Version history (PyOTE description is at end)

version 4.2.3

- fixes block integration that was broken by the new lightcurve selection code.

- partially implements time shift of the reference curve for normalization – the visual shift is implemented, but the supporting math is not in place so the normalization is currently unaffected by the reference lightcurve shift; normalization is done with the unshifted reference curve – this is just a necessary step in the code development..

- adds logging/reporting of the curve (if any) used for normalization and the smoothing interval used.

version 4.2.2

- adds a new tab/panel for controlling the display of lightcurves. From the 'help button' on that tab:

  This panel allows up to 10 lightcurves to be displayed at the same time. If the csv file has more than 10 lightcurves, the first 10 are displayed.

  The **target** lightcurve is always drawn with bright blue dots.

  If a lightcurve is selected as a **reference** to be used for normalization, it is always drawn with bright green dots.

  Unless a lightcurve is designated as a **target** (curve to be analyzed for an event) or is designated as a **reference** lightcurve, its dot color depends on the row it is in - every row has a unique color other than blue or green.

  Lightcurves can be displaced up or down using the **Y offset** spinner to control the displacement. This affects the display position only; the underlying values are not affected. This facility was added to allow the separation of lightcurves that would otherwise overlap in a confusing manner.

  There can only be one lightcurve selected as **target**.

  There can be either 0 or 1 lightcurve selected as a **reference** for normalization.

  Normalization is applied whenever the **normalization reference curve smoothing interval** spinbox is changed from 0. Whenever this number is changed, a new normalization will result. If this number is returned to zero, all normalization is removed and the original values restored.

  The **X offset** spinbox is not yet implemented. Its (future) purpose is to allow the **reference** curve used for normalization to be 'time shifted' for those cases where a drifting cloud affects lightcurves at slightly different times.

- the normalization itself has been simplified so that no user input is needed other than the number of readings to be used in the smoothing procedure. The smoothing procedure is a double application of a first order Savitzky-Golay filter (a straight line) to the points included with extrapolation.

- added (back again) a spinbox on the Setting/Misc. Tab that allows the dot size used in lightcurve plots to be changed.

version 4.2.1
- fixed final report showing NE3 stuff even when NE3 not in use was checked.

version 4.2.0
- improves annotation on **Detectability** plots by including the magDrop information.

- eliminates the display of negative drops in the False-Positive histogram

version 4.1.9
- fixes (hopefully) Win 11 issue involving the directory separator character \
  Win 10 accepts / as a separator – Win 11 apparently does not (unless there is some setting that will persuade Win 11 to accept either separator.

version 4.1.8
- adds option to write sample light-curve from detectability analysis to a csv file that be imported to PyOTE.

version 4.1.7
- adds the requirement to specify an observation duration when doing a detectability test

- during a detectability test, if a detectable event was found, a sample light-curve showing such an event is plotted, otherwise the normal False-Positive plot will be shown.

version 4.1.6
- reenables Cholesky failure messages (disabled for testing)

- Adds test to detectability routine so that the user cannot give an event duration that requires more points than are available in the observation.

version 4.1.5
- changed error bar calculation reporting so that the the value reported in the containment interval report matches that reported in the Excel final report when there is no asymmetry present.

version 4.1.4
- This version adds tools needed to make best use of the Night Eagle 3 camera, the successor to the Night Eagle Astro camera (which is no longer in production). They are all grouped on a new tab page titled Night Eagle 3.

- The Night Eagle 3 is a rolling shutter CMOS camera. As a result, with this camera, the timing of an occultation depends on which row the occulted star is at when the occultation occurs. PyOTE will automatically calculate the needed time correction from the y (row) position that you enter in the spin/entry box at the bottom of this tab.

- For the Night Eagle 3, times extracted by PyOTE require only a VTI correction (if you use an IOTA VTI, there is no correction at all needed).

- Note: if the recording was **not** made with a tracking telescope (so the target star is moving across the image), you will need to watch the video and record the row of the occultation for use in this program. Normally, with a tracking telescope, you will be able to use the y position of the 'target' aperture directly. PyMovie includes this information in the comment lines of the csv file.

- The Night Eagle 3 has a very effective noise reduction system called DNR (Dynamic Noise Reduction). There are 4 levels of noise reduction and you will need to indicate on this tab page what DNR setting your recording was made with.

- The noise reduction is not without a small cost however - the edges of an occultation light-curve will no longer be a step change. Instead, the edges will follow an exponential curve, approaching maximum and minimum intensities asymptotically. PyOTE has the ability to fit such an exponential curve to the D and R transitions, so the time resolution will be restored during the least squares fit and you should feel free to use whatever level of noise reduction you may need. It is recommended that you run your NE3 at a gamma of 0.75 (1.0 gamma is not available) AND that you use PyMovie to linearize the recording (i.e., invert the 0.75 gamma curve of the camera).

- The D and R exponential curves each have their own time constant, measured in frames, that control the frame rate of exponential curve growth. The default values provided are usually enough to provide the starting point for a good fit. This starting point is used by PyOTE during a least-squares driven search for better time constant values. If the starting point given is too far from 'correct', the least-squares search may settle in a local minimum that produces a fit that is visually bad. In this case, change the starting value to something closer to 'correct' and try again.

- When the exponential curve fit algorithm is in use, the light-curve plot PyOTE displays will be changed. Gone is the blue 'camera response' curve, replaced by brown dotted 'theoretical' exponential edge curves that you can use to judge for yourself the goodness of a 'fit'.

- You will probably want to use the most aggressive noise reduction in most cases, but make sure that your expected event duration is long enough that the D transition exponential curve has time to settle to the bottom event plus some time to allow a good determination of the event bottom intensity. At DNR:HIGH, your expected event time should be greater than 30 frames (1 second) to use this setting.

- I suggest the following rules-of-thumb: can use DNR:HIGH for events 2 seconds and

longer; can use DNR:MIDDLE for events 1 second and longer; can use DNR:LOW for events 0.5 seconds and longer.

version 4.1.2

- fixes bug when D only search has been selected

version 4.1.1

- fixes bug where signal columns beyond four were being read as duplicates of column 5.

version 4.1.0

- adds error bars to reported magDrop values.

  This was an interesting exercise in the propagation of errors because the magDrop calculation involves ln(A/B) where we know the error bars for A and B but have to propagate those errors through the ratio A/B to get its error bars and then through the ln() function to get the final error bars. The equations I use were verified through simulations.

version 4.0.9

- fixes issue where PyOTE could not read files that it had written.

version 4.0.8

- fixes bug (introduced by a misspelling in version 4.0.7) that kept csv files with more than four apertures from runnig.

version 4.0.7

- adds the ability to select the PyMovie data column type to be analyzed. There are two main column types that are likely to be used: signal, which has background subtracted and appsum, which has no background subtracted – it's the raw mask pixels summed.

  Some others are available: avgbkd (shows the average pixel value in the aperture); stdbkg (shows the pixel noise in the background pixels in the aperture); nmaskpx (which shows the nuber of pixels in the sampling mask – there are times when this curve can be used to extrat event times).

  Note: this only applies to csv files generated by PyMovie.

version 4.0.6

- minor GUI changes

version 4.0.5

- if you click outside the light-curve, the closest point will be toggled (so you don't have to zoom in to select the leftmost or rightmost point in a light-curve.

- When baseline statistics are calculated, they are reported as baseline mean and baseline snr (instead of just snr, which is the term we use when a drop is present).

version 4.0.4

- a few GUI changes

- removed the Use as secondary check box. Now if secondary curve is selected (spinner is non-zero), it will be used automatically.

version 4.0.3

- implements a major GUI change: uses tabbed folders to group buttons and other controls.

- Removes radio buttons that indicated a D and R solution wanted, or a D-only solution, or an R-only solution. The type of solution is now inferred from D and R region setting.

  When min/max event size is used as the search criteria, then a D and R solution is assumed and found. This type of search criteria does not make sense for a D-only or an R-only solution. Use a D region or R region selection for these cases

- The 'selected curve to analyze' number and the 'curve to normalize' number are now allowed to be the same – no funny jumping around.

  It is even possible to normalize a curve against itself. There may be a use case where this is useful, so I didn't lock it out.

version 4.0.2

- removed a debug print statement that was inadvertently left in. It is responsible for the “mouse event” messages polluting the log file.

version 4.0.1

- increased width of detectibility plots to (hopefully) deal with Windows tendency to truncate the plots. If this doesn't work, then this will be a permanent feature (not a bug) for Win10 installations.

version 4.0.0

- detectibility graphics plots are now being being put in their own folder. The root folder is that of the directory where the light-curve came from. The plots will be found in lightCurveDirectory\DetectibilityPlots\

  In addition, if you supply a duration step size, which is the way to ask the detectibilty calculator to step from the given duration down until an event of that duration fails the False-positive test, only the final plot (where False-positive became non-zero) will be plotted.

  All of the above is just to cut down on clutter in the light-curve directory and to only plot meaningful graphics generated during the minimum duration search.

version 3.9.9

- whew! 3.9.8 did solve the sizing issue. So it is safe to get back in the water so...

This version just improves the right-click context help for buttons in the new Detectibility calculator.

version 3.9.8

- one more attempt to solve the gui sizing issue.

- 'sticky' settings are back but are now stored in pyote.ini instead of the obscure simple-ote.ini

version 3.9.7

- Something inexplicable has happened regarding gui size. Suddenly, the gui size is too large for many people – but I have set the gui size smaller than it has ever been. So, best guess is that an overly large gui size got distributed once (there was a strange event where an extremely large gui size appeared) and is being 'remembered' in the simple-ote-ini file that preserves the 'sticky' values like size and position of the gui and a few other things.

  This version refuses to read any existing .ini files. Hopefully you will be able to run the program, but nothing will be 'sticky'.

  If this 'cures' the problem, I can restore the use of the .ini file and we can get back the sticky stuff.

  This version also removes the 'scrunch' of the lefthand button panel.

version 3.9.6

- another attempt fix the startup gui size issue on smaller legacy monitors. This version has a design size of 1900x1000. It may be necessary to delete the simple-ote.ini file that 'remembers' size and position settings. It is located in whatever directory PyOTE starts up in.

version 3.9.5

- adds automation to the 'detectibility' calculator. If a duration step size is entered, a series of 'detectibility' calculations will be made at decreasing durations until the false-positive probability becomes non-zero. A unique .png (incorporates duration and magDrop in the file name) will be written to the directory of the light-curve and the final plot where detection failed will be left on-screen.

  If the duration step size is left unfilled or is zero, a single 'detectibility' calculation will be made as before.

version 3.9.4

- attempts to fix over-size initial gui.

version 3.9.3

- makes various improvements to the 'detectability' calculator to make it easier to use.

version 3.9.2

- adds a 'detectability' calculator for use in pre-planning event observation.

- 'scrunched' the left panel buttons together to make more room for the user to move the horizontal splitter up (which gives more room to view the lower right panel where the report gets printed).

  I think that this is kind of ugly, and it can be easily returned to the more spacious format if users would prefer the old look over more view space for the report.

version 3.9.1

- adds an snr calculation to the **Calc baseline stats** button.

version 3.9.0

- restored the right-click help to the **Mark baseline region** | **Clear baseline region** | **Calc baseline mean/sigma/corr coeffs** buttons that got lost during the scrollable gui experiment.

version 3.8.9

- Changed the label on the **Write csv file** button to **Save current light-curve** to more intuitively suggest what is hiding behind this button.

version 3.8.8

- fixed a bug that sometimes kept events that contained a single point at the bottom from producing a report.

version 3.8.7

- changed the way curve to analyze and normalization curve are selected to make it less confusing. Now, if there is a conflict (same curve selected for analysis and normalization) the other one is set to 0 (a new value) to 'get it out of the way'. It may still be a little bit confusing when you need to crossover, but if you set the higher numbered curve first, it will be easy.

version 3.8.6

- checks for only a single light-curve in the csv file which was causing a 'list index out of range' error message

version 3.8.5

- changed the way QGridLayout was referenced.

version 3.8.4

- changed the way QTableWidgetItem and QApplication were referenced to accommodate the latest Anaconda version, which has reorganized where those modules are stored. A similar change was made in version 3.7.6 and I do not know how this has gone uncorrected for so long – I hope that I have not missed something.

version 3.8.3

- adds display of light-curve name (if available in the csv file) to the curve-to-be-analyzed

and the normalization curve next to their selection spinners.

version 3.8.2

- 3.7.5 introduced a scrollable GUI for small screens. Unfortunately, that caused resizing problem for all users. The version reverts back to the original GUI design.

- Added a print to the command window of the version number for diagnostic purposes. It will help clarify the situation when an installation has somehow managed to have multiple locations for PyOTE storage as to exactly what version is running.

version 3.8.1

- adds a checkbox to use reading number to annotate the x-axis, even when timestamps are available to use for this annotation. The setting is 'sticky', so your last preference will be remembered and used in the next run.

version 3.8.0

- adds a tutorial button to supply a quick-start for new or infrequent users.

version 3.7.9

- adds a tool that allows a user to manually specify which baseline points to use for the calculation of B (baseline mean) baseline noise (sigmaB) and noise correlation coefficients. This was added primarily to support GPS-flash-tagged light-curves where the flash itself adversely affects the baseline values. With this new tool, the regions of the baseline that are outside the 'flash-zone' can be specified. Look for a button labelled **Mark baseline region** located near the bottom-left of the GUI panel.

  This tool may also be useful for wind-gust situations, headlights from passing cars, bumping the telescope, etc. In the past, such accidents might have required extensive 'clipping' of the light-curve to avoid the affected areas, with a consequent loss of information. Hopefully this tool will alleviate those situations.

version 3.7.8

- added visible left, top, and bottom axes to main plot (with ticks to make easier to associate timestamp with point).

version 3.7.7

- if timestamps are available in the csv file, they are used as x axis labels in the main plot.

version 3.7.6

- changed the way QMainWindow and Qapplication were referenced to accommodate the latest Anaconda version, which has reorganized where those modules are stored. This was causing a fatal startup error.

version 3.7.5

- changes the routine that looks for the latest version of PyOTE to one provided by Kia Getrost. His version contacts the PyPI repository via a json query and is the officially supported way to get version info. My version was based on a 'hack' that depended on

a special feature of pip (the loader that get programs from the PyPI repository) that was marked as 'unsupported'. That worked for many years until the pip programmers removed the 'special feature' as was their right (and the implicit promise/warning, I guess). This caused several users to always get a message that they were not running the most recent version of PyOTE, even though they were. Again: thanks to Kia Getrost for researching the problem and even supplying correct code (worked first time!) for me to use.

- The GUI has been changed so that scroll bars will appear when the standard GUI size exceeds your screen dimensions. THIS IS EXPERIMENTAL BECAUSE I DO NOT HAVE SUCH A SMALL SCREEN TO TEST WITH. If this does not work for you, please let me know immediately!

version 3.7.4

- fixed issue with ‘save main plot’ that caused the graphic to sometimes be written to a previous folder rather than the current folder containing the csv file (FYI: it was a ‘feature’ of the routine that saved this information that, in the interest of efficiency, it would sometimes not write a new folder name to the ‘sticky file’ until later – the fix was to force it to update the ‘sticky file’ on every change)

version 3.7.3

- adds code line needed by any Mac users that are running Big Sur

version 3.7.2

- fixes bug that caused all analysis attempts to 'stall'

version 3.7.1

- fixed a bug in the r limb angle entry

version 3.7.0

- adds right-click help to ast speed label giving the equation to use if asteroid speed is not available but asteroid diameter and maximum duration are specified

- adds suggestion to right-click help on penumbral fit checkbox for how to find the penumbral curve csv file that is provided for practice purposes.

version 3.6.8

- adds modeling of off centerline observations to lightcurve calibration curve generation

version 3.6.7

- clarified the location of the Enable Manual Timestamp Entry checkbox in the pop-up message appears when there are no timestamps in the csv file.

version 3.6.6

- modified the lightcurve demo to show more clearly the camera exposure function and the star intensity function that are convolved with the diffraction

lightcurve to produce the lightcurve as seen by the camera.

The right-click help connected to the Demo button has been expanded to include a discussion of the convolution operation and hopefully provide some context to aid in understanding the star and camera function plots.

version 3.6.5

- added advisory message when False Positive probability plot appears in hopes of stemming in-discriminant use of this number as a 'decider' between a 'miss' and a 'positive' for an observation.

version 3.6.4

- version 3.6.2 would open .xlsx Report file on Mac, but not Windows. This version attempts to fix that (Windows needs a different command to open a file).

version 3.6.2

- In Excel report:

  … at end of filling, I call the OS to open the file. Requires correct association of .xlsx file type with Excel or LibreOffice, or whatever you use to examine spreadsheets

  … now writing numbers into numeric cells rather than text. This allows the cell formatting to control rounding, etc

  … if I can't write to the selected file, I ask whether you might have it open somewhere else already

version 3.6.1

- reduced number of digits in the error bar numbers to 2 written to the Asteroid Occultation Report Form so that the resulting number fits within the allotted cell size. (I'm told that Occult only uses 2 digits anyway.) I also updated the context help associated with the … fill Excel report button

version 3.6.0

- provides the ability to fill entries in the Excel spreadsheet Asteroid Occultation Report Form from PyOTE results, thus eliminating transcription errors.

  A button to allow the user to activate this 'fill' has been added just to the right of

  the … write report button

  NOTE: the normal .xls report form that OccultWatcher creates and prefills (when requested) during your report to OccultWatcher ...

… MUST BE CONVERTED to .xlsx for use by PyOTE!

… For Windows users, Excel will read an .xls file and save it as .xlsx
… For Mac/Linux users, LibreOffice will read an .xls file and save it as .xlsx

Sorry about this extra step, but it was necessary. The downstream tools used by area coordinators work equally well with the .xls and .xlsx forms of the spreadsheet, so there is no problem sending in the .xlsx version.

What gets filled in for you is:

...D/R uncorrected times
...D/R error bars
...SNR
…OTA used
...nominal magDrop (entered in Comments cell of the spreadsheet).

NOTE: you must still open the spreadsheet and enter the exposure setting used for your camera; it was not possible to do this from PyOTE.

After filling in the exposure setting in the spreadsheet, double check the form but it is likely that it is ready for submission.

version 3.5.9

- fixed another bug in penumbral curve fit and removed diagnostic printouts.

version 3.5.8

- fixed a number of bugs in the penumbral curve fit code

version 3.5.7

- automatically turns off the display of the normalizing lightcurve when a normalization is performed. It was a source of confusion to leave the normalizing lightcurve visible because it was sometimes the case that the normalization appeared not to have occurred (when the normalization effect was subtle/minor).

version 3.5.6

- fixed a bug that kept a user from selecting a new file to read if PyOTE had been started from PyMovie. Previously, it had only reopened the same file instead of giving the user a file select dialog to choose from.

version 3.5.5

- fixed a bug that kept occultations from being extracted from lightcurves 5 and up. The lightcurves above 4 could be viewed --- they just couldn't be processed through the event finder because they were a different type (coding error)

version 3.5.4

- made the 'get newest version' code identical to that in PyMovie in hopes that that will resolve the issue that some people experience with pip (or python) installing downloaded pyote in a directory where it subsequently cannot be discovered. The change is minute, so I'm not optimistic, but it's worth a shot.

version 3.5.3

- removed the blank lines between header lines extracted from the csv file and placed in the log file --- this makes it easier to look at the newly added aperture settings (so just a tiny cosmetic change).

version 3.5.2

- Changed the 'smooth reference star procedure' to no longer display the points at the left and right ends; such points are actually extrapolated points with all the hazards that extrapolation can engender. Smoothing functions that use sliding windows always have a problem at either the left edge, the right edge, or both (when a symmetrical smoothing algorithm is employed). They run out of points and have to extrapolate/fake a number of points equal to the window size. Such extrapolated points can exhibit extreme behavior, zooming up or down unexpectedly and unrealistically.

  Previously PyOTE treated this as a cosmetic problem and relied on the observer to be aware of the end point effects and ignore them. But that puts a burden on the user to be well informed about what's going on. As one of the goals of PyOTE is to enable infrequent/inexperienced users to get dependable results without requiring in-depth understanding of the internal workings of the program, we have decided to make the end-point smoothing issue very apparent by doing an automatic 'trim' of the points affected by extrapolation.

version 3.5.1

- Added additional references to the North American Excel Spreadsheet report in the new section of the final report that bangs on about the start-of-exposure timing convention.

version 3.5.0

- When PyMovie files are read, the aperture names are now being used in the data table (lower left panel) as column headings and used during the 'write csv' process. This makes the format of the PyOTE csv file match the PyMovie format so that AOTA can read both PyMovie csv files AND PyOTE csv files.

- NEW: when 'trims' have been placed, a 'write csv' process will honor those values and produce a 'trimmed' output file.

- NEW: when a light curve has been 'normalized', the changed values are written to the data table where, once again, a 'write csv' process will capture the results.

- Added additional reminders that the start-of-exposure timestamp/timing convention is employed.

version 3.4.9

- Added some explanatory language to the “Excel report” section regarding the proper interpretation of the 'false positive probability' number.

version 3.4.8

- This version deals more realistically with high magDrop lightcurves by defining a 'limiting magDrop' as:

  limMagDrop = 2.5 * log10(B / std(A))

  std(A) is the noise level of A.
  B = average baseline intensity
  A = average event intensity

  Normally, we report/calculate magDrop = 2.5 * log10(B / A), but this calculation becomes increasingly unreliable as the value for A gets very small. And when A is noisy, it is even possible to statistically have A become negative for large mag drop lightcurves. This happens more and more as A approaches and then becomes smaller than std(A). For example, if A happened to be equal to std(A), the normal distribution of A values tells us that 84% of possible A values are > 0 and so can be used in the regular magDrop equation. The other 16% of the time we can only report that the calculation could not be performed.

  The above observation suggests that reliable estimates of magDrop require that A be greater than std(A) --- that is the ad hoc reason that we have defined limMagDrop as we have.

  This value is substituted for a calculated magDrop whenever A is less than std(A), i.e., whenever A is at or below std(A).

  limMagDrop values are reported with a leading > symbol to signify that that value is a limMagDrop value. They are easy to spot in the report.

version 3.4.7

- Automatically loads the correct version of Adv2

version 3.4.6

- Adds AAV Version 2 file as a type that can be read (important when Do OCR check is enabled)

version 3.4.5

- Fixes block integration which was failing when more than 4 lightcurves were being processed.

- Made use diff and Do OCR checkboxes sticky.

version 3.4.4

- PyMovie files can have lightcurves extracted from more than 4 apertures (with

user supplied names). This version allows all lightcurves from PyMovie files to be read and made available for processing. Prior to this change, only the first 4 lightcurves were read.

Note: when you change the lightcurve to be analyzed (with the spinner), the log panel will show the aperture name for that lightcurve. That happens when the reference lightcurve is changed as well.

These changes are to PyMovie file treatment ONLY.

version 3.4.3

- Fixed bug that required an entry in dist(AU) and speed(km/sec) edit boxes for a solution to be found (the empty entries were causing an uncaught exception).

  The intention is that PyOTE should work as it always did if a user ignores the new lightcurve parameter panel and makes no entries. This 'fix' was required to make that happen.

version 3.4.2

- Cosmetic change again: added a spinner to control line widths in plots so that a user can adjust for the resolution of the screen in use. I design on a 5120x2880 screen and needed lines to be 3 pixel wide to suit my taste. But some users have screens with 1280 horizontal resolution and those same 3 pixels became unsightly fat lines --- now there's a choice.

version 3.4.1

- Some cosmetic changes: thinner vertical thins for edge position and error bars; checkboxes to control whether the underlying lightcurve is plotted, error bars are plotted, or edges are plotted --- a cleaner plot is the major result and you have better control over the 'look'

- Added a checkbox to disable the automatic display of D and R frames from the video for OCR quality control checks. When there are no concerns about OCR reliability (true for me nearly all the time), it can be annoying to have to close the frames all the time.

- The BIG change is the addition of a penumbral curve fit procedure. It's a bit fiddly, so I included a test lightcurve with the download. I can't give you a specific location for the file because it depends on details of your particular installation. Find where it is by searching for example-penumbral.csv When you find it, copy or move it to some other folder because if you process it where it resides, other files will get added in your installation directory --- we really don't want extraneous non-program files floating around in your installation directory.

  To learn how to use the new procedure (which is a bit 'fiddly'), right-click on the penumbral fit checkbox --- be patient; play around.

version 3.4.0

- Adds the ability to specify a diffraction lightcurve for use in timing the event. A new panel with edit boxes for asteroid/occulting body distance (in AU --- astronomical units) and asteroid/shadow speed (in km per second) has been added. These values are needed in order to calculate a diffraction light curve.

  In addition to modeling diffraction effects, one can add the effect of a finite star disk to produce a penumbral curve. NOTE: PyOTE does not yet have the ability to correctly analyze a penumbral curve where it takes more than 1 or 2 readings for the transition. That project is under way and will be in the next version.

version 3.3.9

- Automatically installs cv2 if not already present. This package is needed for the new frame view feature.

version 3.3.8

- If the video referenced in the csv file can be found, there is now an automatic display of the D and R frames relevant to calculating correct D and R times so that the user can verify that timestamp OCR extracted the correct timestamp values.

version 3.3.7

- A new button (View frame) with an associated spinner for entry of a frame number has been added:

Use this button to view a frame from the video that was used by PyMovie or Limovie to prepare the .csv file that is currently being analyzed. .avi and .ser files are viewable in this manner as well as .fits files inside a FITS folder.

If this button is disabled, it is because the .csv file did not come from PyMovie or Limovie or simply cannot be found/opened.

This feature can/should be used as a final quality control check for a video that contains timestamps extracted using OCR. It is possible for OCR to fail in manner that is **not detected** by PyOTE because the program only verifies that there is a consistent step (delta time) between frames. If a high order digit in the timestamp has been consistently misread, substituting a 8 for a 9 in the minutes field for example, the steps can be consistent while the reported time of the event will be seriously in error.

ALL time reporting is derived from the timestamp(s) associated with D and/or R (the integer values, not the sub-frame values). If those timestamps are correct, the reported times will be correct even when there may be a few missing or duplicated frames. So best practice is to enter the D frame value in the spin box and visually confirm that the timestamp that you can see is the same as that extracted by the OCR procedure. Repeat for R.

Another use for this feature is to handle the case where there is a visual timestamp, but either OCR was not activated during the .csv preparation, or the timestamp overlay came from an unsupported VTI type. The workflow would be to let PyOTE find the D

and R frame values, but before pressing **... write report**, do a **Manual timestamp entry** for the D and R frame entries found by viewing the relevant frames and entering the correct times in the Manual timestamp dialog.

It should be noted that the manual timestamp entry can be performed even when timestamps were already present in the file --- your manual entries will cause all the timestamps to be recalculated.

version 3.3.5:

- Changed usage of max([a, b ,c]) to max(a, b, c) to see if this allows the Numba JIT compiler to work for one user that found version 3.3.4 failed to load/compile.

  This should have no effect on users that already have version 3.3.4 working.

version 3.3.4:

- To shorten the time to find 'solutions', I used the Numba JIT (just-in-time) platform independent compiler that produces machine code from Python byte-code. You may notice a very slight increase in the time to start-up PyOTE because I do those compile operations while PyOTE is being loaded.

version 3.3.2:

- Adds a false-positive probability calculation and printout in the final report. This number is the fraction of 'false drops' found during the 50,000 tests that are greater than or equal to the drop value extracted from the actual observation.

version 3.3.1:

- Adds a 'false positive' detection to the final report. A new plot has been added to the error bar plot. It shows the distribution of drop sizes (B-A) for an event of the size (duration) extracted from the actual observation, but with only correlated noise in the sample (the number of points in this sample is equal to the number of points used in the lightcurve extraction). 50,000 attempts are made to find the deepest event that appears (falsely) when there is only noise being analyzed. If the drop from the actual observation is greater than the maximum size of a 'false drop', we have some assurance that the event extracted from the actual observation did not happen 'by chance'.

version 3.2.9

- Changed main plot so that the scroll wheel only zooms the x axis.
- Changed lightcurve plot so that it conforms properly to 'start-of-exposure'.

version 3.2.8

- Changed font size in help files --- it was fine for Mac but too big for Win10

version 3.2.7

- Removed the 'hover-for-help' and replaced it with a 'right-click-on-item' to get help. This scheme was introduced in PyMovie and I found it easier to use than the 'hover' scheme. In practice, the 'hover' popped up when it was not needed,

so most users eventually disabled it. As a result, it became so tedious to look at help --- enable hover; hover; read; disable hover --- that the help system was used less and less. The right-click-for-help is always available and easily invoked --- hopefully this will encourage more frequent reference to it.

version 3.2.6

- This is a 'cosmetic' release --- there should be NO detectable differences from version 3.2.5 in terms of functionality.
- All python files were brought into compliance with PEP 8 coding standards. Only I care about that.
- More significantly, I removed the dependency on C code by using Numba as a code accelerator instead of Cython. As a result, I no longer need to compile separate code versions for Mac, Windows, and Linux. That makes my life easier, but you should experience no operational changes.
- All this 'cosmetic' work is in preparation for working on PyOTE issues again.

version 3.2.5

- Added special test for Tangra files to detect the empty fields (which MUST be fixed) that Tangra outputs whenever it has trouble extracted a value from an aperture. It prints a message and stops all further processing, forcing the user to attend to and deal with the missing values.

version 3.2.4

- Modified the test for newer version to accommodate the different strings returned by pip 18.1 and pip 19.0+
- Added ability to invoke PyOTE from PyMovie with an externally supplied csv file that is automatically opened.

version 3.2.3

- fixed a long overlooked bug in the loading of the data table (at lower left corner of GUI): when there are four lightcurves, LC4 was set in the table from LC3 (i.e., LC3 == LC4 whenever there was an actual LC4. It was correct in the lightcurves themselves, so no observation analyses have been affected by this bug. It was cosmetic only.

- Added support for the PyMovie csv format

version 3.2.1

- this version makes PyOTE more robust to a common 'cockpit error' that users have been making with Tangra files. Specifically, if a Tangra csv file is opened in a spreadsheet program, then saved from that spreadsheet program, the original csv file gets modified and overwritten by the addition of empty fields at every row sufficient to match the number of columns in the longest header/comment row --- the spreadsheet program did this to satisfy its internal requirement that every row have an equal number of columns. The result is superfluous commas at the end of data lines (that Tangra did NOT put there). Until this version, that 'butchered' file could not be read. This version adds code to parse data lines only up to the first non-empty column. Hopefully this will not

have ramifications in the future (like a format change that has empty fields followed by non-empty fields --- not a likely expectation).

version 3.2.0

- Changed GUI to better align text on min max edit boxes to avoid confusion.

version 3.1.9

- Fixed a bug in the test for a min/max solution search being constrained by a too large min value.

version 3.1.8

- version 3.1.7 was released without an updated version history. Here is what was changed in 3.1.7:

- Added the ability to write the data table that is displayed in the lower left corner of the GUI out as a csv file. Now, if timestamps and block integration operations are performed on the input file, those results can be preserved in a csv file.

  A 'file save' dialog is provided should you wish to change the default name and location of the resulting file. The default name is that of the input file with the text .PYOTE inserted to the left of the .csv extension. The default location is the directory of the input file. It is recommended that you accept these defaults unless you have compelling reasons to do otherwise.

version 3.1.6:

- Values entered in the Manual Timestamp Entry dialog box are now 'sticky', thus making corrections easy to do without requiring re-entry of all data.

  Also trapped is the case where a user has entered a custom frame time but failed to click the radio button indicating that it is to be used.

version 3.1.5:

- Added additional tests of candidate solutions against a straight line so that there should always be agreement between a solution found by a min/max event size search and a marked D and R region search of the same area.

  Previously it was possible for the min/max search, which searches the entire light curve, to be tripped up by what we call a 'competitor'. A 'competitor' is an 'event' with good statistics. However, that 'competitor' may have a small magDrop and so later be rejected when we compare with a straight line solution. That 'competitor' would thus mask an event with slightly worse statistics but a larger magDrop. The change was to test every candidate against a straight line during the search. This does make the search time longer, but not too much longer.

version 3.1.4:

- Fixed error in new dropped reading detection logic when light curve was processed in field mode.

- Cleaned up some language in tooltips.

version 3.1.3:

- Expanded manual timestamp preset time deltas to include NTSC and PAL field times. Also added ability to evaluate numeric expressions entered in the 'Custom time' box: now you can type 1.001/60.0 in that box if you wish.

- Eliminated the 'entry num' column in the data matrix at the lower left of the GUI. The 'entry num' is unused and a possible source of confusion with the frame or field number for the unwary.

- Added **all** the light curves read from the input file to the data matrix display. Previously, only the first light curve values were displayed. This is done in anticipation of adding a 'write csv' button to memorialize the result of a manual timestamp entry.

version 3.1.2:

- Added a test for possible dropped frames identical to that done in R-OTE when manual timestamp is utilized. The test is to calculate the expected number of frames based on standard NTSC/PAL frame times and compare that number with the count of frames enclosed by the early and late timestamps. If there is a mismatch of more than 0.12 frames, a warning is popped up and a log entry made. It is possible to use a 'custom' frame time if your camera differs from either of those standards.

version 3.1.1:

- A convenient way to search for a 'solution' is to set a min and max event size rather than mark D and R regions. This is particularly useful in low snr situations where the D and R edges may be quite diffuse. However, if one sets the min event too large or the max event too small, the resulting 'solution' will be artificially constrained and thus be wrong. This situation is now detected and a log entry as well as a pop-up alert will tell the user to change the limits and try again.

- Three magDrop values are now calculated for each confidence level: the largest magDrop calculated using B + err(B) along with A – err(A); the nominal magDrop calculated using B and A; the minimum magDrop calculated using B – err(b) along with A + err(A)

- The labels on the Find Event button and the Calc Err Bar button were changed to more clearly suggest that after finding an 'event', one should then press the 'report' button to the right in order to complete the process.

version 3.1.0:

- Added a Mac version of a pyote startup file. It is automatically placed on the Desktop the first time pyote is run. Double-clicking on that Desktop file icon will start pyote thereafter.

version 3.0.8:

- Added a Windows batch file to the distribution that, when executed, will startup pyote. The file is called PYOTE.bat and is automatically copied to C:\Anaconda3 (if it is not already there) when pyote is first run. Now, to create a clickable desktop icon for starting up pyote, a user need only go to the C:\Anaconda3 directory, locate the PYOTE.bat file, create a shortcut to it, and drag the shortcut to the desktop. Remember, that file does not appear until the first run of pyote.

  The 'skipped' version numbers were caused by the need for repeated testing of this new feature, each test requiring a new version, even though functionality did not change,

version 3.0.1:

- Restored the vertical splitter between the command/plot area and the table/report area. Somewhere along the line this capability was accidentally removed, and the lack of the splitter was not noticed. Now it's back.

version 3.0.0:

- No code changes. This version is the same as 2.1.6 except that it is built on python 3.7. The previous versions used python 3.6. This allows new users to install the latest Anaconda3 version (which installs python 3.7) without fiddling with archived Anaconda3 versions.

version 2.1.6:

- We now disable the Accept integration button on the first left click in the light curve. As such a click removes the color bars that result from the automatic block integration analysis, it seems intuitive to disable the Accept integration button at that time as well.

version 2.1.5:

- Disable the Accept integration button when user overrides an automatic block analysis with a manual block selection followed by a click on the Block integrate button.

version 2.1.4:

- Corrected a bug that kept manual selection of block integration from being performed after a refusal to accept the automatic block analysis results.

version 2.1.3:

- A minor change to how color bars are plotted when the automatic block integration feature is employed. The edges now appear between data points so the bands are easier to see, particularly for 2 point block sizes.

version 2.1.2:

- To ease usage of the automatic block integration feature, accepting the automatically determined block integration parameters no longer uses a modal

query box, which interfered with the ability to explore/expand the light curve plot. Now there is separate button which gets enabled after an automatic block integration completes.

version 2.1.1

- Added progress bar tracking of block integration analysis because it can take an extended amount of time to complete the analysis when the light curve has many points.

version 2.1.0

- Added automatic determination of 'correct' block size and offset for block integration when user clicks ***Block integrate*** button without selecting the two points normally required to specify integration block beginning and end. The user can choose to accept or reject **pyote**'s opinion of the correct parameters to use when the automation determination is invoked.

version 2.0.9

- Made the selection of Tooltip display 'sticky'
- Duration calculation when D and R span midnight now handled correctly

version 2.0.8

- toolTips changed to invoke and display in a custom dialog box that can be moved and resized to better accommodate legacy displays
- Calc flash timing calculation fixed to properly deal with the non-integer frame numbers that can result from field processed csv files
- Flash timing has been verified to work with integrated light curves
- Made block integration 'sticky' in that a 'Start over' no longer undoes a previous block integration. As a result, once block integration has been performed after a file read, it cannot be done again; a reread of the original file is now required.

version 2.0.7

This version provides several features to ease the processing of light curves that are timed with LED flashes from iPhones (John Grismore's AstroFlashTimer) or Android phones (Eric Couto's Occult Flash) rather than VTI timestamped files

- Adds a button to calculate the edge position of an LED timing flash.
- Adds a checkbox to enable/disable the tooltip messages that appear when a control is hovered over. Tooltip display defaults to 'enabled' because tooltips are an important aid for guiding users initially. Later, when such help is no longer needed, the user can turn them off (they are annoying when you don't need them).
- Adds the ability to select which light curve is to be analyzed. Previous versions would only analyze the first light curve for D and R events. This flexibility is useful in general, but was particularly needed to support LED flash timing.
- Adds a checkbox to force manual entry of timestamp info. This is useful when OCR on a VTI timed light curve has catastrophic errors. It is always employed when using LED flash timing.
- During the error bar calculation, it is possible for the Cholesky decomposition

needed for treating correlated noise to fail. Previous versions treated this as a fatal error and would not produce a final report. This version instead treats the noise as uncorrelated and continues processing to produce a final report.

version 2.0.6

- Added additional instruction in the popup that appears when no timestamps are found in the csv file. This will give casual users additional guidance and clarification for the manual timestamp entry process.

version 2.0.5

- files generated by pyote now contain PYOTE in the filename.

- Timestamps can be corrupted to the point that a timeDelta of 0.0 can result. This version traps that event and reports it clearly --- 2.0.4 failed silently with a divide by zero exception

version 2.0.4

- improves the handling of errors during the reading of Tangra files by showing the offending line in the report panel. Tangra, if it has a tracking problem (i.e., loses it) will emit an empty field for that measurement, leaving it up to the user to decide how to fill in the missing value. Prior pyote versions simply reported 'format error' without providing a printout of the offending line. This version fixes that.

version 2.0.3

- detects and handles situations in which fewer than 14 baseline points are available for calculation of correlated noise coefficients. When fewer than 14 points are available, the correlation coefficients are set to: [1, 0, …] (i.e., coefficients are set to 'no correlated noise')

version 2.0.2

- Note: this version has many significant changes. If you lose confidence in this version, remember that you can always go back to version 1.47 by typing ---

  pip install pyote==1.47

  in an Anaconda console. (Be sure to use double == signs in the command.)

- improved handling of D and R region selection so that one cannot enter an invalid configuration --- automatic corrections/changes are applied.

- incorporates a new 'solver' that no longer requires an initial estimation of baseline noise. This 'solver' is also much faster. With this 'solver', the two-pass modification added in version 1.46 is no longer needed.

- removes unneeded 'analyze noise' buttons and rearranged other buttons to be in-line rather than one above the other to allow the vertical splitter between the plot area and report area more room to change (a help to those using screens

with relatively low pixel densities).

version 1.47

- adds bold red highlighting to message:

! There is something wrong with timestamps at D and/or R or frames have been dropped !

so that it is harder to miss.

version 1.46

- adds automatic recalculation of baseline and event noise parameters utilizing all available data points during a second solution pass; this removes the variability in calculated error bars due to user selection of a necessarily less complete set of data points for noise analysis during the first solution pass.

- adds bold blue text in the 'Excel' portion of the final report to indicate whether or not the light curve was block integrated, trimmed, or normalized. Failing to block integrate a light curve that needed it is a common error. Highlighting the presence or absence of block integration in the most looked at portion of the final report will hopefully help reduce the number of such errors.

Version 1.45

- the initial fully functional release of pyote.

Introduction to *pyote*

Bob Anderson (bob.anderson.ok@gmail.com)

pyote is an occultation timing extraction utility program written primarily in python and distributed through PyPI (the python package repository).

This program is specifically designed for those who will use such a program infrequently; it has been designed to the best of my ability to produce consistent results in the hands of both infrequent and frequent users --- the same results should be obtained no matter who processed the data.

One important feature of the program intended to give confidence to the occasional user is the production of a log file that documents all processing steps/decisions made in sufficient detail that anyones result can be reviewed by more experienced users easily --- it is sufficient to simply send such a reviewer just two things: the light curve and the log file.

1. *pyote* is designed for ease-of-use in the analysis of occultation light curves that can be modeled reasonably well with a model based on geometrical optics. Such light curves are common with star/asteroid occultations when the star is effectively a point source and the asteroid transit speed is such that diffraction effects are masked by the natural integration effect of the camera operation coupled with the frame rate of the video recorder.

2. Correlated noise caused by atmospheric scintillation is frequently present in occultation observations recorded at normal video rates of 25 or 30 frames per second. *pyote* utilizes statistically rigorous calculations to properly characterize the increased uncertainty in D/R time estimates due to such correlated noise. Additional noise correlation is often present due to the relatively slow electronics present in low-cost frame grabbers and in the camera electronics responsible for generating the composite video output.

3. Physically realistic models (but based on geometrical optics) are fit to the light curves with all decisions about details (complexity) of the model used made using the Akaike Information Criterion (AIC). In particular, an AIC calculation is always used to justify or reject sub-frame timing.

4. Maximum Likelihood Estimation is used throughout to determine 'best fit' of model light curves to the actual data.

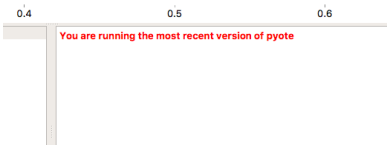
The gui for *pyote* is designed to lead the user through the necessary steps by enabling the buttons in sequence as each task is performed. So, initially, only two principal buttons are enabled: the 'info' button that brought up this document and the 'Read light curve' button. After reading this document, open a light curve, and follow the enabled buttons.

All of the major buttons have hover text associated. To learn (or refresh) how to use the program to analyze a light curve, spending a little time 'hovering' on the buttons will pay dividends.

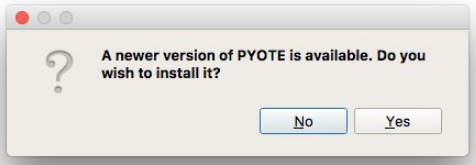
pyote will never change the input light curve, so experimentation is encouraged. There is a 'Start Over' button at the bottom that I encourage you to use freely.

Every step you make in the analysis is recorded in a log file. This is done because experience has shown that some light curves are touchy to analyze and it is useful to ask someone more experienced in running the program to look over your work. With the original light curve and a copy of the log file, your work can be exactly duplicated by someone else. And that log file is never deleted once it is opened for a particular light curve; it is simply appended to, so a record of each 'experiment' is thus always available.

Every time *pyote* is started, it connects to PyPI (assuming you have an internet connection) and checks to see if a more recent version of *pyote* has been added to the repository. If your version is completely up-to-date, you will see this



in the log file panel in the lower right-hand corner of the gui. Otherwise, this will appear:



Normally, you will want to click 'yes'. That will cause your current version of *pyote* to install (but not run) the newest version. Of course, to execute that new version, you will need to do a close and reopen.

As convenient as this is, there is always a small risk that a new version will actually 'break' something and that the 'cure' may take some time to be posted. But it is always possible to return to a specific previous version of *pyote.* The procedure to do this is explained below.

Open an Anaconda Prompt window if you are running Windows.

For a Mac installation, open a command window and type ***source activate***.

Then, type the following line in that command window:

1. pip install pyote==1.42

This command will uninstall the current (flawed) version of *pyote* and installs a specific version, in this case, version 1.42. Note the double == followed by the specific version number to be installed. (You can always determine a version of *pyote* that was working for you by opening a recent log file --- the *pyote* version that produced that log file is recorded there.)