Imaging Automation

Jerry D. Horne
3055 Lynview Drive
San Jose, CA 95148
jdhorne@hotmail.com

Abstract

Multiple software programs, techniques, scripts, and related routines now exist to automate much of the work of taking, processing, analyzing, and extracting data from CCD Images. Five categories of such software programs are examined. The automation capability of Maxim DL and the ASCOM modules is demonstrated, together with specific examples of automation routines to control a telescope, take and process images.

1. Introduction

The term *image automation* can be used in the sense that multiple images can be processed, or multiple operations can be performed, with minimal or no operator interaction. In automating these tasks, this paper will look at what can be done to more quickly take, process, and extract data from, a large number of images. The idea of Imaging Automation was documented thirty years ago by Bijaoui et al (1977), and now routines now exist to handle everything from positioning the telescope, setting filters, exposing images, processing images, and analyzing data.

2. Types of Automation

For the serious amateur and professional astronomer, there appear to be five categories of image automation software available. These software categories range from sophisticated commercial programs, backed by customer response teams, to one-of-a kind scripts written by an amateur astronomer, simply trying to find a way to solve a particular imaging or observing problem.

2. 1. Automation Category 1

This category concerns commercial software programs that both control multiple CCD cameras and allows the user to do at least some image processing. These are usually Windows based programs, often written in C++, usually cost hundreds of U.S. Dollars, and are sold by various vendors. These programs can operate on multiple images simultaneously, have the ability to create and utilize master frames for processing. Some can run scripts or plug-ins, and have ability to integrate with other software programs (e.g. the Sky®) and hardware (telescopes, filter wheels, mounts, etc). One program even has the ability to record commands and play them back. Examples programs able to interface with a number of cameras are Maxim DL/CCD® (2007), Astro Art®, and CCDSoft®. Examples of more proprietary, and somewhat less capable programs, are that software included with specific CCD cameras such as Starlight Xpress, or the Meade DSI.

2. 2. Automation Category 2

The second category of automation utilizes the scripting ability of other commercial programs and/or common software and hardware interfaces to provide another level of automation and specialization. In a sense, these so called "3rd party" programs provide the glue to stick together a variety of software and hardware in a coherent and structured manner. They represent perhaps the ultimate in a user's ability to automate imaging operations.

Typical abilities of such software include automatically taking the images from observing lists, focusing, filter selection, automated dark, flats, and bias images, open/closing and moving the dome, monitoring weather instruments, measuring seeing and image quality, parking the telescope, and warning the operator of error or unusual conditions. Some software even allows multiple user and internet access to an imaging setup.

Examples of this are ACP®, CCD Commander® (2007), and CCD Autopilot® (CCDWare 2007) We also place in this category, the modules of ASCOM (2007), the Astronomy Common Object Modules, since they provide access by multiple programs to multiple types of hardware.

2. 3. Automation Category 3

In this third category of automation software, the programs are designed specifically for image processing, with no hardware control capabilities. These programs can operate on multiple images simultaneously, have the ability to create and utilize master frames for processing, have the ability to extract photometric observations, have multiple filter and processing algorithms, and can usually read and write multiple file formats.

Obvious examples are AIP4Win® Mira®, Canopus®, IRAF® (2007), and a version of Maxim DL®, together with other lesser know programs such as Iris, Cadet; much older programs such a Super Fix or DAOPHOT, and even a Spanish-language program such as Laia.

2. 4. Automation Category 4

This fourth category concerns software tools and spreadsheets designed to perform a specific task related to observing and imaging such as Julian Date conversion, data analysis tasks, or converting one image format to another. These tools are usually provided as shareware or commercial software.

Examples of this category would be Lew Cook's spreadsheet tools (Cook 2007), software from the AAVSO such as *PC Obs* or *WWZ*, Ron Wodaski's tools (Wodaski 2002), or Tony Vanmunster's period analysis software, *Peranso* (Vanmunster 2003).

2. 5. Automation Category 5

The fifth and final category of automation is made up of non-commercial, individually written scripts, plug-ins or stand-alone programs that usually access either the scripting ability of other software programs, and/or the drivers associated with specific hardware. This is probably the most diverse software category, with its members ranging from fairly sophisticated tools to VB scripts that perform a very limited function. These scripts, tools, or sometimes just bits of code are usually available over the internet as shareware or freeware, and may or not have user support or license issues, and of course may or not work for particular applications.

Examples of items in this category are automated photometric extraction, image processing tools, telescope and observing routines, or color and image adjustment software.

3. Automation Requirements

For astronomers wishing to automate a specific imaging task, there appear to be a few obvious requirements. First, one must be able to get the computer to command, or talk to the camera, telescope, or accessory. Secondly, you must be able to monitor or receive feedback from this same hardware. Both of these requirements imply using some sort of programming language or interface to the hardware involved, usually interacting with the hardware driver.

When considering the development of a specific automation task, using Maxim DL®, by Diffraction Limited, provides such an interface, since it has a scripting interface, for both Visual Basic and Visual C++, it can meet a wide range of automation needs, and it has a wealth of embedded commands and functions for the camera, telescope, and for image processing. An automation routine in Maxim DL can be developed as a stand-alone executable program, a plug-in, or a visual basic script, depending on the sophistication of the program, its data storage and access needs, and the ability of the observer/programmer.

A simple automation routine can be run with Maxim DL's ***Run Script*** function, selectable from the File Menu. Similarly Maxim DL's Plug-ins are added and activated via the top level pull-down ***Plug In*** menu. For stand-alone executable programs, the decision to use Visual Basic or Visual C++ is a usually stems from the programmers preference and abil-

```
Dim WithEvents Cam As CCDCamera                 ‘Camera object
Public LX200_Int As POTH.Telescope              'LX200 Interface Object from ASCOM
Public PPP As PinPoint.Plate                    'Pinpoint solution object
Public MaximApp As MaxIm.Application            'maxim application object
Public MaximDoc1, MaximDoc2 As Document         'image document objects
Public Tel_u As DriverHelper.Util               'Telescope Utlities Object from ASCOM
```

```
Set Cam = CreateObject("Maxim.CCDCamera")           'Instantiate CCD camera object
Set LX200_Int = CreateObject("POTH.Telescope")      'instantiate telescope driver
Set PPP = CreateObject("PinPoint.Plate")                'Instantiate Pinpoint Plate Object
Set MaximApp = CreateObject("Maxim.Application")    'Instantiate Application
Set MaximDoc1 = CreateObject("Maxim.Document")      'Instantiate 1st Maxim Document
Set MaximDoc2 = CreateObject("Maxim.Document")      'Instantiate a 2nd Maxim Document
Set Tel_u = CreateObject("DriverHelper.Util")       'Instantiate telescope driver utilities object
```

ity, although certain embedded Maxim DL commands, require a variant-type array data structure, that is handled easiest in Visual Basic, and VB is used for the examples in this paper.

An important extension to Maxim DL's routines come from the Astronomy Common Object Model (ASCOM) modules, developed by Bob Denny, and others. Multiple objects from the ASCOM modules can be coupled with Maxim DL to provide extend a programs access to specific telescope mounts, focusers, and observatory interfaces.

4. Examples of Automation

The key software interface components for Maxim DL is its application and document objects. The application object provides for camera and telescope connections, while the Document object provides access to a variety of functions including image calibration, alignment, photometry, image filtering, and other measurements. The ASCOM modules provide other objects which help implement automation routines.

Typically, the object that the programmer intends to use must first be declared as a local or global variable, then instantiated before use. For example the Maxim DL and ASCOM objects for a camera, telescope, pinpoint plate, application, and document are declared (in VB) as shown in the first box below while the objects are, instantiated as shown in the second box below.

4. 1. Connecting to Telescope and Camera

As a simple example of automation, a script can be used to connect these declared and instantiated objects to the CCD camera and telescope as follows.

In this code example, the telescope and CCD hardware connection is made simply by setting the value of the object connection property to *true*.

```
'------------- Connect Hardware -------------------
' Description:  This routine connects the
' ccd and telescope
'
' Inputs:
'         None
' Outputs: None
'
' Calls:
'       Cam.LinkEnabled
'       LX200_Int.Connected
'   -------------------------
Private Sub ConnectHardware

 'connect to telescope
   LX200_Int.Connected = True

  'connect to CCD
   Cam.LinkEnabled = True
End Sub
```

4. 2. Taking Images

In terms of imaging, there are two basic methods for taking such images using Maxim's scripting capability, either exposing a single image at a time, or using its intrinsic sequence function.

The following two scripts shows an example of taking a single image, and then waiting for the CCD camera to download the image. The exposure routine returns a value which can be used to test the success or failure of the command.

Alternately, a different routine can be used to start an image sequence that has been previously defined in Maxim DL and saved to a file. The sequence file contains the exposure time, exposure type, filter, and other information.

```
'----------------- Take Single Image ------------------
-
'
' Description:  This routine commands the
‘ camera to take an image
'
' Inputs:
‘
'    Image_Duration:
‘          single_exposure time in seconds
'    Image_Type:
‘         integer_Type of image (1 = light, 0 =
dark)
‘   Filter_type :
‘       integer_filter number to use (if applicable)
' Outputs: None
‘
‘ Calls:
‘        Cam.Expose
‘
'     --------------------------
Private Sub TakeImage (Image_Duration As
single, Image_type As integer, Filter_type as
integer)

‘return value for image routine
   Dim temp_result as Boolean

‘take the image
   temp_result = Cam.Expose(Image_Duration,
       Image_type,Filter_type)

' wait for image ready
Do While CCD_ImageDone = False
 ‘allow other processes to continue
   DoEvents
Loop
End Sub
```

```
'------------ Start Sequence  ----------------------
'
' Description:  This routine starts an imaging
sequence
'
' Inputs:
'     Sequence_file:
‘        string_full path and filename of saved
sequence file
‘
' Outputs: None
‘ Calls:
‘       Cam.StartSequence
'       --------------------------
Private Sub TakeSequence (Sequence_file As
String)

   Cam.StartSequence(Sequence_file)

      End Sub
```

4. 3. Loading and Aligning Images

```
'---------- Load and Align Files ---------------
'
' Description:  This routine handles ‘alignment
of the reference and image ‘files
'
' Inputs:
'    File1:
‘    string_Reference File path and name
‘
'    File2:
‘      string_Image File path and name
‘
' Outputs: None
' Calls: MaximDoc1.OpenFile
'          MaximDoc2.OpenFile
'          MaximDoc1.AlignImages
'    --------------------------
Private Sub LoadandAlign(ByVal File1 As
String,
      ByVal File2 As String)

'load reference file into the doc object
     Call MaximDoc1.OpenFile(File1)

'load image file
     Call MaximDoc2.OpenFile(File2)

'don't abort program for align problem
    On Error Resume Next

'call Maxim routine to align them
      Call MaximDoc1.AlignImages(1, False)

End Sub
```

An example of using the document object is shown in a routine to load and align images. Maxim DL provides software access to standard FITS load functions and image align routines.

The code segment in Visual Basic shown above is used to load and align two images. The routine takes as input two complete file names.

4. 4. Astrometric Solve

An automation routine to astro-metrically solve an image using a large star catalog such as GSC, is shown in the following VB code segment by using the Pinpoint Engine within Maxim DL. The routine takes as input the image file name and the approximate RA and Dec of the center of the image.

```
'--------------------- Solve Image ---------------------
'
' Description:  This routine handles the
astrometric
‘   solve of the image
'
' Inputs:
'   RA:
‘     single_approximate Right Ascension
‘     coordinates of center of image
'   Dec:
‘      single_approximate Right Ascension
coordinates
‘      of center of image
'   File1:
‘      string_Image File path and name
' Outputs: None
' Calls: PPP.FindImageStars
‘           PPP.FindCatalogStars
‘           PPP.Solve
'     --------------------------


Private Sub SolveImage(ByVal File1 As String,
   ByVal RA As Single, ByVal Dec As Single)
  'attach image to pinpoint engine
       PPP.AttachFITS (File1)

  ‘ Tell PinPoint where the telescope is pointing
    ‘in RA                                          '
        PPP.RightAscension = RA
     ‘in Dec
        PPP.Declination = Dec

     ‘set image parameters for the CCD
     ‘Horizontal
        PPP.ArcsecPerPixelHoriz = 1.13
     ‘Vertical
        PPP.ArcsecPerPixelVert = 1.09

    'find the stars in the image
        PPP.FindImageStars
    'get the stars from the catalog
         PPP.FindCatalogStars

    ‘solve the image
        PPP.Solve

End Sub
```

4. 5. Stacking Images

Stacking images is useful for a variety of purposes and can be accomplished via an automation routine. The following routine stacks a series of images in a given directory, using Maxim DL's auto-star matching routine, and saves the resulting combined image.

```
‘-------------------- Stack Images -----------------
'' Description:  This routine handles the
combining  or stacking of images
'
' Inputs:
'    temp_filemask:
‘     String_full path & mask of files to be
stacked
‘     (e.g. C:\temp\m57*.fts)
‘    save_filename:  String_full path and name
‘    of stacked image file
'
' Outputs: None
' Calls: MaximDoc1.OpenFile
‘          MaximDoc1.CombineFiles
‘          MaximDoc1.SaveFile
‘  -------------------------
Private Sub Stack_Images(temp_filemask As
String,  save_filename As String)

'Directory of files to stack
   Dim file_Dir As String
‘ first file name
  Dim temp_filename As String

'get path of files
       file_Dir = Mid(temp_filemask, 1,
Len(temp_filemask) - 5)
‘get first file name
    temp_filename = Dir(temp_filemask)
 ‘build path and filename
     temp_filename = file_Dir & temp_filename
 'load the file into the document object
     Call MaximDoc.OpenFile(temp_filename)
 ‘allow for error result from Maxim on stacking
     On Error Resume Next

‘combine with other files in directory
‘ using auto-star matching
     Call MaximDoc.CombineFiles(file_mask, 1,
False, 0, False)

 '   Save the file as 32-bit fits
     MaximDoc.SaveFile(save_filename,
mxFITS, False, 2, 0)
Exit Sub
```

4. 6. Photometry

```
'------ Perform Photometric Measurement -------
------
'' Description:  This routine measures the
‘magnitude  of the variable star in an image
'
' Inputs:
'  VarX:  integer_X coordinates of the variable
star
‘  VarY: integer_Y coordinates of the variable
star
'  CX:   integer_X coords of the comparison
star
‘  CY:  integer_Y coords of the comparison
‘  CM:  single_magnitude of the comparison
star
‘
' Outputs: single_magnitude of the variable
star
' Calls:
‘   MaximDoc2.CalcInformation
'       -------------------------
Private Function GetPhotometry(VarX, VarY,
CX, CY,  CM As Integer) As Single

Dim Star_Info As Variant   ‘  storage for star
info
Dim Var_Int, As Double  ‘ variable star
intensity
Dim Comp_Int As Double ‘comp star intensity
Dim MV As Single   ‘measured mag of var star
Dim MRings(2) As Integer    ‘rings setup

   MRings(0) = 6       ‘ aperture setting
   MRings(1) = 3       ‘gap setting
   MRings(2) = 6       ‘annulus
‘Get variable star data from image
   Star_Info =
MaximDoc2.CalcInformation(VarX, VarY,
MRings)
   Var_Int = Star_Info(11)  ‘Get intensity of var

‘Get comparison star data
    Star_Info =
MaximDoc2.CalcInformation(CX1, CY1,
MRings)
‘Get intensity of Comparison star
    Comp_Int = Star_Info(11)
‘Calculate magnitude of variable star
    MV = CM - 2.5 * Log10(Var_Int / Comp_Int)
‘return the calculated magnitude
```

To perform photometry, an automation routine needs to measure intensities and magnitude of various stars.

The above routine calculates the magnitude of a star in an image, given the X & Y pixel location of the variable, the X & Y pixel location of a comparison star, and the known magnitude of a single comparison star.

For differential photometry, there are of multiple methods for determining the magnitude of the variable star. Three examples that could be incorporated into an automated photometry routine are:

1) Basic V-C: the magnitude of the variable found by using a single comparison star:

$V = (v - c) + C \quad \{e.g.\ V = 3.7 + 12.5\}$

2) Average = Mean of variable magnitudes found using multiple comparison stars:

$V_i = (v - c)_i + C_i \quad \{e.g.\ V_i = 3.7 + 12.5\}$

Then:

$$V = \left(\sum_{i=1}^{n} V_i\right)/ i$$

{e.g. V = (16.2 + 16.3 + 16.4) / 3 }

3) Aggregate – combining all comparison star intensities and magnitudes to form a virtual star to compare with the variable:

$$C_{(total)} = (-2.5)\mathrm{Log10}\left(\sum_{i=1}^{n} 10^{(-Ci/2.5)}\right)$$

{sum comparison magnitudes}

$$I(total) = \sum_{i=1}^{n} I_i$$

{sum intensities}

Then:

$$V_{agg} = -2.5\ \mathrm{Log10}\ (I_V/I_{(total)}) + C_{(total)}$$

{find var mag}

4. 7. Park and Unpark

Automation routines can be utilized to add features to a telescope, that are not part of it intrinsic command set. An automated park and unpark routine for the Meade LX200 telescope can be implemented using Maxim DL and a telescope driver from the ASCOM modules. This can be implemented as follows using the instantiation of the LX200 interface module declared here as LX200_Int.

```
'------------- Park Command --------------------
'' Description:  This routine parks the LX200 at
the '  local hour angle and a given Dec.
'
' Inputs:
'    Horizon_Dec:
'      integer_Value of Dec to park scope at
' Outputs: None
' Calls:
'   LX200_Int. SiderealTime
'   LX200_Int.SlewToCoordinatesAsynch
'   LX200_Int.Slewing
'   LX200_Int.CommandBlind
'          -------------------------
Private Sub Park_LX200(Horizon_Dec As
Integer)

'storage for RA, DEC, local hour angle
   Dim RA, Dec, LMST  As Variant
'get the current hour angle
     LMST = LX200_Int.SiderealTime
'set the target RA to the hour angle
     RA = LMST
  'set the target Dec to the local horizon
      Dec = Horizon_Dec
'slew the telescope to the target coords
Call LX200_Int.SlewToCoordinatesAsync(RA,
 Dec)
'wait for slew to complete
  Do While LX200_Int.Slewing = True
      DoEvents
   Loop
'Set Land Mode to turn off tracking
   Call LX200_Int.CommandBlind("#:AL#")
Exit Sub
```

```
'------------- Un-Park Command ---------------------
-
' Description:  This routine un parks the LX200
‘and places the telescope at the local hour
‘angle  and zero Dec - the normal start
‘alignment position
'
' Inputs:
'          Horizon_Dec:
‘  integer_Value of Declination telescope was
‘ parked at ‘   (the horizon at a given latitude)
‘
' Outputs: None
' Calls:
‘   LX200_Int. SlewToCoordinatesAsynch
‘   LX200_Int.SlewToCoordinatesAsynch
‘   LX200_Int.Slewing
'                              -------------------------
Private Sub UnPark_LX200(Horizon_Dec As
Integer)
  'storage for RA, DEC, local hour angle
     Dim RA, LMST As Variant
    Dim Dec As Double

  'Set Polar Mode to turn on tracking
      Call LX200_Int.CommandBlind("#:AP#")
  'Get current start up RA = LMST
      RA = LX200_Int.RightAscension
  'Set the target Dec
      Dec = 0#
 'sync the telescope to parked coord.
    Call LX200_Int.SyncToCoordinates(RA,
Horizon_Dec)
  'move scope to target = RA=HA, Dec = 0
    Call
LX200_Int.SlewToCoordinatesAsync(RA, Dec)
  'wait for slew to complete
  Do While LX200_Int.Slewing = True
       DoEvents
  Loop
Exit Sub
```

5. Conclusion

Today’s serious amateur and professional astronomers have a wealth of software automation routines that can greatly simply and speed up the collection of images and processing of data. The various automation programs appear to fall into five categories, according to their abilities and design. These programs are available from commercial vendors, or in some cases they exist as shareware or freeware. With a little bit of experience and skill, an astronomer can also craft his own automation routines using the scripting ability of a program like Maxim DL. Like anything else in Astronomy, all it takes is practice and patience.

6. References

ASCOM (2007). Astronomy Common Object Model. http://ascom-standards.org/

Bartels, M. (2003). ASCOM Talk List. http://ascom-standards.org/BartelsOpEd.html

Bijaoui, A, Marchal, J, Ounnas, C. (1977). *Astron and Astrophys* **65,** 71-75.

Cook, L. (2007). http://www.geocities.com/lcoo/

Maxim DL (2007). http://www.cyanogen.com/products/maxim_extras.htm

IRAF (2007). http://iraf.noao.edu/

CCDWare (2007). http://www.ccdware.com/

CCD Commander (2007). http://ccdcommander.astromatt.com/

Wodaski, R. (2002). *The New CCD Astronomy*, New Astronomy Press.

Vanmunster, T. (2003). Peranso. http://users.skynet.be/fa079980/peranso/index.htm