PyMovie – a stellar occultation aperture photometry program
Bob Anderson 9 July 2019

Introduction

For extracting lightcurves (plots of star intensity versus time) from videos of stellar occultation events, the astronomy community already has **Limovie**, the pioneer application in this field, and **Tangra**. A natural question then is, “what are the reasons for yet another one”? The major responses to that question are listed below.

1. The prime motivator that drove the development of this program was the belief that an aperture photometry program that could deal robustly and effectively with videos that have been affected by wind shake would be a useful tool. PyMovie uses the idea of forming a sampling mask from the image itself so as to get a sampling mask that adapts/conforms to the shape of a star image that has been distorted by wind shake. Such a mask minimizes the inclusion of background pixels. The expected benefit from this procedure is an increased SNR, particularly during the occultation as the intensity drops and the image gets smaller. This effect is useful even for videos that do not exhibit wind shake.

2. OS interoperability: PyMovie has been designed from the beginning to run on both MacOS and Windows platforms. It is implemented using Python 3.7 and PyQt5 (for the GUI) and thus inherits the platform independence of those packages. The program will probably run on Linux as well, but has not been tested on that platform.

3. A set of aids for assisting the user in placing a measurement aperture at the occulted star position have been built in:

    1. There is an available on-line connection to nova.astrometry.net. Through that connection, a frame can be submitted to nova.astrometry.net for WCS calibration. With a WCS calibration available (which maps pixel coordinates to RA Dec coordinates), the user can position a measurement aperture at the occulted star position by specifying its RA Dec position. The star does not need to be visible on the video frame.

    2. To aid in getting RA Dec information, there is an on-line connection to VizierR which accepts a UCAC4 star ID and returns an hmsdms ICRS coordinate string ready to use.

    3. There is a simple-to-use tool that will register and sum a hundred or more frames (a simplified RegiStax) and thus generate an image intensified “finder” frame that will make visible any object for which a lightcurve can be extracted.

    4. It is also possible to select two stars in a frame, provide their RA Dec coordinates, and obtain a useable WCS calibration for that frame.

4. Ease of use, particularly for the occasional user, was constantly emphasized during the development. Of course, whether this goal has been achieved will be decided by those users.

5. In the category of weird-but-interesting, for tracking a target star that disappears, in addition to the normal use of one or two (use two if star field rotation must be taken into account) nearby reference stars to provide tracking information, it is possible to analyze backwards from higher numbered frames to lower numbered frames, an easy way to handle a video that has no usable reference stars and the target star disappears during the occultation.

General advice (tips) for new users

Tooltips:

Most of the GUI controls have 'explanations' implemented as a context menu that can be invoked by a right-click on the GUI item in question. If that GUI item already has a context menu, then you should right on the label for that item instead. These 'tooltips' are an important part of the documentation and should be consulted often. They will often contain details that have not been covered by this document.

Folders:

Each avi video should be 'placed' in its own folder --- but they don't have to be physically present in the folder; they can be 'aliases' (MacOS) or 'shortcuts' (Windows) so that there is no disruption to whatever scheme of video storage is already in place --- only a 'pointer' to them is needed, wherever they might already be located. These folders become the standard target directory for output files and other internally generated files that hold things like target star location and WCS calibration information.

There is a button available for taking an avi that has been opened and automatically creating an AVI-WCS folder for that avi. The folder will be created wherever you want it and populated with a 'shortcut' (if you are in a Windows environment) or an 'alias' (if you are in a MacOs environment). That folder will then collect all the files (there can be many) that are related to that avi including its OCR profile (described later), csv files from lightcurve extractions, PyOTE results, etc.

For an observation recorded in FITS format, the sequence of FITS files that comprise the video must be physically present in the folder; there is currently no 'pointer' scheme available as there is for the avi format. But a FITS observation necessarily has its own folder to hold the sequence of individual FITS files, one for each frame, and PyMovie will simply use that folder as the standard target directory.

Suggested workflow sequence for processing a new video:

If the video is an avi, create an AVI-WCS folder for the video using the button provided for this purpose. This is a good practice even if there is no timestamp on the video and there is no need for a WCS calibration (because the location of the target star is obvious, for example). There are still files that will result from PyMovie and PyOTE operations that it will be nice to have collected in an easy to find single that follows this one location.

Open the AVI-WCS folder created above. If you are processing a FITS recording, open its folder instead.

Locate the occulted star (there are sections of this document dedicated to this topic) and place a measurement aperture on it either directly (visually) or via WCS calibaration. If you use either of the methods for obtaining a WCS calibration (nova.astrometry.net or manual WCS calibration) the position of this aperture will become 'sticky' and automatically restored whenever this folder is opened in the future, a useful time-saver. If there is no WCS calibration and target location recorded in the folder, the initial image will have no apertures.

Select a nearby star to use for tracking, set its adaptive mask threshold, and turn it yellow to indicate that it is now a tracking aperture and that all other apertures should be locked to it and follow its frame to frame x-y movements.

If the star field has detectable rotation during the observation, select a second star and add a tracking aperture to it. Now all other apertures will be locked to this pair of apertures and follow not only their frame to frame x-y position changes but also their rotation.

Set a default circular aperture to be used when the adaptive mask generation fails to find 9 or more image pixels above the clipping threshold. (There is an alternative source for a default mask --- this is discussed in the section titled: Use of 'yellow mask' as the default mask.)

Set the clipping threshold for the target aperture.

Start the analysis.

Monitoring the on-going analysis:

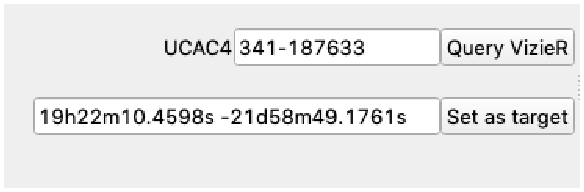
While an analysis is running, you can check on how things are progressing by clicking on Display plots. This will not disrupt or delay the ongoing analysis --- it will just display a snapshot of the lightcurve extraction so far.

At any time, hovering the mouse cursor over an aperture will cause the Thumbnail One and Thumbnail Two images to be connected to the 'pointed-at' aperture so that you can see the current image and its current mask. This same action causes the stats of this aperture to be printed in the text box at the lower left. In addition, the special properties of this aperture will be shown in the status bar at the bottom edge of the GUI.

Pausing and resuming an analysis is seamless: no data will be lost or corrupted.

Use on-line VizieR query to save time:

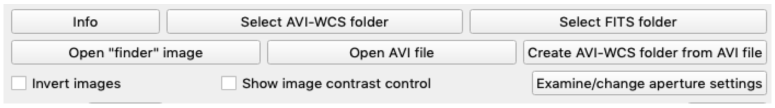
There is often a need to know the RA Dec coordinates (in ICRS hmsdms format) of a UCAC4 object, particularly when doing a 'manual WCS calibration'. To help with this task, PyMovie provides an on-line connection to the VizieR catalog. Here is what it looks like in operation (enter UCAC4 ID in xxx-xxxxx format, then click the Query VizieR button --- the response is for the J2000 epoch and with no proper motion corrections):


UCAC4 341-187633

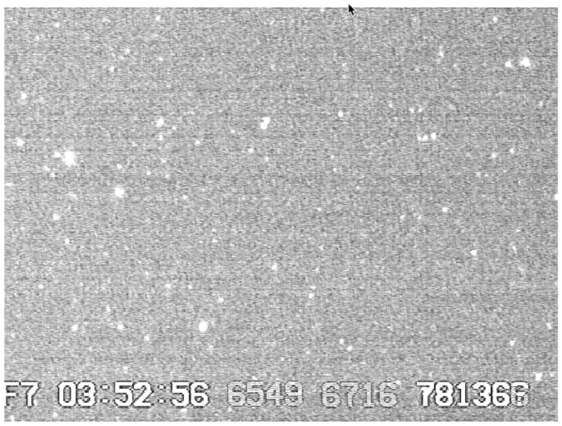
19h22m10.4598s -21d58m49.1761s

Use the image contrast control to customize image scaling:

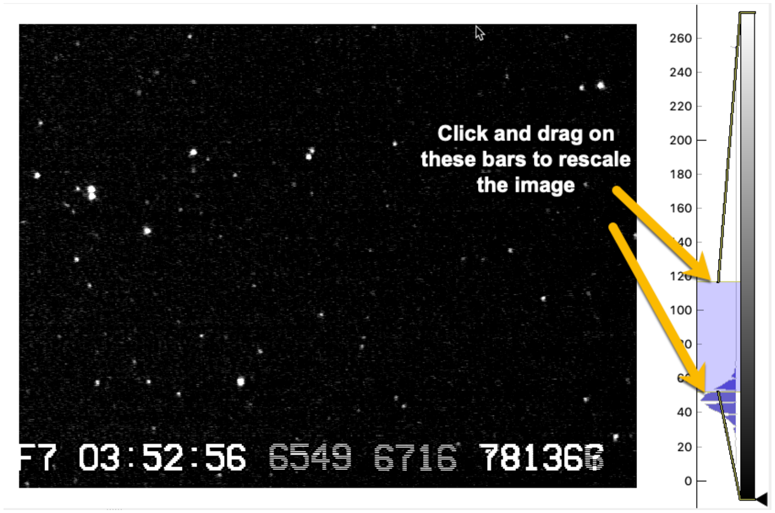
Enabling the 'image contrast' tool shown below ...


Info
Select AVI-WCS folder
Select FITS folder
Open "finder" image
Open AVI file
Create AVI-WCS folder from AVI file
Invert images
Show image contrast control
Examine/change aperture settings

… will make it possible to turn the image display from this …



… to this …

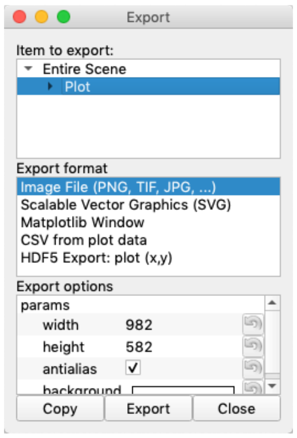

Click and drag on these bars to rescale the image
F7 03:52:56 6549 6716 781366
260
240
220
200
180
160
140
120
100
80
60
40
20
0

Note: closing (disabling the control) locks in the selected scaling for all images.

Note: this setting does NOT affect the underlying data. It will NOT affect values in the extracted lightcurves. This is a visual effect only. When the mouse cursor is placed on a pixel, the intensity value that appears in the Status bar is that of the underlying data, not the visually rescaled data.

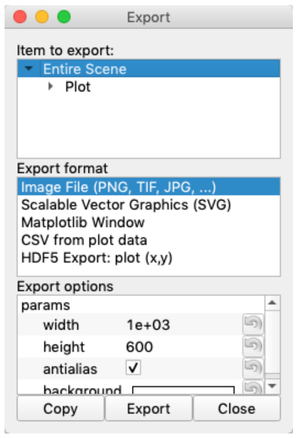
Use this 'trick' to export (record) plots and images:

All images and plots have a context menu that includes an Export option that is useful for saving interesting graphics to the folder in use. BUT, there's a trick to it. When the Export dialog first appears, it will have highlighted Plot as the item to export. You **must** change that selection to Entire Scene as shown in the righthand snip below before clicking on the Export button at the bottom, otherwise an error will be reported. The other defaults are fine as-is.



params

width	982
height	582
antialias	✓
background	



params

width	1e+03
height	600
antialias	✓
background	

Use of 'yellow mask' as the default mask

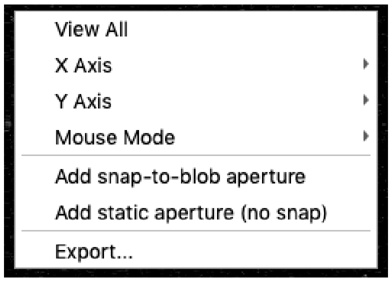
In severe wind shake conditions, the use of a default circular mask may not be effective. In that case, an excessively large radius mask could be needed and the resulting degradation of SNR might become intolerable.

For a case like this, it is possible to specify that the default mask to use in the target aperture be copied from the mask computed from a yellow tracking aperture. The idea here is that the distortions caused by the wind shake will be well represented by the shape of the mask computed from a bright tracking star whereas a similar mask would be difficult or impossible to calculate from a much dimmer target star.

This is an experimental feature that has not yet proved its worth. Use with care.

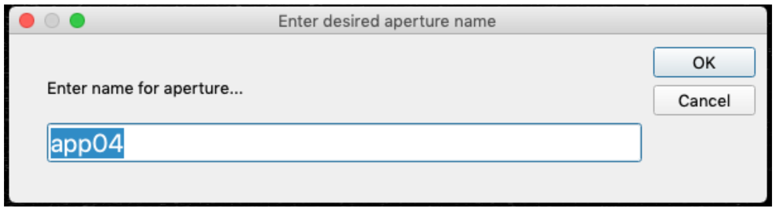
Measurement apertures: placement and properties

Measurement apertures are created and placed on a frame image through a context menu invoked by the usual right-click of the mouse on the frame image. The menu that appears looks like this:



The menu entries of interest are Add snap-to-blob aperture and Add static aperture (no snap).

Whenever an aperture is added, you will be prompted to give it a name. The prompt looks like this:



A unique default name is already filled in, but it is highly recommended that you take the time to provide a better, more meaningful name. The name given will appear on plots, in plot legends, in csv column headings, and in messages composed when the aperture is pointed at with the mouse cursor. A good name will add clarity.

A snap-to-blob aperture will find the largest blob (grouping of white pixels) anywhere in the aperture and center the aperture on that blob (although the centering will not occur until you decide what name to give the aperture). It will automatically determine the proper number of counts above bkavg (background average) to set the adaptive mask clipping point at or near the 1 sigma level, a useful starting point.

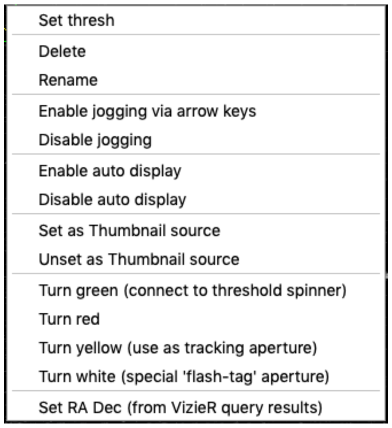
A static aperture is designed to do no 'snapping' at all. Instead, its clipping threshold is set to a very high number, forcing the substitution of a default circular mask. Such a mask has the property that its centroid is not image sensitive and so the aperture will not move when the image underneath changes. This feature is used to advantage when placing a target aperture on a “finder” image --- when the '”finder” image is replaced by an actual frame image, the static aperture stays where it was placed!

For placing a static aperture, it is helpful to zoom into the image until it is easy to put the mouse cursor on a specific pixel. The procedure is: zoom to pixel; place cursor on desired pixel; right click to get context menu; select Add static aperture. The aperture will be placed exactly on the selected pixel.

Note that it is possible to use the arrow keys to adjust the position of an aperture in pixel sized steps. When an aperture is first created, jogging is disabled. If you want to 'jog' an aperture, it is necessary to explicitly enable that feature using the context menu for the aperture.

It is even possible to use this feature to 'stack' an aperture directly on top of another aperture. One might do this so that a particular star could used both as a tracking star with a 3 sigma clipping threshold yet also be measured with a fixed circular aperture for some test.

Apertures that have been placed have certain properties that can be changed via a context menu (invoked as usual by right-clicking somewhere inside the aperture) that looks like this:


Set thresh
Delete
Rename
Enable jogging via arrow keys
Disable jogging
Enable auto display
Disable auto display
Set as Thumbnail source
Unset as Thumbnail source
Turn green (connect to threshold spinner)
Turn red
Turn yellow (use as tracking aperture)
Turn white (special 'flash-tag' aperture)
Set RA Dec (from VizieR query results)

Here are the meanings of the menu items:

Set thresh – when clicked, sets the mask clipping level to the value contained in the spinner labelled set mskth counts above bkavg, a quick way to set a particular value for this parameter.

Delete – removes the aperture.

Rename – allows the name to be changed.

Enable/disable jogging – connects/disconnects the aperture from the arrow keys. It is possible to have multiple apertures connected to the arrow keys at the same time.

Set/unset as Thumbnail source – during an analysis run, at each frame change, the Thumbnail One and Thumbnail Two images are updated using whichever aperture has last been set as the Thumbnail source via this menu item.

A 'green' aperture is 'connected' to the set mskth counts above bkavg spinner so that incremental up/down changes to the mask clipping level can be conveniently made.

A 'red' aperture, when able to move on its own, will only snap to a blob that is withing 8 pixels of the aperture center so as not to be 'attracted' by a brighter near-by star in the aperture.

A 'yellow' aperture is a tracking aperture. It snaps to the brightest, biggest blob anywhere in the aperture and drags other apertures with it (except 'white' apertures). When a yellow aperture is present, all apertures are locked together in a constellation and move as a unit (again, with the exception of a 'white' aperture which is never part of a constellation). If there are two yellow apertures (the maximum number that is allowed) the aperture constellation will both translate (move in xy directions) but also rotate as a unit if the yellow apertures rotate.

A 'white' aperture is very special and is relevant only when using 'flash timing' as an alternative to a VTI timestamp overlay for timing an occultation. One special property of a white aperture is that it never moves. It remains where initially placed, even if there is a 'yellow' aperture defined. The other special property is that the 'signal' from a white aperture is the sum of **all** pixel values in the aperture; there is no background subtraction applied. This aperture is designed to detect a low intensity flash (from an LED or cell phone) that floods the image at a known time. This flash signal can be utilized by PyOTE to accurately time the video to within a few milliseconds, if the flash does not saturate the pixels. Because the signal from this aperture is the sum of a large number of pixels, it is easy to ensure that saturation does not occur yet still have a strong detected signal. The aperture remains at a fixed location because the precise positioning of the 'flash' LED or cell phone may have resulted in shadow areas and brighter areas, so it is useful to freeze the position of the sensing aperture in a brighter area.

Set RA Dec – this is used only during a manual WCS calibration procedure (there is a later section that explains its use).

Apertures can have have sizes of 21 x 21, 31 x 31, 41 x 41 and 51 x 51. The choice is not critical and the default of 51 x 51 is nearly always a good choice. The size of the aperture does control how many pixels are considered by the Robust Estimator (see a later dedicated section for details on this item). That routine discards as outliers 50% of all pixels, and utilizes all pixels in the aperture, including the star pixels themselves. A too small aperture might have this calculation compromised if there are an excessively large number of star pixels in the aperture on a percentage basis. Therefore, it is safer to use a larger aperture than a small aperture.

How to best place an aperture at the occulted star position

Placing a measurement aperture at the occulted star position is a critical, but often troublesome, starting point for a lightcurve extraction. In priority order (best option to worst option), the following methods are supported:

If the occulted star is bright enough to be visible in a frame, use the star chart associated with the observation to locate the target star and add a snap-to-blob aperture there.

If the occulted star is not visible, use frame stacking and summing to generate an intensified image “finder” frame (there is a dedicated section on this topic). Then, use the observation star chart to find the, now hopefully visible, target star. Add a static aperture (NOT snap-to-blob) there.

If the video field of view is sufficiently wide (15 arcminutes or more), select a frame for which a WCS calibration is desired, trim off the timestamp (if any) and send the resulting image to nova.astrometry.net (see dedicated section on this topic). With a WCS calibrated frame, it is now possible to place the measuring aperture at the occulted star position by specifying its RA Dec coordinates. If you have a UCAC4 number for the target star, the Query VizieR button will get you the needed coordinates in proper format very quickly and easily. The advantage of this procedure is that no star chart is needed, just the RA Dec coordinates of the occulted star.

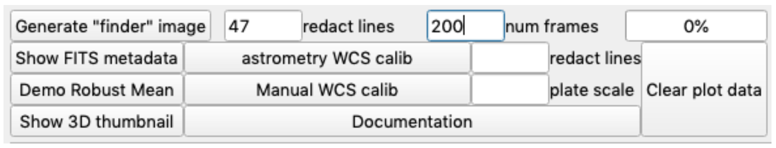
As a last resort, if there are two good reference stars for which you know the UCAC4 numbers, you can follow the somewhat tedious but quite effective Manual WCS calibration procedure. There is a section of this document dedicated to describing that process. It is an alternative to using nova.astrometry.net when the video field of view is too small (all too often the case for the bigger telescope/camera combinations used for occultation observations).

Generating a “finder” image

Some occultation observations involve an occulted star of such low magnitude that it is essentially invisible in any single frame of the video because the signal from the star is barely above the background noise level --- visually invisible. Yet, if a measurement aperture could be positioned at the target location, a lightcurve, albeit a noisy one, can be extracted from the video.

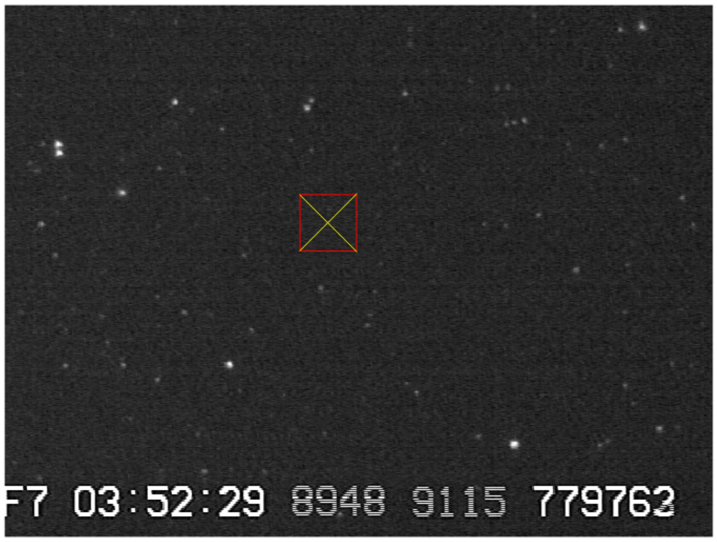
An approach that is sometimes used to deal with this situation is to employ a stand-alone program called RegiStax to 'stack' a hundred or so frames of the video to get an enhanced/intensified image and use this image in conjunction with a star chart to properly place an aperture at the occulted star position. PyMovie will open such an image (if it is a .bmp file) and allow an aperture to be placed using that image. But, there's a better way ...

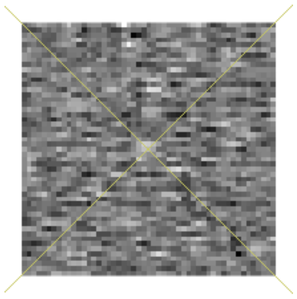
The “finder” image methodology has proven to be so useful and easy to use that we have incorporated the ability to generate such a 'finder” image into the application and thus eliminate the need to invoke (and learn) RegiStax. It is accessed by the portion of the GUI shown below:



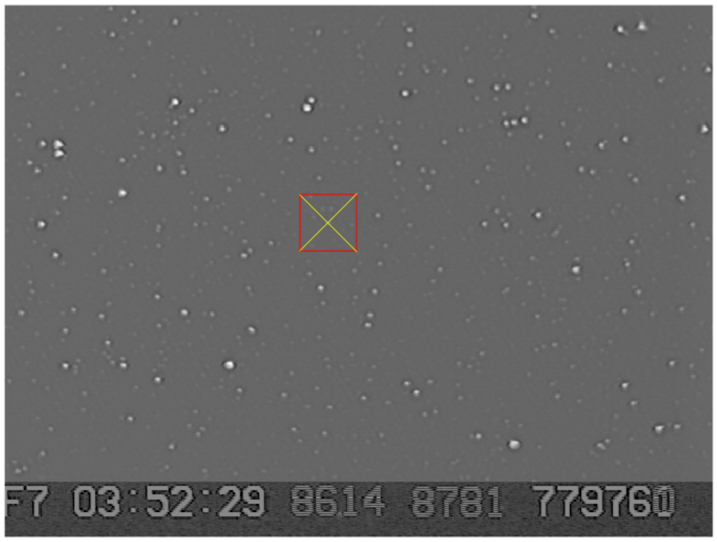
In the GUI snip shown above, we have specified that 47 lines from the bottom are to be 'redacted' (this removes the timestamp --- more on this topic later) and that 200 frames are to be stacked.

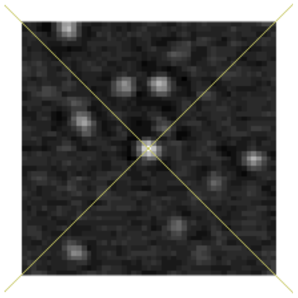
The starting frame looked like this (shown with its corresponding Thumbnail):





It is difficult to see the target star in this image or thumbnail, even though there is already a correctly placed aperture that tells you exactly where to look.





But the “finder” image that was generated from a 200 frame stack looks like this (shown with the Thumbnail blowup of the target aperture):

… and now it is easy to see the target star and to properly position a measurement aperture over it.

In both images, there is a visible timestamp although the one in the “finder” image looks a bit different. That is because, for the 'stacking and summing' algorithm used by PyMovie, it is necessary to remove the timestamp, process the images, then add back the timestamp from the first (reference) frame so as to clearly identify which frame all the other frames were aligned with.

It is necessary to remove the timestamp because PyMovie uses the Fast Fourier Phase Correlation algorithm to align each frame with the first frame before summing the pixels. That algorithm finds the x-y translation that results in the highest correlation (match-up) between **all** the pixels in the two images. It is thus essential that there be no substantial set of pixels that do **not** move, such as a timestamp. For that reason, the number of redact lines needs to be set so that all of the timestamp information is removed. A yes/no dialog box is presented prior to running the stacking procedure that asks whether the timestamp lines have been removed. You can answer 'no' to try a different number.

Note: hot pixels are another source of pixels that don't move. In practice, their presence has not kept the stacking procedure from working, probably because their numbers are so small.

Using nova.astrometry.net for WCS calibration of an avi frame

nova.astrometry.net operates a web site that will do a robust, 'blind' identification and WCS calibration (mapping of pixel coordinates to ICRS RA Dec coordinates) of any images submitted to it. A 'blind' identification means that no information about plate scale or sky location of the image is needed.

This service is magical, in my view, but it comes with a caveat: the image to be identified must have a wide enough field of view. I found it works fast and accurately with 15 arcminute wide images. Unfortunately for the occultation community, it is often the case that the field-of-view (fov) is less than this.

When nova.astrometry.net is given a 'too-small-fov' image, it fails to find a solution and takes a very long time (up to an hour) to reach that conclusion. Even worse, the web site processes up to 8 jobs at a time (the number of cores in their server) in the order in which they were presented (not one per user). So, if you persist in submitting these 'too-small-fov' images (by closing PyMovie and starting over multiple times), when you have submitted 8 such images, the web site will no longer process further images and you will have succeeded in locking out users from around the world. I have talked to them about this issue and asked if there is a way to 'cancel' a job: their response is that there is no way at present.

So, if you use PyMovie to submit an image to this site and you don't get a 'solution' within 5 minutes (2 minutes was typical for me in early testing), give up and use another method. Please don't crash their site by persisting.

While the site is designed for 'blind' identification, it does work faster if you give it information about where to look in the sky and what zone around that location to search in. PyMovie gives it the location of the target and specifies a radius of 1 degree for the search cone. In addition, it is assumed that It may be helpful to remove any timestamp overlay as those features will be interpreted as stars and the inclusion of non-existent stars can hardly help the search/identification process. To remove the timestamp lines before submitting the image, put an appropriate number in the redact lines box. Before PyMovie sends the image to nova.astrometry.net, you will be asked to confirm that the timestamp lines have been satisfactorily removed. If they haven't, answer 'no' and try again.

Performing manual “two star” WCS calibration

It is possible to determine the WCS calibration for a frame with fair accuracy (+/- 1 pixel) if there are two conveniently available reference stars with known RA Dec positions visible in the frame. If the stars have a known UCAC4 designation, the built-in VizieR query will be the best way to get the needed ICRS coordinates in a ready-to-use format.

It is best to use reference stars that are spaced as far apart as possible while keeping one reference star close to the expected target position. This is because a critical parameter that is needed for accurate mapping between pixel coordinates and RA Dec coordinates is plate scale. Plate scale is the ratio of angular distance to pixel distance between the two reference stars; that ratio will be less affected by slight errors in deciding which pixel in a star image to designate as its center when the distances involved are larger.

The plate scale used in a manual WCS calibration is always printed out. Since the plate scale for a given telescope/camera setup will be a constant, there is the possibility of, over time, determining ever more accurate plate scales for your particular setup(s) by keeping records and manually entering that number to improve accuracy and thus relieve the requirement for widely spaced reference stars.

The recommended workflow for carrying out a manual WCS calibration is:

Click on the Manual WCS calib button to start the process.

In the text box will appear a line prompting you to place aperture 1.

It is important to choose this first reference star to be as close as possible to where the target star is expected to be. This will give the most precise placement of the target aperture.

Enter the UCAC4 number of the star in the VizieR query edit box and click on Query VizieR.

Zoom into the image until it is easy to put the mouse cursor on a particular pixel.

Decide which pixel best represents the position of the star (brightest is usually a good choice) and add a static aperture at that point.

If you're unhappy about the placement, activate jogging for the aperture and use the arrow keys to reposition it.

Right click on this newly placed aperture and then click on the Set RA Dec menu item. This writes a small file into the folder-in-use that associates the x-y coordinates of the aperture with whatever RA Dec information was present in the Query VizieR result box.

In the text box will appear a line prompting you to place aperture 2. Repeat the procedure that you used for aperture 1 but this time choose a reference star that is far from reference star 1. This is done so that the estimate of plate scale can be as accurate as possible.

When aperture 2 has been placed, you will be prompted to enter the RA Dec of the target star (if not already supplied from an earlier run) and an aperture will be placed automatically at the target star location.

The next time you open this folder, the video will be positioned at the reference frame (which doesn't have to be frame 0) and the target aperture will be placed in the correct position --- very convenient for doing repeat/alternative lightcurve extractions.

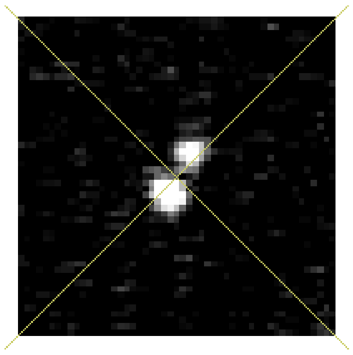
Discussion of Robust Mean calculation

The estimation of the average background pixel value in an aperture is crucial for extracting reliable and repeatable lightcurves. In order to calculate the intensity of a star, one sums all the pixel values selected by the sampling mask, and then subtracts a 'background' value which is calculated by multiplying the number of pixels in the sample mask by the average background pixel value. Therefore, any errors or biases in the average background pixel value estimation are amplified by a factor equal to the number of pixels in the sampling mask. This is one of the motivations behind the use of close fitting sample masks. It is also clear that quantization effects, such as those that would be present if a median value of the background were taken as a robust estimator of the mean (and therefore producing only integer values), would introduce a major noise source as well.

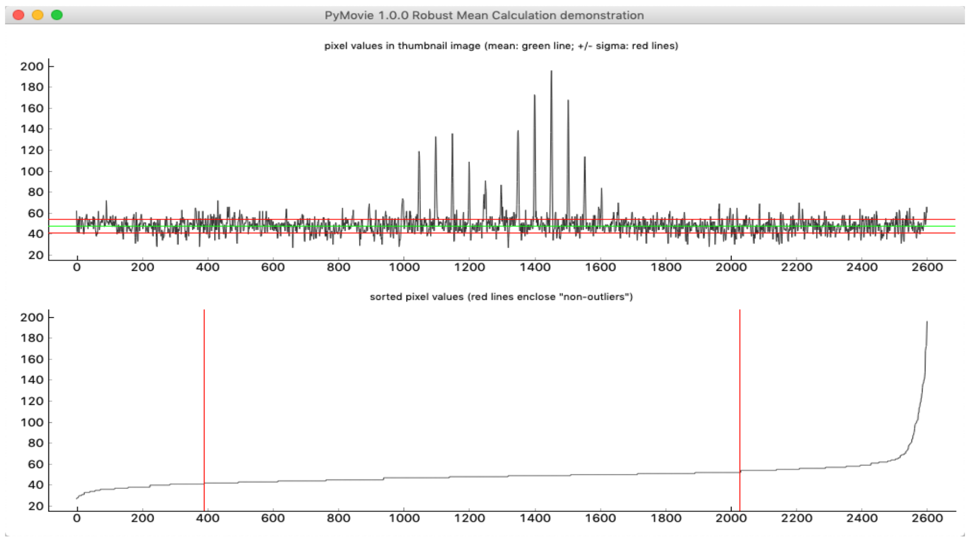
It is a common practice, in aperture photometry that uses circular masks, to add an annular ring around the central selection circle and estimate the background using the pixel values from that ring (perhaps with a small buffer zone as well). PyMovie does not follow this practice. Instead, it uses a 'robust estimator' on **all** the pixels in an aperture, with no attempt to exclude the pixels from the star itself, relying instead on the properties of the 'robust estimator' algorithm to ignore/exclude those pixels from the calculation automatically.

The term 'robust mean estimator' denotes an algorithm that is able to ignore values in a data set that are 'outliers'. An 'outlier' is a data value that is so far from the 'real' mean that its value should be excluded from the calculation because its inclusion would have an unduly large effect on the final value.

The PyMovie 'robust mean' estimator works by first sorting the data into ascending order. Then, a selection is made from the middle portion of the sorted data that excludes at least 50% of the data points from the two ends. Outliers in such a plot are points at the beginning of the plot and at the end of the plot. Where to make this selection is determined by positioning selection 'goal posts' such that the slope of a line connecting the last point of the low end 'outliers' and the first point of the high end 'outliers' is as close to zero as possible. A picture of this process in action will hopefully add a useful visualization of the process. Below is picture of the aperture to be operated on:



And here is how PyMovie handled the mean and standard deviation estimation:



The upper sub-plot turns the 2D image into a 1D array by doing a row-by-row plot of the pixel values in the aperture (the rows are simply plotted one after the other). Pixels from the stars are obvious in 14 of the 51 rows.

The bottom sub-plot show the placement of the 'selection' goal posts on the sorted values from the upper plot. Only the points enclosed by the vertical red lines were included in the calculation of the mean value. But the sorting procedure has pushed all the bright pixels from the stars to the right-hand edge of sub-plot where they are easily ignored and so the 'goal posts' clips them out of the calculation.

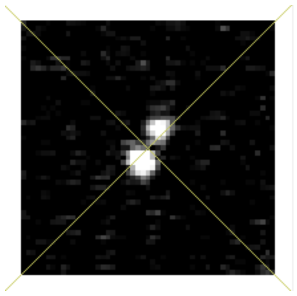
The standard deviation of the data points (second moment) is estimated using the well-know Median Absolute Deviation calculation (only the data points between the selection 'goal posts' are used) and multiplying that result by 1.4826 to correct it to reflect our assumption of an expected Gaussian distribution for background pixel values.

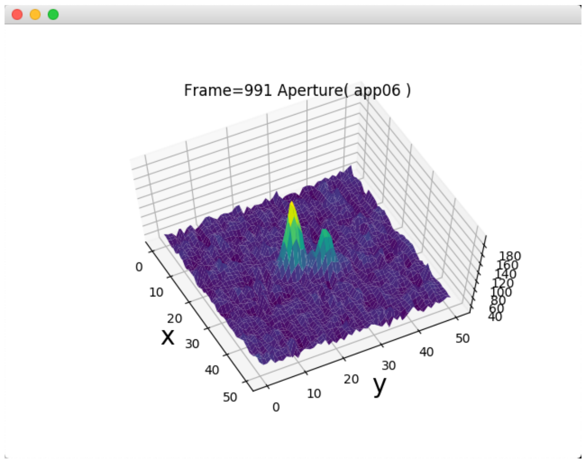
The standard deviation of the background values is used by PyMovie whenever a snap-to-blob aperture is placed. In that case, the adaptive mask clipping threshold is automatically set to the average background value + one standard deviation. We often refer to this as a 1 sigma mask. It is a good starting point for most usages.

If an aperture is to be used for tracking, it reduces tracking 'jitter' caused by fluctuations in the calculated mask shape if the mask is calculated at the 3 sigma level. This is effective if the tracking star is bright, but should be used with caution if the star is not so bright.

3D thumbnail viewer

It is possible to get a rotatable 3D view of the image being displayed in Thumbnail One. Click on the button in the lower left corner of the GUI that is labelled Show 3D thumbnail. A pretty picture of limited value other than satisfying curiosity will appear. The 3D view can be rotated to provide different view directions by clicking on the plot and dragging with the mouse. The 3D view can be used to determine if the target star is ‘saturated’ or nearly so. Saturated stars often show more noise in the light curve than is present in fainter stars due to the maximum brightness of the star being clipped by the circuitry of the camera. Here's an example:

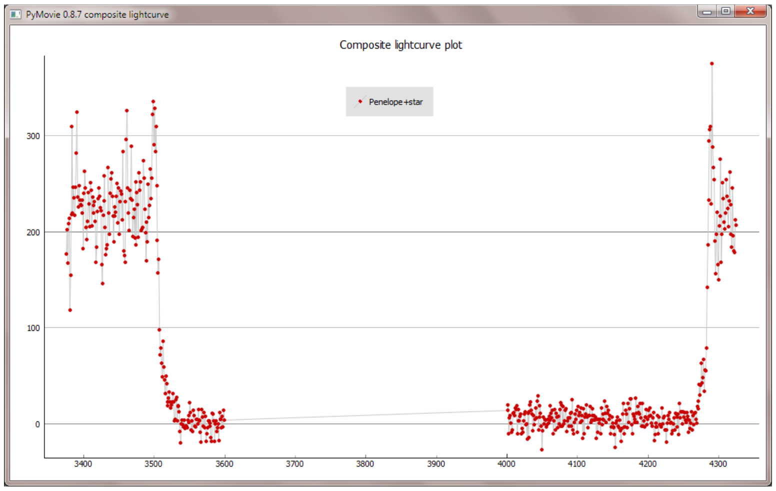




Backtracking: a technique for solving a specialty (rare?) 'tracking' problem

The specialty 'tracking' problem to be considered is when there is no reference star in the frame that can used to guide the target measurement aperture AND the occulted star disappears completely during the occultation AND it is possible that the target star position will have shifted during the event (as it would in a drift scan video).

In this case, one can take advantage of the fact that PyMovie can perform a lightcurve extraction 'backwards' (from higher numbered frames to lower numbered frames). Here is an example of a lightcurve generated in just that manner (Credit: Tony George; recent observation analysis):



The lightcurve was generated by first running from a starting frame of 3390 to a stop-at-frame of 3600 --- the normal 'forwards' direction. Then, the starting frame was set at 4320 and the stop-at-frame set to 4000 (this inversion is sensed by PyMovie and causes it to step backwards through the frames).

The resulting set of data points is sorted into ascending frame number order for use by the plotting routines and the csv export routine. Each data point is correctly associated with its frame number and placed in the correct position for independent determination of D and R by an appropriate occultation timing extraction program.

Timestamp handling

PyMovie reads the QHY 174 GPS timestamp (at least as composed by the SharpCap capture software) from the FITS header for each frame and adds it to the csv output file. For that camera, PyMovie is complete. For other cameras, only time will tell because ...

… files from the QHY 174/SmartCap setup encode the timestamp using a particular header name and a date-time format that appears generic, but may not be a standard that will be followed by other cameras. If/when other names/formats are encountered, they will be added to PyMovie upon request.

PyMovie is able to extract timestamps from VTI (Visual Time Inserter) overlays using optical character recognition (OCR) for the following VTI models:

IOTA VTI 2
IOTA VTI 3
BoxSprite GPS 3
Kiwi 40
Kiwi 41

To achieve the lowest possible OCR error rate, PyMovie allows/requires the user to manually position each character selection box by jogging them into visually optimum positions with the keyboard arrow keys. While this is tedious, it is also extremely flexible and works in cases where automatic box placement would fail. This manual method was also selected based on a belief that, for a particular recording chain of camera, VTI, and frame-grabber, the timestamp characters would always appear at the same screen position. Therefore, once the user has positioned the boxes once, that operation would not have to be repeated for other files recorded with the same equipment chain with the same settings. PyMovie allows an OCR profile to be named and saved for reuse to take advantage of this.

The other part of OCR is the selection of 'model digits'. These are the images that are subsequently compared against what is found inside the character selection boxes as each field is processed. PyMovie allows/requires the user to select the model digits 0 … 9 that are to be used. For a VTI that allow the background to 'leak' into the character boxes (not a good thing --- suppress this if you have the option, particularly with the Kiwi VTI), this selection of model digits becomes increasingly important as it has been found that including the actual background in the model digits reduces OCR error rates significantly. When an OCR profile is saved, both the box positions and the model digits are saved along with the VTI type.

If an OCR profile is loaded from list of saved profiles, it may be the case that while the box placement is correct, the model digits present when that profile was saved may not be representative of the digits present in the new file. It is easy to invoke a 'retrain' of the model digits and select a new set. This procedure is easy and takes little time and gives the best possible timestamp extraction.

For the Kiwi VTI, there is little visual difference between the '6' and '8' characters and also between the '8' and '9' characters. These are called 'confusion characters'. PyMovie reduces the number of 'confusion' errors by employing a secondary test whenever a 6, 8, or 9 is 'found' by the model digit matching process. For those characters, an examination of pixel values in specific locations is used to decide which of the two 'confusion' characters to report as the digit found.

The recommended workflow for activating timestamp OCR on a new avi is (we assume that no saved OCR profile is yet available):

Open the avi.

Click on the Create AVI-WCS folder from AVI file button. This will create the folder, fill it with a pointer to the avi and open the folder --- this is necessary because timestamp processing is only possible while operating from within an AVI-WCS folder.

The select VTI dropdown list control will become enabled.

From the dropdown list, choose the appropriate VTI. For IOTA, you will need to choose which format is present: safe mode, or full screen mode). For Kiwi, you will need to choose either left or right, which refers to the fact that the Kiwi timestamps have two possible positions that change at every rollover from one minute to the next.

The choice made above will cause a set of character selection boxes to placed on the image.

Reposition each selection box as required. Right-click on a character box to get a list of operations available: of interest at this point is the jogging controls.

Jog each box (or box group) until the character(s) enclosed is/are visually centered. It may help to click Show properties while doing this as it shows an expanded view of the character box in Thumbnail One and Two (they are the same image repeated).

For character boxes that are empty, enable jogging on the matching other field character box and jog the pair together, aligning to the character that is present in one of the boxes.

In general, be careful to disable jogging on boxes that are already properly placed so that you don't undo your work when you start jogging some other box.

After all boxes have been accurately placed, record the model digits 0 … 9.

When all model digits have been recorded, timestamp reading will automatically begin.

You should right-click a selection box and click show model digits to verify that you have recorded the digits 0 … 9 in the correct order.

You will probably want to save this OCR profile, so click on the Select/save custom ocr profile button. Give it a meaningful and memorable name and click Add current profile to list.

The recommended workflow for activating timestamp OCR on an avi that has been recorded using the same equipment chain that has previously been initialized for OCR and its profile saved is:

Open the avi.

Click on the Create AVI-WCS folder from AVI file button. This will create the folder, fill it with a pointer to the avi and open the folder --- this is necessary because timestamp processing is only possible while operating from within an AVI-WCS folder.

The select VTI dropdown list control will become enabled. IGNORE this and instead click the Select/save custom ocr profile button and choose an appropriate profile from the list.

If that choice places all the character selection correctly (it should), it remains only to see if the model digits retrieved from the saved profile match those of the current video.

Right-click on a character box and then click show model digits. If those digits look similar, you're done.

If there is a significant difference between the model digits and the video (perhaps because of background peeking through), ritht-click on a character box and click on retrain model digits. That will erase the model digits loaded from the saved profile and allow for a new sel of digits to be recorded.

Acknowledgements

As I recall, this project had its genesis in a discussion I had with Tony George some months ago. He said something like:

> "It would be really nice to have a program that could extract lightcurves without a great deal of hassle from videos that have been recorded in windy or gusty conditions. I get quite a few videos sent to me that have obvious wind shake effects and they are a pain".

As it happens, I had been recently dabbling in image processing using the OpenCV Python library and so replied (foolishly?): "I might be able to do something about that".

One thing led to another and on to yet another thing and PyMovie is the result. Throughout the development, Tony has been the primary external tester and I have used his struggles and observations and suggestions to (hopefully) refine the GUI and procedures and workflows in the hope of making a new user's out-of-the-box-experience a good one.

Tony also created and hosted a PyMovie Beta Test Group to help test the program and provide suggestions for improving the user experience (UX is the new term for this). Meeting most Thursdays at 6 pm via Zoom for several months, this group added their test results, experiences, and suggestions for improvements/changes. As of this writing, the members of the beta test group were:

Jerry Bardecker
Ted Blank
Tony George (leader)
Ernie Iverson
Greg Lyzenga
Chuck McPartlin
John Moore
Ted Swift
George Viscome

If you find this program useful in your work, be sure to thank Tony for getting this project kicked-off in the first place.

Contact information

I can be contacted at bob.anderson.ok@gmail.com

Tony George can be contacted at: triastro@oregontrail.net

Appendix A:

A study of the use of adaptive masks versus fixed circular masks in dealing with wind shake.

As the prime motivation behind the development of PyMovie was to employ the idea of adaptive masking (forming a sampling mask from the image itself) to better deal with videos that were subjected to wind shake, it's worth taking the time to explore the effects of this technique in comparison with the more standard use of fixed size circular apertures. Simple question: Does it do any good (or harm)?

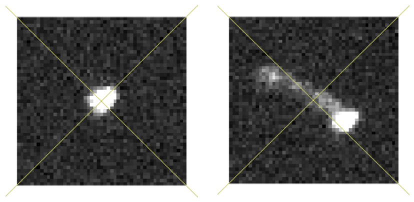
To illustrate the lightcurve characteristics obtained when adaptive mask extraction versus fixed size circular mask extraction is employed, I will use an example video shared with me by Eliot Young, et al, of the August 15, 2018 occultation of UCAC4 341-187633 by Pluto.

The production of this video was supported by a grant from NSF/SWRI.

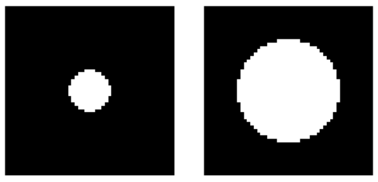
The interest in this particular occultation is that the lightcurve has D and R transitions that are gradual and that the particular shape of these transitions can be used to infer details of the atmosphere of Pluto.

The video was recorded by a QHY-174GPS camera (12 bit A/D) attached to a largish, but still portable telescope. The frame exposure time was 0.250 seconds. Unfortunately, while otherwise a good observation video, wind gusts during the observation shook the telescope and the effects of this 'wind shake' are significant in some regions of the recording --- but it makes a good study case for our purposes here.

Let's take a look at what wind shake images look like. Below are two zoomed thumbnails (51 x 51 pixels) of a bright star in the Pluto video. On the left, we're looking at frame 15. On the right, we're looking at frame 113. It's an example of what wind shake can do to a star image.



If we use fixed circular masks and want to ensure that we include all the pixels affected by the star image, we would need to use the following masks:

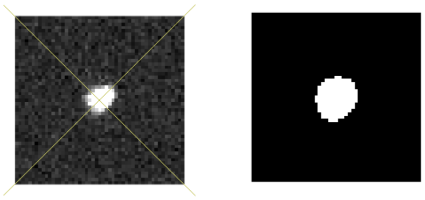


The number of pixels in the above masks are 121 and 553 respectively. For the case of the 553 pixel mask in particular, a high proportion of the sampled pixels necessarily come from the background. That introduces noise from two major sources. First is the normal Gaussian noise that grows as sigma * sqrt(num_background_pixels) where sigma is the estimated standard deviation of the pixel noise. Second, and often even more significant, is that any small error in the estimate of what value to use as the average background gets amplified by the number of pixels in the mask. This is because we typically report a background_corrected_intensity calculated as sum(masked pixel values) – num_background_pixels * average_background_pixel_value.

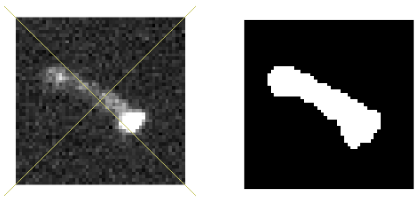
It is always necessary to use a statistically robust estimation of the average background pixel value and the noise level (sigma). A median estimate is sometimes used as a robust estimator of the mean, and while it's pretty good and simple and fast method, it reports the background as an integer; as a result, given an actual background value that is close to say 27.5 counts, the median value estimate of the background will fluctuate between 27 counts and 28 counts from frame to frame. In the example above, with 553 pixels in the mask, that translates to a frame to frame intensity change of 553 counts! That can be quite a lot of extra noise, particularly for low intensity images. (PyMove uses a robust estimator that avoids this quantization effect and outputs floating point estimates of pixel mean and standard deviation --- this is discussed and illustrated in a separate section).

An adaptive mask is calculated by assigning the value 1 to all image pixel values that exceed a given threshold level and 0 to all others. In PyMovie, the original image is subjected to 5 x 5 Gaussian blur before the thresholding action is applied. This serves to slightly broaden the mask to better capture the pixels from the wings of the image.

Here is what the adaptive mask looks like for frame 15 when calculated with a threshold set at 1 sigma (27 counts) above average background:

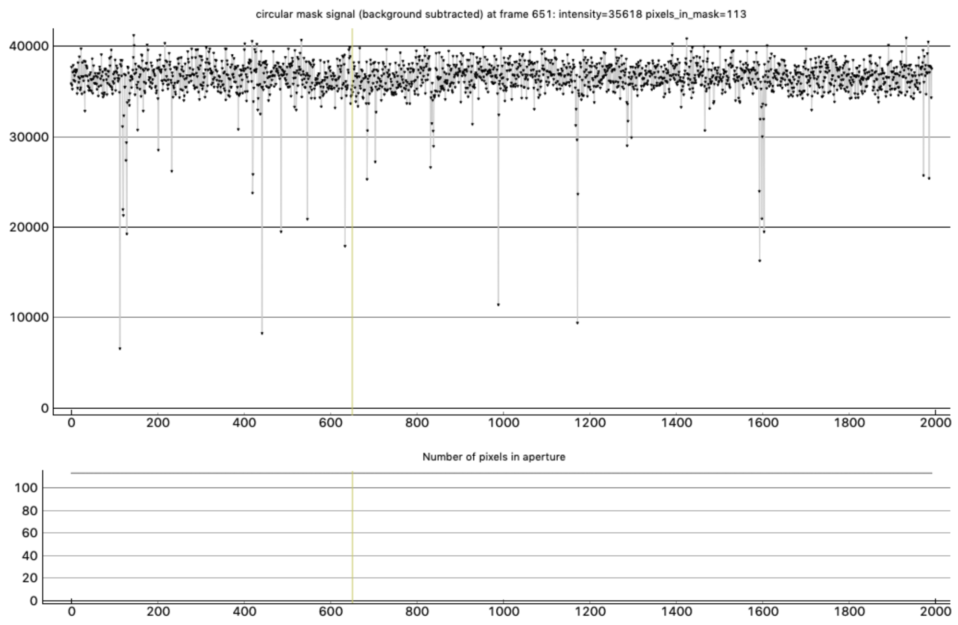


And for frame 113, the adaptive mask under the same conditions as above looks like:



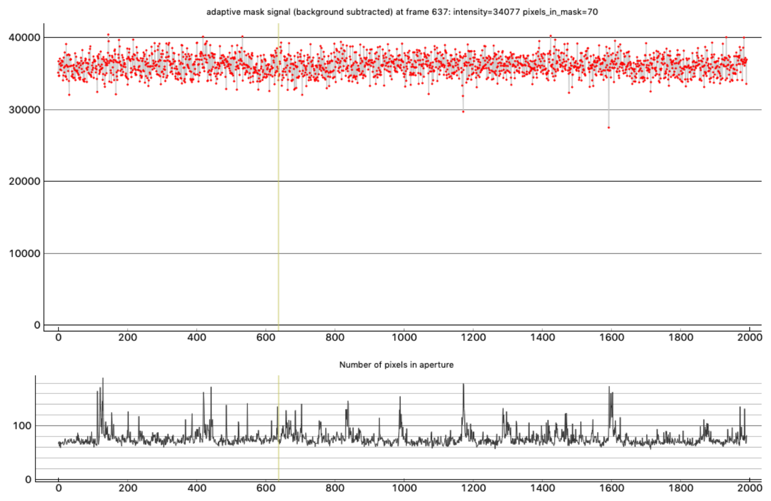
The number of pixels in the above masks are 131 and 330 respectively.

To show the differences between the use of a fixed circular mask versus an adaptive mask, we'll extract the lightcurve of a bright star from the Pluto video. In the plot below, a fixed size circular mask (6 pixel radius containing 113 pixels) was used. Note: it is a convention in PyMovie that if an intensity value was calculated using a fixed size mask, it is plotted with a black symbol for that data point. If an adaptive mask was used, the data point will be plotted with a colored symbol.



The sub-plot at the bottom is not particularly interesting as it simply shows the number of pixels in the sampling mask and, in this case, it was a constant value of 113. But the lightcurve itself (the upper sub-plot) shows occasional regions where the calculated intensity deviates significantly from the average – those are regions affected by wind shake where the star image got 'smeared' and 'diluted' so that, from the point of view of a fixed circular mask, the image was much dimmer.

In the plot below, using the same star as above but employing an adaptive mask with the thresholding limit set at 2 sigma (54 counts to be specific) above average background, we get:

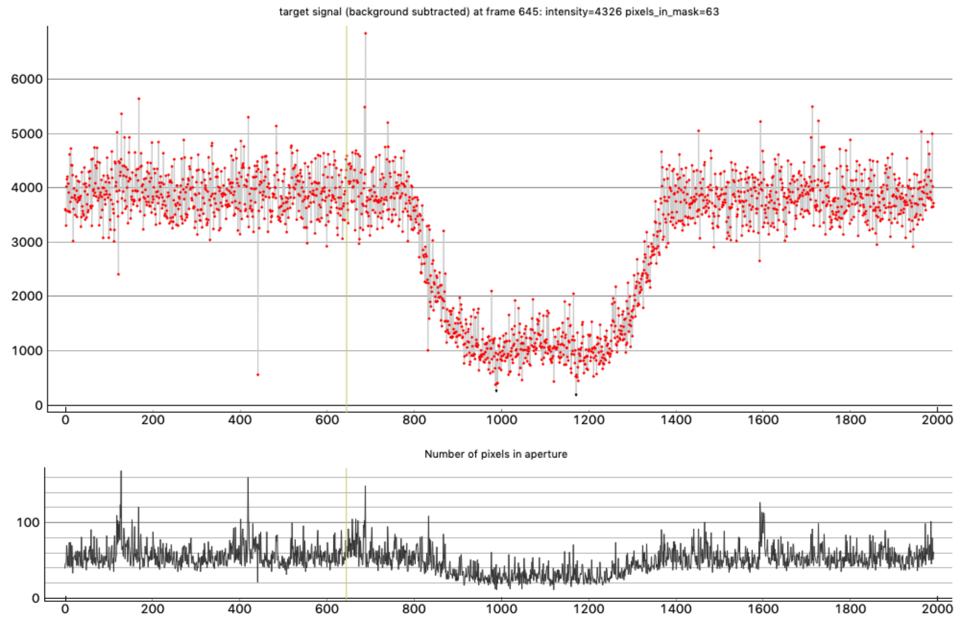


Now the lower sub-plot is more interesting because it shows the number of pixels in the mask that was fit to the actual star image. The regions of wind shake show up clearly in this sub-plot as sharp increases in the size of the enveloping mask (number of pixels in the mask). But the lightcurve in the upper plot shows barely any indication of the presence of wind shake. The adaptive mask technique was successful in dealing with the wind shake frames in the sense that the effects are almost completely suppressed in the extracted lightcurve, yet the average intensity matches very closely that obtained using a fixed circular mask.

I should add that these good results are due in part to the fact that a 12 bit video camera was in use. That means that there was enough dynamic range available so that a 'smeared' image could still record with fidelity the photon flux from the star. Had an 8 bit camera been in use, it's likely that some of the 'smearing' would have effectively erased portions of the image as the photon counts at some of the pixels fell below the ability of an 8 bit a/d to 'see' the accumulated photons. I have personally seen 8 bit videos where several frames were so affected by wind shake that no star image at all was present; it was completely erased.

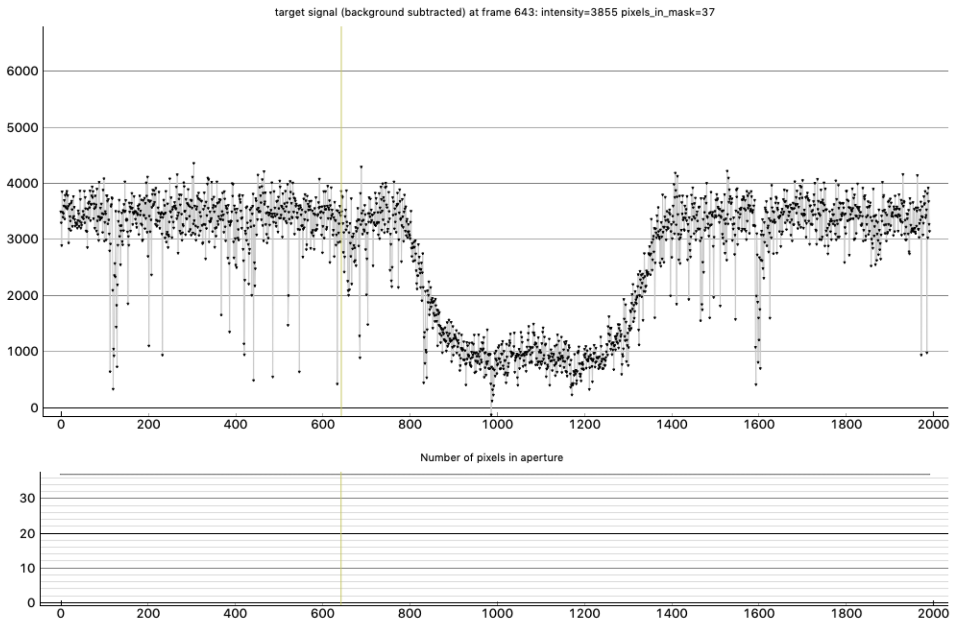
Next we explore the extraction of the Pluto occultation lightcurve using adaptive masks and fixed masks.

In the first example, shown below, the target mask was computed using a 0.5 sigma clipping level (13 counts above background), the tracking star masks were generated using a 2.0 sigma clipping level, and a default fixed circular mask of 3.8 pixels radius (45 pixels) was selected. This default mask is needed for the situation where the light from the target star is so low that fewer than 9 pixels exceed the clipping level in the computed mask. Effectively, there is no star there to form a mask from, so a default mask is substituted instead. This happened twice: those instances are indicated by the two black data points in the lightcurve near frame 990 and again near frame 1180.



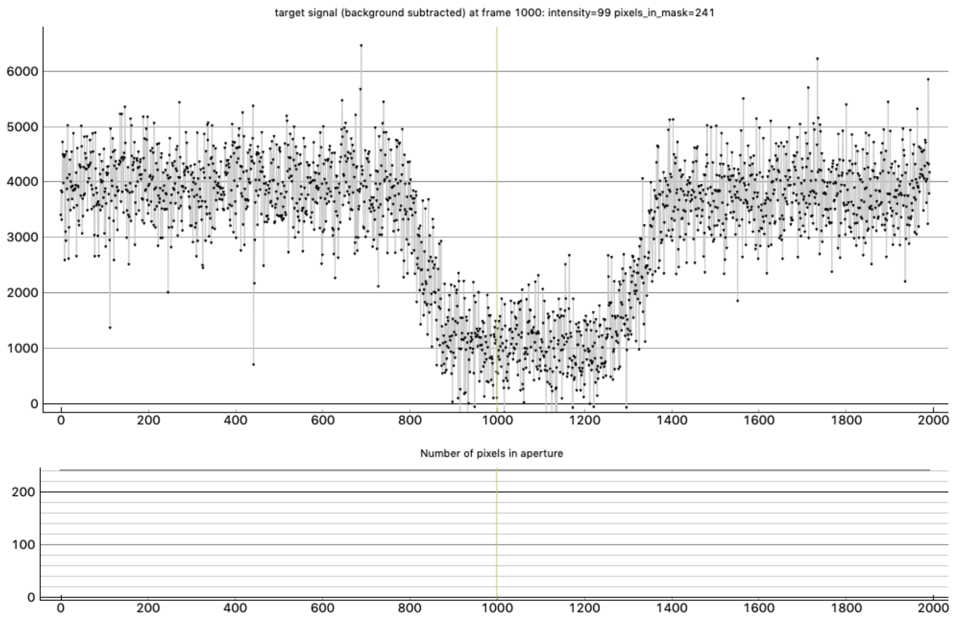
Note in particular the manner in which the sampling mask tightened up as the intensity of the star image decreased during the occultation. You can also see qualitatively the SNR increase relative to the baseline.

For the next example, a small fixed mask 3.3 pixels in radius containing just 37 pixels was employed instead of the adaptive mask. This is the approach one would take to improve the SNR during the occultation when using normal fixed size sampling masks.



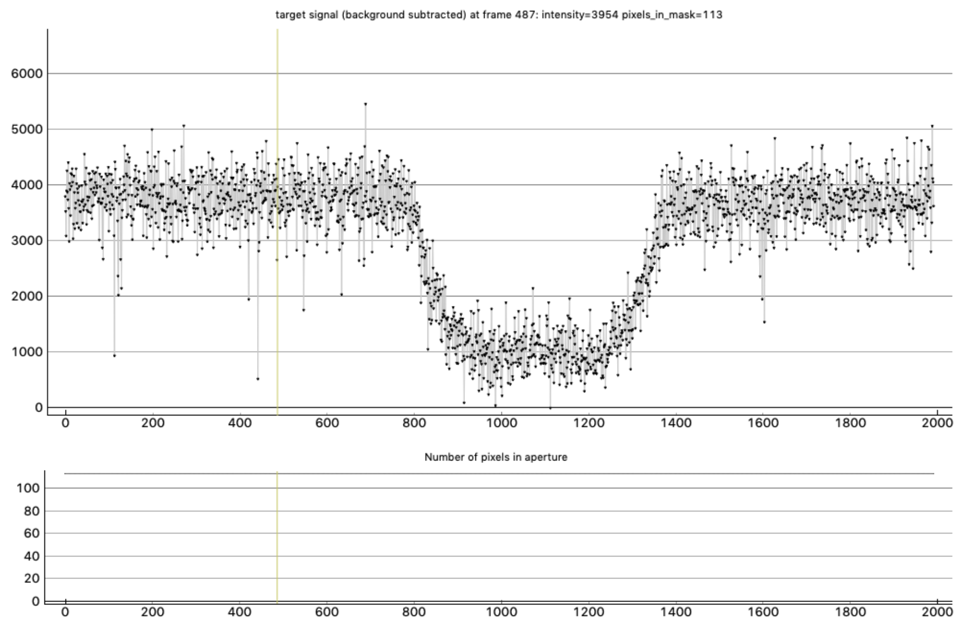
Note that there is some loss of gathered light (the baseline value B is about 3500 instead of 3800) and the wind shake effect becomes apparent in the light curve as intensity points that drop to near zero. But, if one can look past/ignore the wind shake artifacts, qualitatively the SNR during the occultation is similar to that obtained using the adaptive mask.

If one were bothered by the obvious wind shake artifacts and gave priority instead to reducing those artifacts, a larger fixed size aperture would be selected to more completely enclose the star image when it gets 'smeared' in the wind shake regions. Selecting a fixed circular mask of 8.8 pixel radius containing 241 pixels, the resulting lightcurve is shown below.



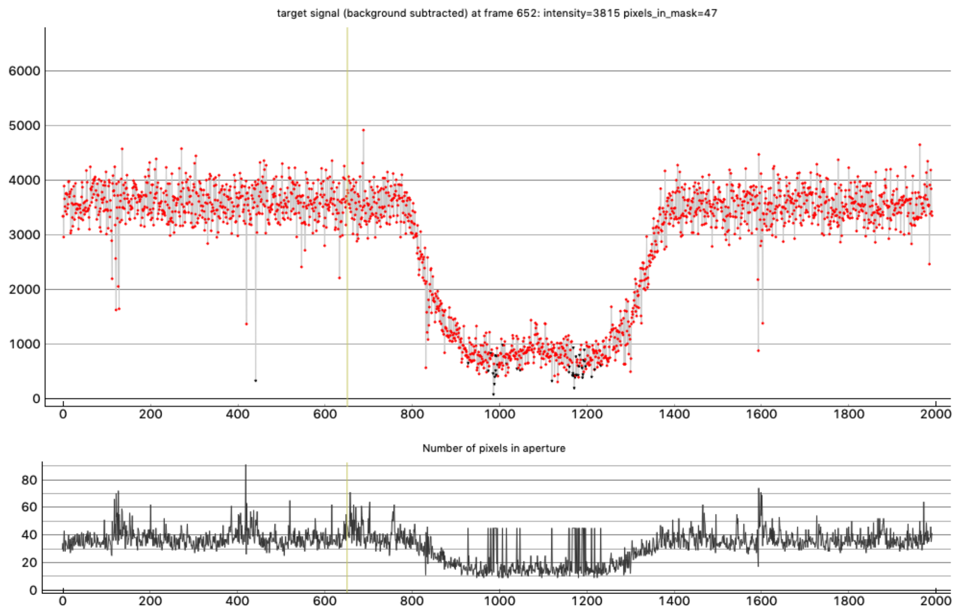
This results in better wind shake artifact suppression in the baseline, but those effects continue to appear frequently at the bottom of the occultation. But the larger mask has also caused a reduced/degraded SNR during the occultation itself.

We can see that a 3.3 pixel radius fixed circular aperture (37 pixels) is uncomfortably tight, producing a lot of wind shake artifacts, and that while an 8.8 pixel radius circular aperture (241 pixels) deals better with wind shake, it is at the expense of reduced SNR during the occultations. So one might then ask: What about a 'Goldilocks' compromise mask of 6.0 pixel radius containing 121 pixels? The resulting lightcurve is:



In this 'Goldilocks' lightcurve, wind shake effects are still apparent, but there is only a slight loss of gathered light (B is back to about 3800) and an improved SNR through the occultation region is obtained.

If we employ a similar 'Goldilocks' approach using adaptive masks for the target star, we get the example shown below. Here the target mask was generated at the 1.0 sigma clipping level (instead of the 0.5 sigma level used in a preceding example) resulting in a smaller mask (fewer pixels). The expectation/hope is that a higher SNR will be achieved during the occultation because of this tighter mask. The resulting lightcurve is shown below:



We do in fact get the hoped for improvement in SNR during the occultation. Qualitatively, this is the best of all the lightcurves so far, but the gathered light level (baseline value) decreases to about 3600 and there are a few more wind shake artifacts. In addition, there are more points at the bottom of the occultation where a mask could not be computed and so a default circular mask of 3.8 pixels radius (45 pixels) was substituted instead (indicated by the black dot instead of red in the upper sub-plot). Still, this seems to be an effective compromise: the wind shake artifacts are so obvious that they can be readily removed/ignored and this example has the best SNR in the more interesting and important parts of the occultation light curve.

In concluding this study:

> My experience in generating the examples for this discussion was that it was far less 'fiddly' to find an acceptable clipping level for generating the adaptive mask than selecting a good radius for a fixed circular mask.
>
> If a video does not have wind shake, the adaptive mask will always exhibit better SNR characteristics as an image gets dimmer.