values are bound to the selected image extents. Not implemented yet.
`DecimalPlaces`	`int`	Number of decimal digits	
`Increment`	`double`	Increment or decrement value when the up or down buttons are clicked	
`Maximum`	`double`	Maximum possible value	
`Minimum`	`double`	Minimum possible value	

Example:

```
[Adcis.Geometry.Coordinates(1,
 AxesProviderArgumentIndex=0, Minimum=0)]
static public void
  CoordinatesExample(Adcis.Images.IImage
 inIm, Adcis.Geometry.ICoordinatesD coord)
{
   // ...
}
```

Adcis.System.Functions.DefaultValue attribute

Applies to: Any argument

Constructor arguments:

Type	Optional	Value	Comment
Int		Zero-based index of an function argument	
Object		Default displayed value in the Functions panel	The object given must have the expected type for the selected argument

Or

Type	Optional	Value	Comment
Int		Zero-based index of an function argument	
Type		Type of the actual class to instanciate	
`Object []`		Array of values passed to the class constructor	

Note: the actual class to instantiate generally has the same name as the argument type, without the leading 'I', as in the following examples:

Argument type	Class to instanciate
Adcis.Geometry.ICoordinatesD	Adcis.Geometry.CoordinatesD
Adcis.Math.IRangeD	Adcis.Math.RangeD
Etc…	

Example:

```
[Adcis.System.Functions.DefaultValue(0, 1)]
   static public void DefaultValueExample1(int
    number)
   {
      // ...
   }

[Adcis.System.Functions.DefaultValue(0,
 "abc")]
static public void
 DefaultValueExample2(string name)
{
   // ...
}

[Adcis.System.Functions.DefaultValue(0,
 typeof(Adcis.Geometry.CoordinatesD),
 new object[] { 100.0, 150.0,
 Adcis.Geometry.AxisEnum.aXY })]
static public void
 DefaultValueExample3(Adcis.Geometry.
 ICoordinatesD coord)
{
   // ...
}
```

Adcis.System.Enum attribute

Applies to: Enumerations (custom or built-in)

Constructor arguments:

Type	Optional	Value	Comment
Int		Zero-based index of an function argument of an enumeration type	

Int[ ]		Allowed values of the enumeration type	

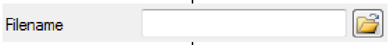
Example:

```
public enum MyEnum
{
   MyEnumValue1 = 1,
   MyEnumValue2 = 2,
   MyEnumValue3 = 3,
}
 // …
[Adcis.System.EnumAttribute(0, new int[]
 { 1, 3 })]
static public void EnumExample(MyEnum
 inMyEnum)
{
   // ...
}
```

Adcis.System.FileName attribute

Applies to: `string`, `string[]`

Constructor arguments:

Type	Optional	Value	Comment
Int		Zero-based index of a function argument of a `string` (`string[]`) type representing a file name (respectively, an array of file names)  Filename	When set, an "Open file(s)" dialog button is available:

Optional properties:

Name	Type	Value	Comment
`IsOutput`	`Bool`	True if argument is an output value. Default is false.	

Example:

```
[Adcis.System.FileName(0)]
static public void FileNameExample(string
 filename)
{
   // Open image
   Adcis.Images.IImage image = new
    Adcis.Images.Image(filename);
   Aphelion4.Application.Images.Add(image);
}
```

Adcis.Math.Interval attribute

Applies to: `Adcis.Math.IIntervalD`

Constructor arguments:

Type	Optional	Value	Comments
Int		Zero-based index of the function argument to which this attribute applies	
Int		(Ignored)	

Int	✓	Zero-based index of a IImage function argument	The selected image band count and pixel values will be used to automatically adjust the display of the interval

Example:

```
[Adcis.Math.Interval(1, 0, 0)]
static public void
 IntervalExample(Adcis.Images.IImage inIm,
 Adcis.Math.IIntervalD inInterval)
{
   // ...
}
```

Adcis.Images.Padding attribute

Applies to: `Adcis.Images.Padding`

Constructor arguments:

Type	**Optional**	**Value**	**Comment**
Int		Zero-based index of the function argument to which this attribute applies	

Int		Zero-based index of a IImage function argument	The entered value will not exceed the selected image data type maximum value (e.g., 255 for a 8-bit unsigned image)

Example:

```
[Adcis.Images.Padding(1, 0)]
static public void
 PaddingExample(Adcis.Images.IImage image,
 Adcis.Images.Padding padding)
{
   // ...
}
```

Adcis.Images.NewImageAllowed attribute

Applies to: `Adcis.Images.IImage`

Constructor arguments:

Type	Optional	Value	Comment
Int		Zero-based index of an `IImage` function argument for which a “New Image” entry will be proposed	

Int	✓	Zero-based index of an `IImage` function argument for which a “New Image” entry will be proposed	
Int	✓	Zero-based index of an `IImage` function argument for which a “New Image” entry will be proposed	

Or

Type	**Optional**	**Value**	**Comment**
Int[ ]		Zero-based indices of `IImage` function arguments for which a “New Image” entry will be proposed	

Note: This attribute is implicitly set if the argument name begins with “out”.

Example:

```
[Adcis.Images.NewImageAllowed(0)]
static public void
 ImageExample1(Adcis.Images.IImage image)
{
   // ...
}
```

```
// Attribute is implicitly set because
parameter name starts with "out"
static public void
 ImageExample2(Adcis.Images.IImage outImage)
{
   // ...
}
```

Adcis.ObjectSets.NewObjectSetAllowed attribute

Applies to: `Adcis.ObjectSets.IObjectSet`

Constructor arguments:

Type	Optional	Value	Comment
Int		Zero-based index of an `IObjectSet` function argument for which a “New Object Set” entry will be proposed	
Int	✓	Zero-based index of an `IObjectSet` function argument for which a “New Object Set” entry will be proposed	
Int	✓	Zero-based index of an `IObjectSet` function argument for which a “New Object Set” entry will be proposed	

Or

Type	Optional	Value	Comment
Int[ ]		Zero-based indices of `IObjectSet` function arguments for which a "New Object Set" entry will be proposed	

Note: This attribute is implicitly set if the argument name begins with "out".

Example:

```
[Adcis.ObjectSets.NewObjectSetAllowed(0)]
static public void
 ObjectSetExample1(Adcis.ObjectSets.
 IObjectSet objectSet)
{
   // ...
}

// Attribute is implicitly set because
parameter name starts with "out"
static public void
 ObjectSetExample2(Adcis.ObjectSets.
 IObjectSet outObjectSet)
{
   // ...
}
```

***Adcis.System.Functions.NullValueAllowed* attribute**

Applies to: Any argument

Constructor arguments:

Type	Optional	Value	Comment
Int		Zero-based index of an function argument	
String		`TypeName`	

Constructor arguments:

Type	Optional	Value	Comment
Int		Zero-based index of a function argument for which an empty value is accepted	
Int	✓	Zero-based index of a function argument for which an empty value is accepted	
Int	✓	Zero-based index of a function argument for which an empty value is accepted	

Or

Type	Optional	Value	Comment
Int[ ]		Zero-based indices of the function arguments for which an empty value is accepted	

Example:

```
[Adcis.System.Functions.NullValueAllowed(0)]
static public void NullValueExample(IImage
 image)
{
   if (image != null)
   MessageBox.Show("Image name is: " + (image
    as Adcis.Images.Image).Name.ToString());
   else
   MessageBox.Show("Image is null");
}
```

Adcis.System.Number attribute

Applies to: System primitive types (integer of floating-point value numbers).

Constructor arguments:

Type	**Optional**	**Value**	**Comment**
Int		Zero-based index of a function argument	

Optional properties:

Name	**Type**	**Value**	**Comment**
`DecimalPlaces`	`int`	Number of decimal digits	Automatically set to 0 for integer types
`Increment`	`double`	Increment or decrement value when the up or down buttons are clicked	

`Maximum`	`double`	Maximum possible value	
`Minimum`	`double`	Minimum possible value	
`RangeArgIndex`	`int`	Zero-based index of an IImage function argument	When set, the selected image first band minimum and maximum values override the `Minimum` and `Maximum` properties, respectively

Example:

```
[Adcis.System.Number(0, Minimum=0,
 Maximum=100, Increment=0.1,
 DecimalPlaces=2)]
static public void NumberExample(float
 percent)
{
  // ...
}
```

Adcis.Math.Range attribute

Applies to: `Adcis.Math.IRangeD`

Constructor arguments:

Type	Optional	Value	Comment
Int		Zero-based index of the function argument to which this attribute applies	
Int		Zero-based index of a IImage function argument	The selected image pixel values will be used to automatically adjust the default values of the range

Example:

```
[Adcis.Math.Range (1, 0)]
static public void
 RangeExample(Adcis.Images.IImage inIm,
 Adcis.Math.IRangeD inRange)
{
   // ...
}
```

G. 2. g. Additional attributes to customize the behavior of a method

Adcis.System.Functions.DefaultOverload attribute

Applies to: Any method declared as public static.

Constructor arguments: None

When a function has several overloads, the one that has this attribute will be selected by default (see the *Overloads* section).

Example:

```
static public void
 OverloadedFunctionExample(int x)
{
   // ...
}

[Adcis.System.Functions.DefaultOverload]
static public void
 OverloadedFunctionExample(int x, int y)
{
   // ...
}
```

Adcis.System.Functions.Exclude attribute

Applies to: Any method declared as public static.

The functions that have this attribute will NOT be added to the Aphelion functions list.

Adcis.System.Functions.NoProgress attribute

Applies to: Any method declared as public static.

Constructor arguments: None.

When this attribute is set, the pop-up window indicating the function progress is not displayed. This is particularly useful while debugging a function (see the *Debugging an Aphelion macro* section).

Example:

```
[Adcis.System.Functions.NoProgress()]
static public void NoProgressExample()
{
   // User can interact with Aphelion
   interface while this debug window is open
   Debugging.Tools.ShowDebugDialog();
}
```

G. 3. Loading a macro as an Aphelion plugin

When one or more custom functions have been defined in a macro, it is usually practical to have them automatically loaded when Aphelion starts. To do so, the macro must be compiled as an Aphelion plugin and added to Aphelion plugins list:

G. 3. a. Compiling a macro as a plugin

At the beginning of the C# macro, in the APHELION_HEADER region, add a line such as:

```
output "MyFunctions.dll”
```

Or:

```
output auto
```

Where “MyFunctions.dll” can be replaced by any other suitable DLL file name.

Then, compile the macro in the C# macro window: a `MyFunctions.dll` file is generated in the same folder as the macro file.

If `auto` is specified as an output filename, then Aphelion generates an automatic output filename, based on the source file name, (e.g., MyFunctions.1.dll, MyFunctions.2.dll, etc., if the source filename is “MyFunctions.cs”).

Note: the same output DLL name can be used only once in an Aphelion session. If you wish to recompile the same macro file, it is necessary to change this name, or use the `auto` keyword.

G. 3. b. Load the generated DLL as a plugin

Copy the generated DLL to any suitable directory on your computer. Then, in Aphelion Tools→Options window, select **Plugins** and add the path to the generated DLL file name.

Note: if a full path is not provided, Aphelion will search the folder where AphelionDev.exe is placed.

G. 4. Debugging an Aphelion macro

G. 4. a. Useful functions to debug a macro

The `Debugging.Tools` class offers a set of useful static methods to debug a macro. To have them available, the `Adcis .DebuggingTools.dll` library must be referenced (see *Referencing DLLs in a C# macro*).

For example:

```
Debugging.Tools.ShowDebugDialog();
```

displays a pop-up window with the specified text. The macro execution is interrupted until the user closes the window. However, on the contrary to a standard modal dialog box, the user has the possibility to interact normally with Aphelion interface (display images, change the Visualization Window properties, display/undisplay ObjectSets, etc.) while this window is open.

Note: The debugged function must have a *Adcis.System.Functions .NoProgress* attribute (see the *Adcis.System.Functions.NoProgress attribute* section) for interactions to be possible with the Aphelion GUI;

Even if the function progress window is not displayed, the mouse cursor generally has a “wait” shape during macro execution. However, this does not prevent interactions with Aphelion GUI.

Other useful functions are:

```
Debugging.Tools.Add(Adcis.Images.IImage image)
Debugging.Tools.Add(Adcis.ObjectSets.
 IObjectSet os)
```

to add an image or ObjectSet into Aphelion GUI.

G. 4. b. Using an external debugger

Aphelion C# macro window has no embedded debugging tools. However, an external debugger can be used to debug a macro. For example, use the following steps to debug a C# macro using Microsoft® Visual Studio®:

- Compile and load the macro as a plugin (see Loading a macro as an Aphelion plugin)
- Start Microsoft® Visual Studio®
- Attach Microsoft® Visual Studio® to the AphelionDev.exe process (through Tools→ Attach to Process…). Make sure the "Managed code" is selected in the "Attach To" text box.
- In Microsoft® Visual Studio®, open the C# macro file
- Set a breakpoint in the function to debug
- Select the function to debug in Aphelion C# Macro window, set its parameters and click Apply.

Note: For Microsoft® Visual Studio® to be able to debug, the C# source file, DLL file and PDB file must not be altered or moved after they have been compiled and loaded in Aphelion.

The function execution stops when the breakpoint is hit. At this point, you can see and alter variable values, or use the Immediate Window to execute statements, e.g., call the Debugging.Tools.* methods (see *Useful functions to debug a macro*).

APPENDIX H COLOR SEGMENTATION

Detecting changes in pixel intensities in binary and gray-scale images is the basis for many powerful algorithms for detecting objects in those images. Applying these techniques to the three bands or channels of color images increases the effectiveness of detection, discrimination, and measurements of objects in color images.

Aphelion Dev provides a broad range of individual algorithms that can be combined by the Dev user to create such powerful techniques for object detection. Dev also includes one tool that provides the user with a simple and effective, parameter-driven method for many object detection problems. It detects objects using Color Interval segmentation, the color equivalent of gray-level threshold.

The Aphelion suite of products also includes an optional Dev extension product that seamlessly integrates three additional color segmentation tools that use different techniques: color distance, region growing, and morphological partioning.

While these four techniques are previously described, more technical detail is provided in the following paragraphs.

H. 1. Color Interval

The Interval algorithm is based on processing of the three color bands independently in the user-selected color space (e.g., RGB). An object is defined to be a set of adjacent pixels having color band values that are each within a given band tolerance factor of a reference color. The reference color value is defined by the user clicking (i.e., selecting a pixel) inside an object that is representative of the objects to be detected. The user can vary the tolerance factors by adjusting three, on-screen dials. The band tolerances can be adjusted individually or collectively (as a single factor). The effects of tolerance factor changes are displayed in real-time in the Visualization window as a graphics overlay, in a user-selected color (see color button adjacent to the Opacity slider).

A color pixel is in an object if its color band values lie in their respective band intervals as defined below:

[Va - Ta, Va + Ta]; [Vb - Tb, Vb + Tb]; [Vc - Tc, Vc + Tc]

Where

a, b, and *c* are the 3 bands in the color space.

Va, Vb, and Vc are the color band values of the reference pixel.

Ta, Tb, and Tc are the tolerance values for each color band (defined by the three, band dials). The tolerance values for any band can range between the minimum and maximum values for the selected color band. For example, the range of the tolerance dials for RGB color bands is 0 - 255 (assuming 8-bit color pixels) and for the Hue band is 0 - 2π.

Furthermore, if Vx < Tx, then Vx-Tx = 0 and if Tx > Vmax, then Vx+Tx = Vmax

The Window Size parameter defines the size of the neighborhood used to estimate the color at a given pixel location (in the Tools→Options→ Advanced→Behavior→ObjectExtraction window). The Color Estimation Method parameter can take the following values: cemAverage, cemMinMax, and cemMean (in the Tools→Options→Advanced→ Behavior→ObjectExtraction window).

Note: It is often the case that the interesting objects in an image have more than one significant color interval. After selecting the first reference pixel and setting the interval dials, additional color intervals can be added to the segmentation. This is done by control right-clicking on additional reference pixels as needed. Thus, each added color interval forms a union of intervals with the first color interval.

H. 2. Color Distance

This segmentation method is based on the generalized Euclidean formula for measuring distance between two points (i.e., square root of the sum of the squares). For example, the distance between two color pixels is the square root of the sum of the squares of the distances between the pixels in each color band. When there is a scaling difference between the bands of a color space, the bands are first normalized before distances are computed. Various distance formulae are available depending on the color space chosen Tools→Options→Advanced→Behavior→ObjectExtraction→ColorSpace window. A threshold is performed in the *Color Space* based on the sphere whose center is a user-specified reference point and whose radius is the computed distance value. The pixels contained by the sphere form the object defined by the distance formula and the reference point.

The reference point is either a single pixel or a neighborhood of a single pixel depending on the value specified by the parameter *Tools→Options→Advanced→Behavior→ObjectExtraction→ColorEstimation WindowSize*. If the neighborhood contains more than one pixel, the color of the reference point is defined by the *Color Estimation Method* parameter which can take the following values: Average, Median, and MinMax (see **Tools→Options→Advanced→Behavior→Object Extraction→ColorEstimationMethod**).

Note: 1.) If an object contains multiple, significant colors, then multiple color space spheres can be created and combined. To do this, hold down the control key while clicking a reference point for each such color in an area of the object representative of that color. The threshold operation will then be performed using the union of the spheres.

2.) All spheres will have the same radius, namely, the value of the Distance parameter.

3.) The effects of adding reference points and changing the Distance value are displayed in real-time in the Image Viewer window.

H. 3. Region Growing

The Region Growing algorithm is a semi-automatic segmentation based on user-defined seed points placed either inside objects of interest (left-click) or in non-object regions (shift-left-click). A seed point is either a single pixel or a neighborhood of a single pixel depending on the parameter *Seeds Size* found at the Tools→Options→Color screen.

A seeded region's color is defined to be either a) the average color of the starting seed or b) the current average color value of the seeded region's pixels, depending on the *Adaptive Colors* parameter found at the Tools→Options→Color screen.

Starting from the seed points as initial regions, other qualifying pixels are iteratively added to each seeded region until no further qualifying pixel can be found. A pixel is qualified to be added to a seeded region if all of the following are true:

- The pixel is not part of an existing seeded region;
- The pixel is adjacent to at least one of the region's boundary pixels; and
- The pixel's color value is within the specified color *Distance* setting of the seeded region's color.

The growth of a seeded region ends when there are no remaining pixels qualified to be in that region.

A typical Region Growing object extraction proceeds as follows.

1. Place an object seed in a structure you want to extract as an object.
2. Vary the color *Distance* value for that seed so that the object is extracted as close as possible to the desired result.

3.a. If the color *Distance* value chosen does not completely extract the object, then repeat step 1 and add a new object seed in an area where the current object is not yet extracted.

3.b. If the chosen *Distance* value extracted regions not interior to the object, then unextract such a region by dropping a non-object seed on it. It may be necessary to repeat this step multiple times to completely unextract a region. When you are satisfield with the current object's extraction, return to step 1. This three-step process is repeated until all structures of interest are satisfactorily extracted (i.e., covered by the graphics overlay with the selected *Opacity* color such that no non-object areas are covered).

Note that at least one object seed must be placed in each object of interest. Each time a new seed is dropped in the image, the then current Distance value is permanently fixed for the immediately prior seed and that value becomes the initial value for the new seed. The Distance value can then be varied for the new seed without affecting the results for the prior seeds.

H. 4. Morphological Partition

The Morphological Partition method is a semi-automatic segmentation based on a watershed transform. A partition of the color image into a set of regions is computed, using the color gradient. Only the N dominant regions are kept, or in other words, only the last N lakes (in terms of lake volume) of the watershed are kept in the result of the segmentation. The *Number of Regions* slider is used to specify the maximum number of dominant regions of the partition that will be kept in the final result.

When the mouse pointer is moved into a region, that regions turns green in the object overlay to preview the region as an object. The green color is removed when the mouse pointer moves out of the region.

The user must identify each region that forms an object of interest by left-clicking or control-left-clicking in those regions. If the user left-clicks in a region, it will become selected as an object and all other objects previously selected will become unselected. If the user control-left-clicks in a region, that region becomes selected as an object, but any previously selected regions remain selected as objects. A selected object will switch from the green preview color to the color specified in the Opacity color box when the mouse pointer is moved out of the selected region.

Note: The Morphological Partition is first computed when this segmentation method is selected. It is a process that may take a while depending on the image size and the number of maxima and minima in the image. Once it is computed, the effects of moving the Number of Regions slider are displayed immediately.

H. 5. Color Space Definitions

Aphelion Dev supports the following color spaces:

RGB:	Red, Green and Blue
HSI:	Hue, Saturation and Intensity
HSV:	Hue, Saturation and Value
YUV:	Luminance and Chrominance
Lab:	Lightness and Red/Green and Yellow/Blue Chrominance
YIQ:	Luminance and Chrominance
XYZ:	Virtual primary color space
Luv:	Lightness and Chrominance
LCH:	Lightness, Chromaticity/Saturation, Hue

H. 5. a. YIQ color space

RGB → YIQ conversion

$$Y = 0.299 \text{ x } R + 0.587 \text{ x } G + 0.114 \text{ x } B$$

$$I = 0.596 \text{ x } R - 0.275 \text{ x } G - 0.321 \text{ x } B$$

$$Q = 0.212 \text{ x } R - 0.532 \text{ x } G + 0.311 \text{ x } B$$

H. 5. b. HSI color space

RGB → HSI conversion

$$\text{Intensity} = I = \frac{(R + G + B)}{3}$$

$$S = \begin{cases} 1 - \dfrac{\min[R, G, B]}{3} & \text{if} \quad I > 0 \\ 0 & \text{if} \quad I = 0 \end{cases}$$

$$H = 0 \text{ if } I = 0 \text{ or } S = 0$$

$$H = \begin{cases} 2\pi - a\cos\left(\dfrac{0.5(2R - G - B)}{\sqrt{R^2 + G^2 + B^2 - RG - RB - GB}}\right) & \text{if} \quad B > G \\ a\cos\left(\dfrac{0.5(2R - G - B)}{\sqrt{R^2 + G^2 + B^2 - RG - RB - GB}}\right) & \text{if} \quad B \leq G \end{cases}$$

H. 5. c. HSV color space

RGB → HSV conversion

$$\text{Max} = \text{Max}(R,G,B) \qquad \text{Min} = \text{Min}(R,G,B)$$

$$S = \frac{\text{Max} - \text{Min}}{\text{Max}}$$

$$R' = \frac{\text{Max} - R}{\text{Max} - \text{Min}}$$

$$G' = \frac{\text{Max} - G}{\text{Max} - \text{Min}}$$

$$B' = \frac{\text{Max} - B}{\text{Max} - \text{Min}}$$

If saturation, S, is equal to 0, then H is undefined (i.e., the color has no hue therefore it is monochrome), otherwise:

If $R = \text{Max}$ and $G = \text{Min}$:	$H = 5 + B'$
Else if $R = \text{Max}$ and $G \neq \text{Min}$:	$H = 1 - G'$
Else if $G = \text{Max}$ and $B = \text{Min}$:	$H = R' + 1$
Else if $G = \text{Max}$ and $B \neq \text{Min}$:	$H = 3 - B'$
Else if $R = \text{Max}$:	$H = 3 + G'$
Otherwise:	$H = 5 - R'$

H. 5. d. XYZ color space

RGB → XYZ conversion

$$X(\lambda) = 2.7690\ R_c(\lambda) + 1.7518\ G_c(\lambda) + 1.1300\ B_c(\lambda)$$

$$Y(\lambda) = 1.0000\ R_c(\lambda) + 4.5907\ G_c(\lambda) + 0.0601\ B_c(\lambda)$$

$$Z(\lambda) = 0.0565\ G_c(\lambda) + 5.5943\ B_c(\lambda)$$

Note: The conversion above depends on the color temperature. Refer to technical papers to learn more about this color conversion.

xyz color space

$$x = \frac{X}{X + Y + Z}$$

$$y = \frac{Y}{X + Y + Z}$$

$$z = \frac{Z}{X + Y + Z}$$

Note: The **xyz** corresponds to the new color space and ***XYZ*** the old color space.

H. 5. e. L*u*v* color space

XYZ → L*u*v* conversion

$$L^* = \begin{cases} 116\sqrt[3]{Y/Y_W} - 16 & \text{if} \quad Y/Y_W > 0.008856 \\ 903.3 \cdot Y/Y_W & \text{if} \quad Y/Y_W \leq 0.008856 \end{cases}$$

$$u^* = 13 \cdot L^* \left(u' - u'_W\right)$$

$$v^* = 13 \cdot L^* \left(v' - v'_W\right)$$

$$u' = \frac{4X}{X + 15Y + 3Z}$$

$$v' = \frac{9Y}{X + 15Y + 3Z}$$

u'_W and v'_W have the same definitions as u' and v' but are applied to the white point reference.

$$u'_W = \frac{4X_W}{\left(X_W + 15Y_W + 3Z_W\right)}$$

$$v'_W = \frac{9Y_W}{\left(X_W + 15Y_W + 3Z_W\right)}$$

H. 5. f. YUV color space

RGB → YUV conversion

$$Y = 0.299\,R + 0.587\,G + 0.114\,B$$

$$U = -0.147\,R - 0.289\,G + 0.436\,B$$

$$V = 0.615\ R - 0.515\ G - 0.100\ B$$

H. 5. g. La*b* color space

RGB → L*a*b* conversion

$$\text{Luminance} = L = 0.2989\ R + 0.5866\ G + 0.1145\ B$$

$$a^* = 500 \times \left(f\left(X / X_W\right) - f\left(Y / Y_W\right)\right)$$

$$b^* = 500 \times \left(f\left(Y / Y_W\right) - f\left(Z / Z_W\right)\right)$$

$$f(x) = \begin{cases} \sqrt[3]{x} & \text{if} \quad x > 0.008856 \\ 7.787x + 16/116 & \text{if} \quad x \leq 0.008856 \end{cases}$$

H. 5. h. LCH color space

Lab → LCH conversion

$$L = L^*$$

$$C = \sqrt{a^{*2} + b^{*2}}$$

$$H = \begin{cases} 0 & \text{if} \quad a^* = 0 \\ \left(\arctan\left(b^* / a^*\right) + k\pi / 2\right) / 2\pi & \text{if} \quad a^* \neq 0 \end{cases}$$

k=0 if a*>=0 and b*>=0

k=1 if a*>0 and b*<0

k=2 if a*<0 and b*<0

k=3 if a*<0 and b*>0

APPENDIX I OPTIONS→ADVANCED MENU

Appearance		
Sub-Category	**Parameter Name**	**Description**
Drawing	DefaultArrow Properties	Graphic properties of arrows associated with Line shapes. See below for the description of the properties
	DefaultArrow Properties > AutoScale	Automatically scale the arrows when a zoom in or a zoom out is applied to the image
	DefaultArrow Properties > EndSize	Default size of the arrow at the end of the line
	DefaultArrow Properties > EndStyle	Default style of the arrow at the end of the line
	DefaultArrow Properties > Proportion	Scaling factor applied to the arrows when ProportionSize is set to True
	DefaultArrow Properties > ProportionalSize	The arrows are automatically rescaled depending on the line length when the parameter is set to True. The arrow size is applied without rescaling when it is set to False

Sub-Category	**Parameter Name**	**Description**
Drawing	DefaultArrow Properties > StartSize	Default size of the arrow at the start of the line
	DefaultArrow Properties > StartStyle	Default style of the arrow at the start of the line
	DefaultCircle Properties	Graphic properties of Circle shapes. See below for the description of the properties
	DefaultCircle Properties > AutoScale	Automatically scale the Circle boundary and center when a zoom in or a zoom out is applied to the image
	DefaultCircle Properties > CenterVisible	Display a cross in the center of the circle
	DefaultCircle Properties > InnerSize	Default size of the reticle in the cross defining the circle center. Only used when Symbol is set to CrossHairs[2]
	DefaultCircle Properties > Size	Default size of the cross defining the circle center
	DefaultCircle Properties > Symbol	Default style of the cross defining the circle center

[2] If the parameter Symbol is set to either psCrossHairs1, psCrossHairs2, or psCrossHairs3.

Sub-Category	**Parameter Name**	**Description**
Drawing	DefaultPoint Properties	Graphic properties of Point shapes. See below for the description of theproperties
	DefaultPoint Properties > AutoScale	Automatically scale Point shapes when a zoom in or a zoom out is applied to the image
	DefaultPoint Properties > InnerSize	Default size of the reticle in the cross defining a point. Only used when Symbol is set to CrossHairs[2]
	DefaultPoint Properties > Size	Default size of the cross defining a point
	DefaultPoint Properties > Symbol	Default style of Point shapes
	DefaultStandard Properties	Graphic properties of any shapes. See below for the description of the properties
	DefaultStandard Properties > BackColor	Default background color applied to objects computed in the Object Extraction and/or Objet Editing tasks
	DefaultStandard Properties > BackMode	Default background mode applied to objects computed in the Object Extraction and/or Objet Editing tasks

Sub-Category	**Parameter Name**	**Description**
Drawing	DefaultStandard Properties > FillColor	Default color that fills objects computed in the Object Extraction and/or Objet Editing tasks. No filling is done if Style is set to Null
	DefaultStandard Properties > FillStyle	Default style filling objects computed in the Object Extraction and/or Objet Editing tasks
	DefaultStandard Properties > HatchStyle	Default style of hatch filling objects computed in the Object Extraction and/or Objet Editing tasks
	DefaultStandard Properties > LineColor	Default boundary color of objects computed in the Object Extraction and/or Objet Editing tasks
	DefaultStandard Properties > LineStyle	Default boundary style of Line shapes
	DefaultStandard Properties > LineWidth	Default width of objects computed in the Object Extraction and/or Objet Editing tasks
	DefaultStandard Properties > Visible	Define if objects computed in the Object Extraction and/or Objet Editing tasks are visible or not

Sub-Category	**Parameter Name**	**Description**
Drawing	DefaultText Properties	Graphic properties of text. See below for the description of the properties
	DefaultText Properties > AutoScale	Automatically scale the text when a zoom in or a zoom out is applied to the image
	DefaultText Properties > Font	Default font and size of the font applied to a text
General Appearance	ListSeparator	Symbol used when displaying arrays (e.g. ObjectSet attributes containing arrays)
Geometry	Location	Position of the upper left corner of the user interface
	Size	Size of the window of the user interface
Images	BlinkEnable	Enable object blinking when selecting a 3D object
	ClipImageView	Clip shapes and text defined outside the image frame
	DefaultColor Space	Default color space when reading image files
	DefaultIsoValue	Default IsoValue when creating a new IsoSurface viewer
	DefaultSub SamplingFactor	Default subsampling factor when displaying a 3D image

Sub-Category	**Parameter Name**	**Description**
Images	DefaultThreeD Viewer	Default viewer used when displaying a 3D image
	DisplayThreeD ImagesOnOpen	Automatically display a 3D image when opening the image from disk
	FitToWindow	Fit the image to the window size when displaying the image
	JPEGQuality	Compression quality when writing JPEG images (0 = worst, 100 = best)
	MIVTColumn Count	Number of columns in the Visualization window. Value 2 for this parameter and 1 for the next displays 2 images side by side. Value 1 for this parameter and 2 for the next one displays 2 images on top of each other.
	MIVTRowCount	Number of rows in the Visualization window
	UseImageBounds Only	Limit the scrollable zone to the image
	UseMemory SavingImage View	2D ImageView controls to minimize the amount of memory used

Sub-Category	**Parameter Name**	**Description**
Images	UseMemory SavingIsoSurface	Select a memory saving implementation of the IsoSurface viewer (very useful on PCs with a small amount of memory)
	WindowBack Color	Background color of the Visualization window
ObjectSets	CaseSensitive Attributes	Specify if attribute names are case sensitive or not
	CSVDelimiter	Delimiter used when exporting an ObjectSet as a CSV file
	DefaultAttribute	Name of the default attribute (BITMAP by default)
Behavior		
Sub-Category	**Parameter Name**	**Description**
Acquisition	AutoFreeze	Freeze the live image when the window is no longer active
	AutoStop	Complete all on-going captures when selecting a new Task
	SelectCaptured Image	The captured image view is automatically activated when the Snap button is pressed.

Sub-Category	**Parameter Name**	**Description**
Developer	BasicScript HistoryRecording Enabled	Functions run from the Function Panel are automatically recorded in the BasicScript Command window
	InputImage SelectionMode	Define the way input images are selected in the Functions Panel
	RecentlyUsed Functions MaximumSize	Maximum function number stored in the recently used function list
	ShowCompati-bilityGroups	Add a menu to display functions in the Process menu as in version 3.2 of Aphelion
	TrackOutput Image	Automatically display the output image after applying a function
	TrackSelected Image	Automatically display the image when it is selected in the Input Image dialog box
Editng	Connectivity	Connectivity of shapes generated in the Object Editing task
Importation	BandCount	Value of the “Number of bands” field
	Color	Value of the “Color image” checkbox

Sub-Category	**Parameter Name**	**Description**
Importation	Colorspace	Value of the “Color space” field
	HeaderSize	Value of the “File header size” field
	Height	Value of the “Image size” field (first parameter)
	Interleaved	Value of the “Interleaved bands” field
	PixelDataType	Value of the “Pixel data type” field
	SwapBytes	Value of the “Swap bytes” field”
	ThreeD	Importation mode (3D or 2D)
	Width	Value of the “Image size” field (second parameter)
Kriging	DefaultMaximum Offset	Value of the maximum offset used for variograms
	IncludeSinus Cardinal	Use Sinus Cardinal functions when automatically generating a model
	KernelSize	Kernel definition for the filtering process
	ModelAutoFill	Automatically generate the settings of the kriging model during model creation

Sub-Category	**Parameter Name**	**Description**
Kriging	ModelAutoFill ModelAutoFill Maximum FunctionCount	Maximum number of functions allowed when automatically generating a model
	ModelAutoFill MinimumError Variation	Convergence criterion when automatically generating a model
	OverwriteImages	Flag to overwrite input images when applying the filter
	SillAutoFit Locked	Flag to disable sill autofit (advanced parameter)
	ProcessingThread PoolThreadCount	Number of threads used when performing Image / ObjectSet processing operations (Set to value 0 forautomatic settings)
Object Extraction	ColorEstimation Method	Estimate the color at the seed location when performing a user- assisted Object Extraction
	ColorEstimation WindowSize	Window size used to estimate the color at the seed location when performing a user-assisted Object Extraction
	ColorSpace	Default color space used when extracting objects

Sub-Category	Parameter Name	Description
Object Extraction	ConnectedTo SeedsOnly	Keep only objects connected to the seeds when performing a user- assisted Object Extraction
	Connectivity Graph	Connectivity used when performing a user-assisted Object Extraction
	FillHoles	Fill holes when performing a user- assisted Object Extraction
	Opacity	Opacity value used when performing a user-assisted Object Extraction
	Partition MaximumRegion Count	Maximum number of regions when performing morphological partition based Object Extraction
	RegionGrowing AdaptiveColors	Select the adaptive color method when performing a Region Growing segmentation
	RegionGrowing SeedSize	Seed size when performing a Region Growing segmentation
	RegionGrowing SeedsVisible	Display seeds when performing a Region Growing segmentation

Sub-Category	**Parameter Name**	**Description**
Object Extraction	TransparentColor	Color value used when performing a user-assisted Object Extraction
Files		
Sub-Category	**Parameter Name**	**Description**
Files	LastFolder	Define the last folder used to save any Aphelion data as a file. This value is automatically set when a file is saved.
	ReloadImages And ObjectSets	Reload images and ObjectSets created during the last Aphelion session
	RootProjectFolder	Root folder for all project sub-folders
General		
Sub-Category	**Parameter Name**	**Description**
Files	SaveImages	Prompt the user to save images
	SaveObjectSets	Prompt the user to save ObjectSets
	SaveScripts	Prompt the user to save scripts/macros
Identification	Administrator Password	Password required to enter in Administrator Mode

Sub-Category	**Parameter Name**	**Description**
Identification	AutoEnter Administrator Mode	Automatically enter in Administrator Mode when starting Aphelion if the Administrator password is empty
	CompanyName	Company Name
	Culture	Culture information as defined in Windows
	ForceEnglish Translation	Enforce the translation to English for every character string in the user interface
	Identification Code	Machine identification code
	LicenseCode	Installation code
	ProductName	Aphelion Dev/Lab
	UserName	User's name
	Version	Version number
Misc		
Sub-Category	**Parameter Name**	**Description**
Diagnostics	ShowVTKLog Window	Display the VTK log window
	TraceLevel	Level of tracing. From 0 to 3 (0: highest priority information, 3: lowest priority information)
	Verbose Exceptions	Flag to display stack information when an exception occurs