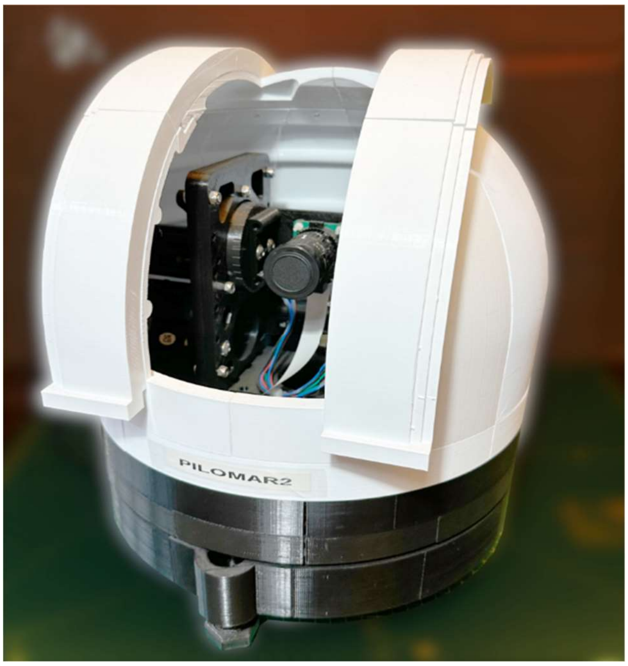
Pi-lomar manual

(DRAFT)

Contents



Overview

Pi-lomar is a 3D printed miniature observatory. The telescope uses a Raspberry Pi Hi Quality camera, linked to a Raspberry Pi single board computer. There is also an RP2040/RP2350 based microcontroller connected which controls the motion of the telescope in realtime while the Raspberry Pi handles higher level functions.

The software for Pi-lomar is written in Python3 and uses several freely available python packages to perform all the actions required.

Pi-lomar is a demonstrator to show what can be achieved with basic hardware and a 3D printer. It can capture interesting images of the night sky, but it is not a high resolution professional product.

Pi-lomar gathers multiple images of a selected target which can then be manipulated (stacked) separately after the observation is complete to produce an even more detailed image. Pi-lomar does not perform this image stacking.

This document accompanies the version released xxxx 202x src/pilomar.py version 1.1.0 circuitpython/code.py version 1.1.0 PCB version 2023-12-14 3D printed structure V13 Raspberry Pi 3B, 4B and 5 Pimoroni Tiny2040 and Tiny2350

What does Pi-lomar do?

Pi-lomar allows you to select an object in the night sky and track it as the sky moves, or even as the object moves against the night sky. While tracking an object it can then collect a set of photographs of the object. You can then download these photographs and process or combine them to produce more detailed astrophotography images of the night sky.

What does Pi-lomar NOT do?

- It does not automatically stack the images. It only captures the individual frames needed for you to perform stacking via some other package.
- Pi-lomar uses relatively wide angle budget lenses, it does not magnify tiny objects in the sky. It is best for capturing larger star fields and objects.
- Pi-lomar generally captures only standard camera wavelengths. By default it will not detect Infrared light for example. (You CAN modify your sensor to include some Infrared wavelengths though!)

What can Pi-lomar track?

Pi-lomar has the following lists of objects.

- Solar System objects
  The planets, plus the Moon
- Satellites
  Such as International Space Station and Chinese Space Station.
  (Note: Objects in Low Earth Orbit move VERY quickly, they may be too fast for the telescope to keep up.)
- Hipparcos catalog
  A list of over 100000 stars visible in the sky.
- Messier catalog
  A list of nebulae, asterisms and galaxies.
- Meteor shower catalog
  A list of common meteor showers.
- Comet catalog
  A list of regularly visiting comets.
- NGC catalog
  A more complete list of nebulae and galaxies.

It can also be given specific locations to track.

- RA-DEC co-ordinates
  A specific point in the night sky that moves as the Earth rotates.
  Use this for new targets that are not in any of the regular lists above.
- ALT-AZ co-ordinates
  A specific point in the sky that does not move. The telescope remains motionless taking photographs as the sky rotates past it.

What images does Pi-lomar generate?

Pi-lomar can generate JPG, DNG and FITS [(1)] images. It uses the operating system's default camera utility[(2)] to take photographs, including its built-in RAW options to retrieve the raw sensor data. Raw sensor data is stored in .DNG or .FITS files. The raw data can produce better stacked astro images.

When stacking images it is better to use the DNG or FITS [(1)] raw images. It is easier to view the .JPG images, but they have some compression and loss of detail.

[(1)] *FITS images can only be generated when running Pi-lomar on the 64bit Bookworm O/S. You must have the astropy package installed to support .FITS generation.*

[(2)] *Raspberry Pi operating systems provide either raspistill or libcamera utilities for dealing with the camera. Pi-lomar will work with both.*

Pi-lomar can also collect several types of control images to assist image stacking. Each image is stored in a specific folder for the observation session. You will have to take some additional steps to gather some of these images correctly. Your stacking software will tell you which types of images it wants you to gather.

- LIGHT IMAGES
  These are the main photographs of the target being observed.

- DARK IMAGES
  These are taken in the current observation conditions but with the lens cap on. These register the ‘noise’ in the sensor.
- FLAT IMAGES
  These are taken in daylight. These establish further characteristics of the camera such as vignetting, dust or dead pixels.
- DARK FLAT IMAGES
  These are taken with the lens cap on, but with the same exposure time as the FLAT images. Establishing more electrical noise characteristics of the camera.
- BIAS IMAGES
  These are taken with the lens cap on, but with a very fast exposure time. Measuring even more characteristics of the camera. Also known as OFFSET images.

It also generates a couple of image types to help with development/tracing.

- TRACKING IMAGES
  Pi-lomar uses a special method to check it is staying on target. It compares the latest image of the target against a theoretical image. It uses the difference between these two images to correct for any alignment issues and keep the target in view. Each time the calculation is performed some images are generated here to help with tracing.
- PREVIEW IMAGES
  Pi-lomar occasionally takes one of the target images and adds some scales and labels to the image to help you understand what you are looking at. These are preview images. At the end of the observation these images can be combined into a short AVI animation file.

This document contains information to help with the build and operation of the telescope.

Building Pi-lomar

The Instructables website contains the official build instructions for the project, however there are some additional details here to assist with your build.

Which bolts to use?

Generally they are all "M5 FLANGED BUTTON HEAD SCREWS" aka "Dome Allen Key Socket Bolts", their head has a wide low profile which fits into small spaces.


The majority are 20mm and 25mm lengths, a small number of 30mm, and 4 of the 12mm length in the camera cradle ... In many cases you can use 25mm lengths instead of 20mm, so stock up mainly on the 25mm ones and get a small number of the 12,20 and 30mm bolts. The M5 bolts take nylock nuts in most cases. One or two will be regular nuts if there's a space problem, like the nut insert in the platform build.

The exceptions are :-

(1) The bolts for attaching the Lazy Susan, those are M4x25mm recessed head bolts. The recess is so that they sit beneath the surfaces for easy movement. Nylock nuts again.

(2) The Stepper motors take M3 x 10mm screws to attach to the housing.

(3) The screws to attach the coupler to the azimuth drive wheel, you'll need to match the screw to the hole size on your particular couplers if you use them. I used narrow self tapping screws for that.

Using different stepper motors

The stepper motor is the NEMA17 2A 0.9Degree / step motor. You can use different motors if you change some configurations. Some of the alternative motor configurations are available via the Motor Tools menu, others will need changes to the project's parameter file.

If you have 1.8Degree motors you will find that the telescope will move TWICE as far as you intend whenever it is told to move. You can adjust for this by changing the AltitudeMotorStepsPerRev and AzimuthMotorStepsPerRev parameters. A 0.9degree motor has 400 steps, a 1.8degree motor will have 200 steps. Pi-lomar and the motorcontroller will adjust their calculations according to the parameter.

You can also configure microstepping which gives quieter but slower operation. Motor power may be reduced, you can compensate for this by increasing the current limiter on the DRV8825 drivers.

First run

The first time you run Pi-lomar it will generate the initial parameter file for the project. It will ask for your home longitude and latitude. You can have multiple lon/lat locations stored in case you want to operate the telescope in multiple locations.

The first time you run the program it will download and process a database of stars. This is the Hipparcos star catalog. On a RaspberryPi 3B with 32bit operating system it will take up to an hour constructing this catalog before you can start making observations. On a Raspberry Pi5 with a 64bit operating system it is much faster.

Whenever you start Pi-lomar it will ask you to select a target. You have a menu listing the different types of target. Select something simple, for example a SOLAR SYSTEM target which you know is visible. If your chosen target is not visible you will get a warning.

Then select the 'Begin Observation' option from the menu. This will make a first observation attempt with default settings. It's as simple as that! You do not have to collect any of the other control images (bias,flat,dark etc). You will have already captured some data you can work with!

When the observation finishes it will tell you which folder contains the images captured. You can run these through your choice of stacking software, or copy these onto another PC to process them there.

There are many more options on the menus, but you can get started very easily this way. You can later modify settings from the menu to change the exposure or the number of frames captured and many other behaviours.

Your first few runs are likely to reveal fine-tuning problems with your build, there are options on the menus to help you to fine-tune the telescope and get everything working as expected.

Pi-lomar main menu

Select target

Use this to change the target. See the target selection section of this document. This also runs automatically when you start the program.

Begin observation

Starts an observation with the current target and settings.

Status
Summary of the current settings and state of the telescope.

GOTO target
Telescope moves to point at the target but does not begin tracking or taking images. Useful sometimes for setting up the telescope at the start of an observation session.

Home all motors
Parks the telescope back at the home position. Pointing due south at the horizon.

Set exposure time
Set the exposure time in seconds for each individual frame captured. Values between 1e-6 and 200 seconds are allowed.

Set light batch size
Set the maximum number of images to capture in the session. This may not be reached if the observation has to end for some reason.

Set control batch size
Set the maximum number of images to capture for all the different control images that can be taken.

Take dark frame set
Capture the DARK FRAME control images.

Take flat frame set
Capture the FLAT FRAME control images.

Take bias/offset frame set
Capture the BIAS FRAME control images.

Take dark flat frame set
Capture the DARK FLAT FRAME control images.

Take preview frames
Take 'preview' images of whatever the camera is currently pointing at. Preview images include several extra markings and labels to assist with setting up the telescope and verifying what is being observed. (They are also captured automatically during normal observation runs)

Take auto frames
Capture fully automatic images through the camera. All other settings are ignored. This is useful for setting up, focusing and debugging the telescope during assembly.

Observation schedule menu
Maintain a schedule of observations to perform. The telescope can then run through the schedule relatively unattended until all the required observations have been made.

Tracking tools
Submenu of utilities for configuring the drift tracking mechanism. Pi-lomar has a rudimentary drift tracking solution to compensate for small setup or operational errors. You can see the current configuration together with the results of the latest tracking calculation. You can also modify some of the behaviours here.

Motor tools
Submenu of utilities for the motor control system. You can finetune the position of motors and exercise them from here.

Microcontroller tools
Submenu of utilities for the microcontroller that handles the motors. You can reset the microcontroller and turn on/off status LEDs from here. There are also tools to extract UART communication stream from the log files and display them in realtime or zip them for sharing.

Camera tools
Submenu of utilities for the camera handler. You can restart the camera process from here if you have problems.

Miscellaneous tools
Submenu of miscellaneous tools available in the software. Mainly functions related to parameter settings and general debugging of the system.

Development tools
Submenu of whatever development experiments are underway. You can view and edit parameters here.

Target selection
Pi-lomar comes with various lists of astronomical objects. If the object you want is not in the lists you can give the RIGHT ASCENSION and DECLINATION of the object, or even just the ALTITUDE and AZIMUTH to point at.

Pi-lomar always asks you to select a target at startup, you can then change the target at any time from the main menu.

Resume last observation
Pi-lomar remembers the last object you were observing, select this option to resume with the same target and settings.

Repeat earlier observations
Pi-lomar remembers all your recent targets and settings. This presents a list of those targets and shows whether they are currently visible. Choose any visible target to resume that particular observation.

Over time this list grows quite large so there is a cutoff limit (configurable in the Parameter file).

Solar system object
Pi-lomar uses a JPL list of the planets and the Moon, it also has data about the orbit of the International Space Station and the Chinese Space Station.

Hipparcos star catalog
The Hipparcos catalog lists many thousands of stars, if you know the HIP number of a particular star Pi-lomar can track it.

Messier catalog
Pi-lomar has a list of the Messier objects. If you know the catalog number (eg M101) you can choose those as targets.

New General star catalog (NGC)

The New General Catalog is also in Pi-lomar. You can choose deep space objects from the catalog via their NGC numbers. This catalog also includes many IC and PGC catalog items.

Comet

There is a catalog of comets in the system from the Minor Planet Center. New comets are constantly being discovered and known comets revised so the catalog will need updating from time to time. But it can track comets as they move through the sky. You will be warned at startup if the comet list is getting old.

Meteor shower

Pi-lomar has a catalog of common meteor showers. This sets the telescope in a special mode where it points to a fixed part of the sky and photographs the sky waiting for a meteor to pass. (16mm lens or wider is best for this). You can use the meteor detection utility to check if you caught any!

Aurora

This will switch to 'aurora' mode. It will point to the nearest pole and start gathering images in "FastImageCapture" mode. It will also generate a Keogram of the observation.

Space stations/satellites

A limited number of space stations/satellites are known. Objects in Low Earth Orbit like the ISS may move too quickly for the telescope to keep up.

RA-DEC coordinates

If you know the RIGHT ASCENSION and DECLINATION of an object you can ask Pi-lomar to track it even if it isn't in any of its catalogs.

Fixed ALT-AZ point

You can set a specific point around and above the horizon. It will not move as the night sky rotates, it will stay fixed on this point.

Suggest target

This lists potential targets chosen from various catalogs. It checks the visibility, size and brightness of the planets, Messier, NGC and Comet lists. It presents the potential targets in a list which you can select from.

You can also sort the list by entering '@' at the prompt. Each time you enter '@' the list of targets will be sorted by different criteria. Name, Azimuth, Altitude, Brightness and Size.

You can adjust the threshold for size and brightness via the parameter file.

Note: Comet brightness is an estimate based upon the position and characteristics in the MPC comet data. As comets approach the Sun their actual brightness can be considerably different from the estimate.

Searching

Some catalogs are very large (Hipparcos), and some have complex names (Comets). To help with searching through large lists Pi-lomar lets you enter partial values. It will present a refined list of matching entries and you can limit the search further until you find the item you want. If you enter an exact match search term that is selected.

Updating the catalogs

The catalogs are generally stored in the /data folder.

The NGC, Meteor and Messier catalogs are unlikely to change, you won't need to update those.

The solar system and Hipparcos catalogs will automatically download fresh copies if you delete the existing data files.

The SpaceStation orbit data is downloaded automatically from the celestrak website. It is cached locally on disc for a few days before being updated. To force a refresh, just delete the cache file in the /data folder.

The Comet catalog from the Minor Planet Centre is very frequently updated, and often the comet you want to see is a new one! If the comet is not already in the catalog you can download a fresh copy of the comet data file into the /data directory.

Observation schedule menu

This has basic options to schedule a series of observations to run automatically. This demonstrates a more unattended mode of operation.

Add target

When you start running the program the observation schedule is empty. You can add targets to the schedule here. You can select targets from all the usual options and catalogs.

You can add as many targets as you like to the schedule. Pi-lomar will work through the list performing each observation in the sequence that you added them to the schedule.

NOTE: Each entry in the schedule will use the current exposure time and frame count. You can change both of those from the main menu and also from the Schedule menu.

You can alternatively use the 'Load schedule' option to import a saved target list from disc. You can edit and rearrange this json file on disc using a conventional editor, then you can reload the edited schedule back into the program using the 'Load schedule' option.

Run schedule

This runs through the schedule performing an observation for each item. The list is executed in the same sequence that the entries were added.

Each observation will capture as many frames as the schedule dictates and will use the exposure time for the entry too. The observation ends when the frame limit is reached, or if the target goes out of range or sets.

The telescope will then move on to the next target in the list. If the next target is not yet in range (ie has not risen) then it will be skipped.

When all targets in the list have been processed the schedule is complete and control returns to the menu.

During a series of scheduled observations you can terminate an individual observation as usual with the 'x' key. The schedule will then move on to the next target in the list and resume there.

While executing a schedule, some features may be skipped, for example the telescope will not ask about generating preview movies as each observation completes. This is to allow the telescope to operate in a more unattended fashion.

Control images are not automatically captured during a schedule run. You must capture those manually before or afterwards.

Set exposure time

Use this to set the exposure time for 'light' images. Set this BEFORE adding a target to the schedule.

Set frame count

Use this to set the number of 'light' images to be captured for the target. Set this BEFORE adding a target to the schedule.

View schedule

View all the targets in the current schedule. If you want to change details in the schedule without starting again from scratch you can 'Save schedule', edit the .json file with an editor, then 'Load schedule' to reload it into the program.

Clear schedule

This will clear the schedule entirely.

Load schedule

This will load the schedule from disc. It uses a fixed filename in the /data folder. You can modify this file yourself with an editor if you need to before loading it back into the program.

You may also want to generate your own schedule list via your own external routines. The software will accept a schedule as long as the structure of the json file is correct.

Save schedule

This will save the current schedule to disc. It uses a fixed filename in the /data folder. You can then modify the file yourself with an editor, then load the modified version back into the program.

Sort by RA

By default the targets are observed in the same sequence that you added them to the schedule. You can reorder the schedule by Right Ascension, so that the targets are more likely to be captured as they follow each other across the sky.

Tracking tools menu

This has utilities to analyse and configure the drift tracking mechanism.

Tracking status

Show the current configuration of all the elements of the drift tracking mechanism.

Set tracking exposure time

Specify the exposure time to be used when capturing live images for the drift tracker.

Set search area

You can extend the search area that the drift tracker uses when matching the live view against an expected map of the sky. If your telescope is set up well you do not need a wide search area (set a value of 1.0). If your telescope set up is less precise you can extend the search area (max value 4.0).

This makes the 'map' larger on each axis. So a value of 1.0 creates a search map the same size as the camera view. A value of 2.0 creates a search map 2 times larger on each axis, ie 4 times the area of the camera view. A value of 4.0 will therefore create a search map 16 times larger than the camera view.

Set search overlap

The drift tracking tries to locate the live view within the search map. It starts in the centre of the search map and gradually works outwards towards the edges. Search overlap tells the algorithm how far to 'shift' the search area with each attempt.

10% means that only 10% of the previous search area is used in the next search. This performs fewer searches, each one covering more unique areas. So you search the entire map faster, but less thoroughly.

90% means that 90% of the previous search area is used in the next search. This performs many more searches, each one will only small changes from the previous. So you search the entire map more slowly but more thoroughly.

Set Latest Tracking filter

The drift tracking mechanism can apply enhancement filters to the 'latest live image' that it captures. You can choose which filter is applied here.

Test Latest Tracking filter

You can test the action of a "latest tracking filter" here.

Motor tools menu

This has utilities to test and tune the motors for the telescope.

About motors

Lists the configuration of the motors.

Home all motors

Move all motors to their home position.

Tune azimuth position

Use this to adjust the position of the motor if it is not matching the requested azimuth. Tuning a motor will move the motor a given number of 'steps', but the 'angle' registered in the telescope will not change. This is for 'correcting' positions.

Tune altitude position

Use this to adjust the position of the motor if it is not matching the requested altitude. Tuning a motor will move the motor a given number of 'steps', but the 'angle' registered in the telescope will not change. This is for 'correcting' positions.

Move azimuth to angle

Move camera to a specific azimuth position.

Move altitude to angle

Move camera to a specific altitude position.

Exercise azimuth motor
Move the azimuth motor between MIN and MAX positions to test the range of movement.

Exercise altitude motor
Move the altitude motor between MIN and MAX positions to test the range of movement.

Microstepping options
Submenu that allows you to select from various common microstepping configurations. You can adjust microstepping behaviour directly in the parameter file, but this sub menu will simplify the task for you.

Zip motor status log
Extract all the motor status messages from the current log file and store them in a separate ZIP archive.

Stop all motors
Immediately stop all motor movement.

Microstepping options
Submenu that allows you to select from various common microstepping configurations. You can adjust microstepping behaviour directly in the parameter file, but this sub menu will simplify the task for you.

About motors
List motor configuration details.

Current settings
This shows the current microstepping settings.

No microstepping
Both motors will move using ‘FULL STEPS’.
Advantages: FAST, GOOD STARTING POINT.
Disadvantages: NOISY, LOWEST PRECISION.

Slew 1/2, Observe 1/2
Both motors will move using ‘HALF STEPS’.
Advantages: SMOOTHER MOVEMENT, REASONABLY FAST
Disadvantages:

Slew 1/4, Observe 1/4
Both motors will move using ‘QUARTER STEPS’.
Advantages: EVEN SMOOTHER MOVEMENT
Disadvantages: NOTICEABLY SLOWER.

Slew full, Observe 1/2
GOTO/HOME movements use FULL STEPS.
Observation movements use HALF STEPS.
Advantages: FAST GOTO/HOME. SMOOTHER OBSERVATIONS.
Disadvantages: Noisy GOTO/HOME movements.

Slew full, Observe 1/4

GOTO/HOME movements use FULL STEPS.
Observation movements use QUARTER STEPS.
Advantages: FAST GOTO/HOME. EVEN SMOOTHER OBSERVATIONS.
Disadvantages:

Slew 1/2, Observe 1/4

GOTO/HOME movements use HALF STEPS.
Observation movements use QUARTER STEPS.
Advantages: REASONABLE FAST GOTO/HOME. SMOOTHER OPERATION.
Disadvantages:
Recommended: For 400 step 0.9Degree motors.

Slew 1/4, Observe 1/8

GOTO/HOME movements use QUARTER STEPS.
Observation movements use EIGHTH STEPS.
Advantages: VERY SMOOTH MOVEMENT.
Disadvantages: SLOW GOTO/HOME MOVEMENT.
Recommended: For 200 step 1.8Degree motors.

Microcontroller tools menu

Utilities to help with testing and controlling the microcontroller.

About motorcontroller

Display statistics and status of the microcontroller and communications.

Restart microcontroller

This will restart the microcontroller. Either a hard reset or a soft reset depending upon how the microcontroller is powered.

Start message handler

The message handler starts automatically, however you can restart it here.

Shutdown message handler

The message handler shuts down automatically as the software closes, but you can also do it from here.

Flush command queue

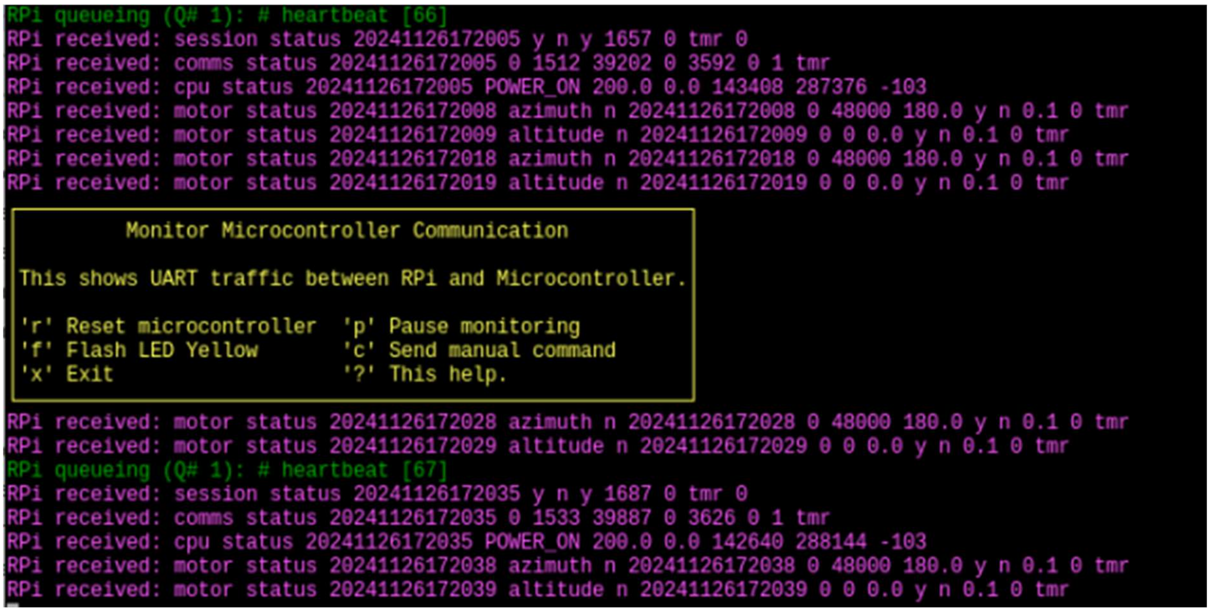
Remove all commands from the RPi to the microcontroller that have not yet been sent.

Zip comms log

Extract communication messages to/from the microcontroller from the log file and write them to a separate log file. This is also zipped.

Monitor communication

Open a monitor to show the realtime communication between the RPi and the microcontroller. You can also perform some special functions from here.



Messages sent from the RPi to the Microcontroller are shown in GREEN.
Messages received by the RPi from the Microcontroller are shown in FUSCHIA.

Microcontroller LEDs on

By default the microcontroller's RGB LED is active. If you have turned it off, you can turn it back on here.

Microcontroller LEDs off

By default the microcontroller's RGB LED is active. You can turn it off here to reduce stray light within the observatory dome.

Microcontroller GPIO power ON

You can turn ON the GPIO controlled power to the microcontroller here.
WARNING: This can override the software protection for preventing conflicting power supplies mixing if both GPIO and USB power is provided to the microcontroller.

Microcontroller GPIO power OFF

You can turn OFF the GPIO controlled power to the microcontroller here.
WARNING: If the microcontroller is powered by a USB cable, the microcontroller will NOT power down until you remove the USB cable.

Camera tools menu

Utilities for the camera.

About camera

Display configuration and status of the camera.

Choose image types

Pi-lomar can save images in multiple formats. You can choose the image formats here. This also automatically selects the correct camera driver to support your choice of image types. This is easier and safer than changing the parameter file manually. The system should only offer you choices that your O/S will support which reduces the potential for configuration errors.

Choose capture mode

The camera normally captures and fully processes images on at a time. You can switch to FAST capture mode where the camera captures the images but does no immediately process them. Use this when gathering lots of images as quickly as possible.

Start camera thread

The camera handler starts automatically in its own thread. You can restart it here if you need to.

Stop camera thread

The camera handler thread will shut down automatically as the software closes. You can shut it down earlier here.

Sensor cleanup off

You can turn off the on-chip image cleanup here.

Sensor cleanup on

You can turn on the on-chip image cleanup here.

Auto detect camera

Re-run the routine which checks that the camera is attached and responding. This will ENABLE/DISABLE the camera as required.

Choose lens

Common lenses are listed here, this lets you select the correct parameters for 16mm and 50mm lenses. (Restart required after changing)

Calibrate FoV

If you are using a different lens this will let you estimate the field of view and adjust the lens parameters in the software to match.

Process image files

If you have used FAST CAPTURE MODE. You can complete the image processing here. Run this option at the end of the observation. It will finish extracting DNG files etc.

Build keogram

Constructs a keogram for the observation data captured.

Enable camera

Manually enable the camera.

Disable camera

Manually disable the camera. Pi-lomar will create simulated images instead.

Miscellaneous tools menu

Miscellaneous utilities are here.

About system

Show status and statistics about the system.

Show parameters
Show the current state of all the parameters.

Edit parameters
Edit the parameter file in the nano editor. This is a convenience utility to help you find and edit the file. You will need to restart the software after editing to make all the new parameter settings take effect.

Set local timezone
Information only. You can set your local timezone here. Pi-lomar operates in UTC. Setting this value currently does not change the behaviour of the software. This is for future developments.

Edit target history
Pi-lomar remembers the last few observations you have made. This makes it easier to repeat those observations if you need to restart the system or if you are capturing images over multiple nights. You can edit the history file here in the nano editor.
NOTE: This is a 'JSON' format file, take care if you edit it to retain the correct structure.

Show folder structure
This shows the different folder names that the current observation will use. You can see where the different types of images and files will be stored.

Debug mode on
This is the same as using the 'd' key during an observation. With debug mode ON, the dashboard is removed, you get a simple scrolling display of major actions. This makes it much easier to see errors if you are tracing a problem.

Debug mode off
This is the same as using the 'd' key during an observation. With debug mode OFF, the dashboard is displayed during an observation. This shows a lot of information about the operation of the telescope, but can make unexpected error messages difficult to see.

Choose color scheme
By default pi-lomar uses a GREEN ON BLACK color scheme. You can choose from a few preset alternatives here. WHITE, RED, GREEN and BLUE are available.

Choose individual color
If the preset color schemes are not suitable for your display you can edit individual colors directly here. There are 255 different colors available.

Development tools menu
You will find tools here related to developing the pi-lomar software. They may be useful if you are adapting the software yourself.

Create overlay
This will generate a .png file with a transparent background. This can be used as an overlay to layer on top of your photographs to mark significant stars. This is similar to the 'preview' image, but you can use it in your own image editing software.

Satellite passes

If the current observation target is from the satellite list, this will show probable RISE/SET times for the next few days. NOTE: The satellite may not be visible for the entire pass, it needs to reflect sunlight too. (Experimental feature)

Check / remount storage

At startup the software will try to detect any USB storage available. If found it will store images on the USB storage rather than the system SD Card. You can re-run the USB storage check here.

Zip trajectory log

This will extract the trajectory segments from the log file and save them as a separate zipped file. This helps when developing the trajectory mechanism.

GPIO status

You can see the status of the GPIO pins here.

Show parameters

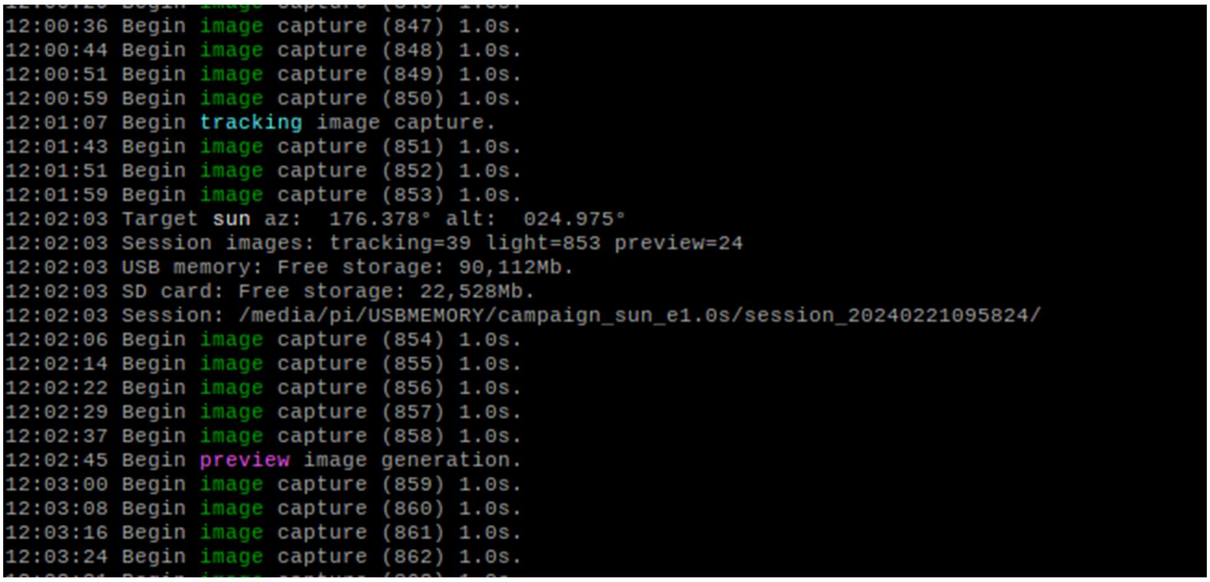
This will show the current parameter settings.

Edit parameters

You can edit the parameter file here using the nano editor. This is a convenience function, if you change the parameter file you will still need to restart the software for your changes to take full effect.

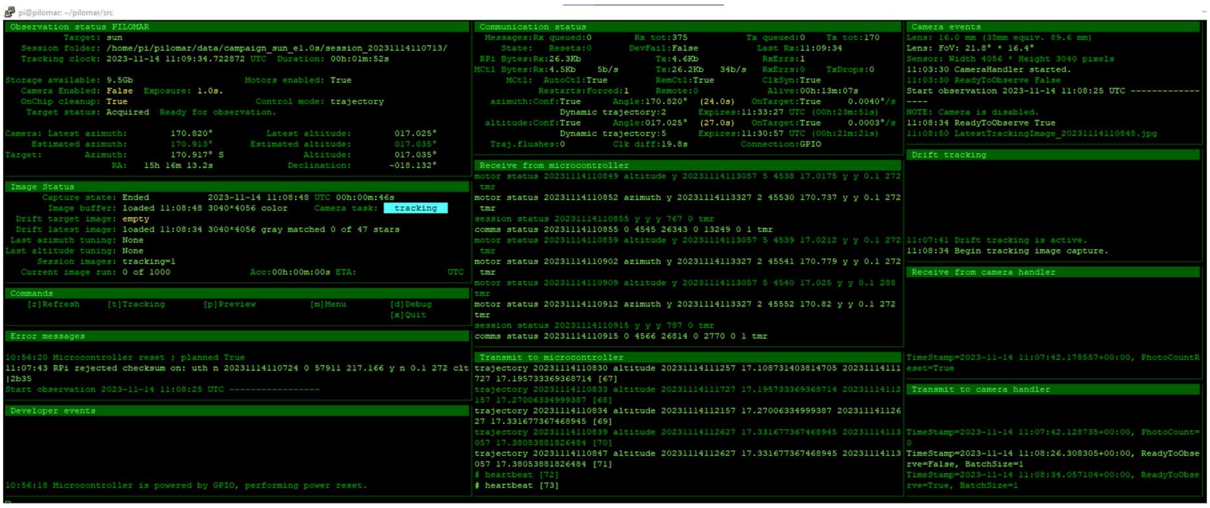
Pi-lomar observation dashboard

When you are in an observation run there are two display types available. By default Pi-lomar runs in 'debug mode' – which just lists key items to the screen as they occur. This makes it easy to spot error messages. There is no GUI for Pi-lomar at the moment.



```
12:00:36 Begin image capture (847) 1.0s.
12:00:44 Begin image capture (848) 1.0s.
12:00:51 Begin image capture (849) 1.0s.
12:00:59 Begin image capture (850) 1.0s.
12:01:07 Begin tracking image capture.
12:01:43 Begin image capture (851) 1.0s.
12:01:51 Begin image capture (852) 1.0s.
12:01:59 Begin image capture (853) 1.0s.
12:02:03 Target sun az:  176.378° alt:  024.975°
12:02:03 Session images: tracking=39 light=853 preview=24
12:02:03 USB memory: Free storage: 90,112Mb.
12:02:03 SD card: Free storage: 22,528Mb.
12:02:03 Session: /media/pi/USBMEMORY/campaign_sun_e1.0s/session_20240221095824/
12:02:06 Begin image capture (854) 1.0s.
12:02:14 Begin image capture (855) 1.0s.
12:02:22 Begin image capture (856) 1.0s.
12:02:29 Begin image capture (857) 1.0s.
12:02:37 Begin image capture (858) 1.0s.
12:02:45 Begin preview image generation.
12:03:00 Begin image capture (859) 1.0s.
12:03:08 Begin image capture (860) 1.0s.
12:03:16 Begin image capture (861) 1.0s.
12:03:24 Begin image capture (862) 1.0s.
```

You can start/stop debug mode by pressing the 'd' key during the observation. When you stop debug mode, the screen is cleared and a dashboard is presented instead.

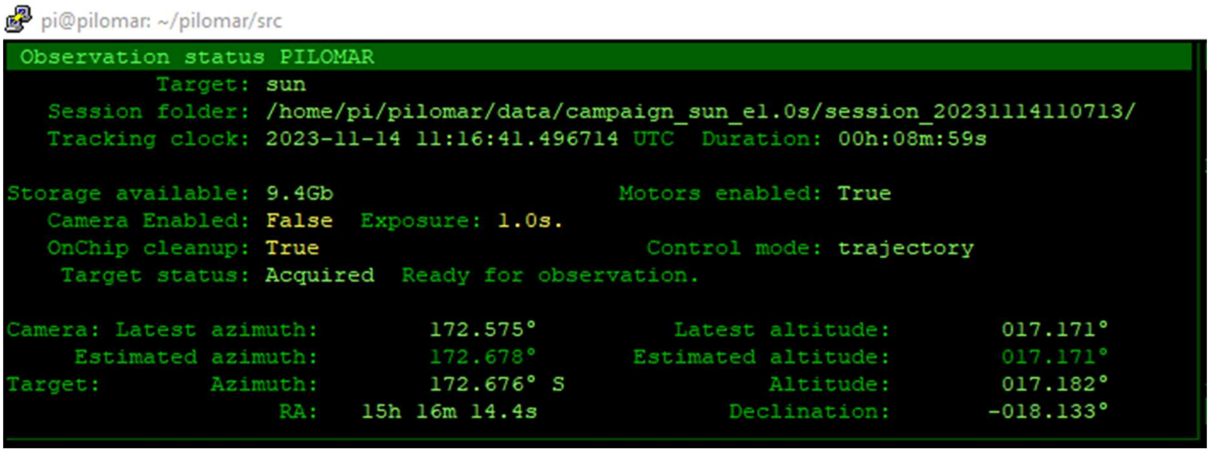


The dashboard shows a group of character based windows. The number of windows shown depends upon the size of the terminal window. If you maximise the window you will get the most information shown. If you have a smaller window open then you will only see basic information. You can also adjust the font size on your terminal widow to show more or less information. The display always shows the most important information, even for quite small terminal windows. The dashboard reacts to the size of the terminal window dynamically.

Pressing the ‘d’ key will switch the dashboard display back to the simpler debug display.

This display is updated by Pi-lomar at the end of each loop by the main process. You can see information about the computer, camera, target and telescope on these displays. You can also view the communication between the main program and other processes.

PILOMAR main



Target

The name of the selected observation target.

Session folder

The location where all the images and related files are stored for the observation run.

Tracking clock

Shows the current UTC time that the telescope is using to track the object. It is followed by the elapsed time since the start of this observation run.

Storage available

The amount of storage available for saving images and related files on the Raspberry Pi. The observation will automatically stop when the memory falls below about 500Mb. The storage shown is for the mounted USB memory stick if available, otherwise it's the main SD card.

Camera enabled

Indicates that the camera is enabled. The program will run without the camera enabled, it will generate fake images instead. The camera is disabled automatically if it is not detected at startup.

Exposure

Exposure time that the camera is using. If pilomar is using raspistill to take the images, it can take twice the exposure time to capture an individual image.

OnChip cleanup

The Sony sensor in the camera performs some image cleaning automatically. This can degrade the raw data that is used for astrophotography. You can turn this off in order to get more pure raw data from the sensor. (See the 'DisableCleanup' parameter)

Control mode

Shows how the telescope is being controlled. 'trajectory' means that the trajectory of the target has been calculated and the telescope is automatically following the trajectory.

Target status

Tells whether the telescope is on target or not. 'Acquired' indicates that the telescope is positioned on the target. Observations cannot start until the target is acquired AND the trajectory is known.

Camera Azimuth/Altitude

The last reported angles for the camera. These are the positions that the motor controller reports back to the Raspberry Pi regularly. This is updated periodically.

Estimated Azimuth/Altitude

An estimation of the current position of the camera in real-time based upon the last reported position from the camera.

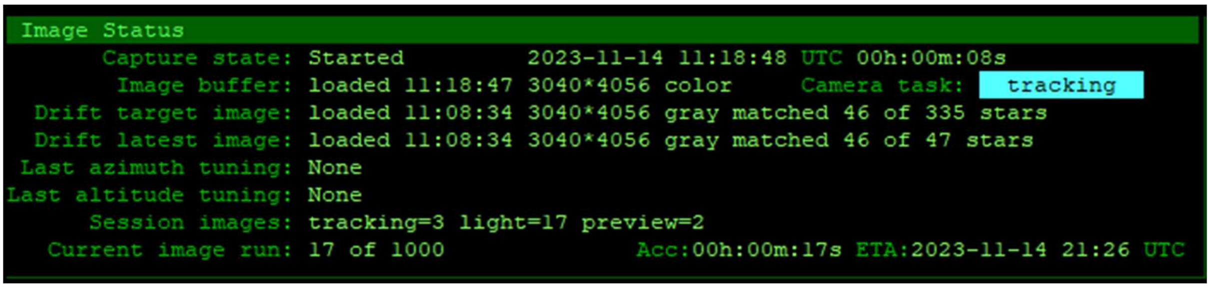
Target Azimuth/Altitude

The currently calculated angles for the observation target. It is normal for the camera angles above to be delayed slightly from these angles because of the communication delay from the microcontroller.

Target RA/Declination

The currently calculated astronomical location of the target. This is converted into the Azimuth/Altitude positions based upon your Observer position and the tracking clock.

Image status



Capture state

Shows whether the camera is actually taking an image or not. After each camera image is captured the Raspberry Pi must perform a number of other tasks. The line also shows the UTC time when the camera entered the capture state, and the elapsed time.

Image buffer

This shows when the camera's image was loaded into the Raspberry Pi's image buffer for processing. It also shows the time, dimensions and color/grayscale format of the image.

Camera task

The camera handler performs multiple tasks, it captures images but also performs optical tracking, and other tasks. This shows which particular task is being done.

Drift target image

When performing optical tracking, the camera handler calculates an 'expected' image of the stars and target. This shows the state of the expected image, and a measure of the 'matching' between the expected image and the actual image.

> matched aaa/bbb stars
>
> bbb is the number of stars found in the expected image. This is calculated by pi-lomar.
>
> aaa is the number of stars found in BOTH the expected and actual images when they are compared. This is calculated by astroalign.

Drift latest image

When performing optical tracking, the camera handler takes a special image of the target designed to match the 'expected' image it has calculated. This shows the state of this special image. It also shows a measure of the 'matching' between expected and actual images.

> matched aaa/bbb stars
>
> bbb is the number of stars found in the actual image. This is calculated by pi-lomar.
>
> aaa is the number of stars found in BOTH the expected and actual images when they are compared. This is calculated by astroalign.

Latest azimuth tuning

If the optical tracking indicates that the telescope is drifting off target, the camerahandler can ask the motorcontroller to correct the position of the telescope. This line shows that latest correction to the azimuth of the telescope.

Latest altitude tuning

If the optical tracking indicates that the telescope is drifting off target, the camerahandler can ask the motorcontroller to correct the position of the telescope. This line shows that latest correction to the azimuth of the telescope.

Session images

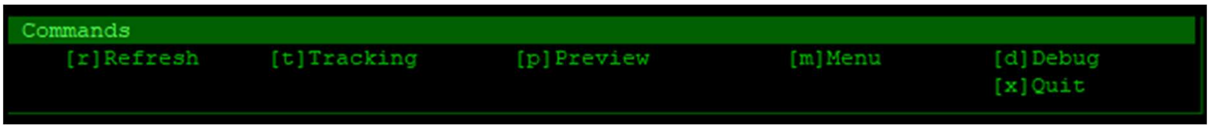
The telescope gathers several types of images. This summarises how many of each image have been captured. Not all features are available in all versions of the pilomar software, or they may be disabled.

- tracking = The number of TRACKING images captured.
- light = The number of LIGHT images generated. These are the ones you will use in your astrophotography stacking program.
- preview = The number of 'preview' images generated. These are samples of the telescope's view with useful markings to help identify what the telescope is looking at.
- …

Current image run

How many LIGHT images have been fully captured, and how many in total were requested. The 'acc' field shows the accumulated exposure time for all the images captured so far. The ETA estimates the date/time when the entire set will have been gathered.

Commands

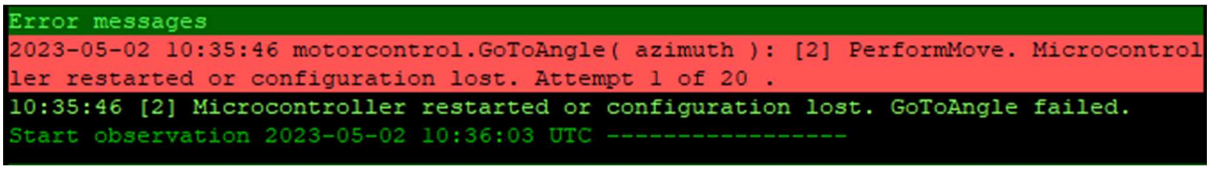


A summary of the command keys that the program recognises while an observation is underway.

- [x] To quit back to the menu.
- [r] To refresh the display if it gets corrupted for some reason.
- [m] To view a submenu of some actions you may want to perform DURING an observation.
- [d] Debug mode control. If you turn DEBUG ON then the display is cleared and a more simple scrolling text display is used. This is useful for catching error messages if you are chasing a problem. If you turn DEBUG OFF then the full screen display is shown, but there is a risk that error messages get lost as the display refreshes very often.
- [+] Increase exposure time. You can dynamically increase the exposure time by pressing the '+' key. This will DOUBLE the current exposure time.
- [-] Decrease exposure time. You can dynamically decrease the exposure time by pressing the '-' key. This will HALVE the current exposure time.
- [t] Drift Tracking on/off. You can enable/disable drift tracking during an observation with this command. Sometimes it is useful when setting up, or when dealing with difficult images that the tracking does not recognise well.

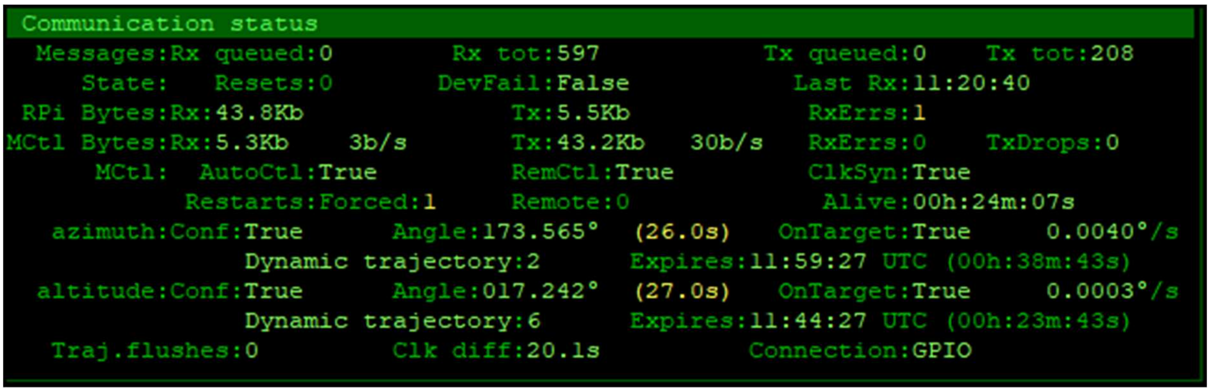
- [p] Preview. The telescope generates preview images at regular intervals, if you need to generate a preview image more quickly use this command. It will trigger an immediate preview (once the current camera task has completed).
- [u] UTC. Pi-lomar works internally in UTC, you can show dates and times in UTC or your Local time zone (See LocalTZ) by pressing this key.
  NOTE: Timestamps in internal messages between processes are always shown in UTC. Only the summary status displays can switch between UTC and LocalTZ.

Error messages



Scrolling text window showing recent error messages or important events that have occurred. Many error messages are also recorded in the log file, and will be summarised again when the observation ends.

Communication status



The Raspberry Pi has overall control of the telescope, but it uses a microcontroller to control the motors and position the camera. The RPi and motorcontroller communicate through a UART connection. This window shows details about the UART comms.

Generally the RPi tells the microcontroller what to follow and what trajectory it will take. The microcontroller then reports back how it is following the target.

Messages Rx queued

The number of messages that have been received from the microcontroller but have not yet been processed by the RPi.

Rx total

The total number of messages received from the microcontroller.

Tx queued

The number of messages waiting to be sent to the microcontroller.

Tx total

The total number of messages sent to the microcontroller.

State Resets

How many times has the microcontroller reset?

Dev fail

True indicates that the microcontroller has reset too many times, or failed to communicate at all. In which case the RPi considers that the device has failed. No observation is possible. Investigation is needed.

Last Rx

The time when the last message was received from the microcontroller.

RPi Bytes Rx

The number of bytes received by the RPi from the microcontroller.

Tx

The number of bytes send from the RPi to the microcontroller.

Rx Errs

The number of messages that the RPi received from the microcontroller, but rejected them. They are rejected if a simple checksum error is identified. Usually as a result of a reset.

MCtl Bytes Rx

The number of bytes that the microcontroller reports it has received.

Tx

The number of bytes that the microcontroller reports it has sent.

RxErrs

The number of messages that the microcontroller received from the RPi, but rejected them. They are rejected if a simple checksum error is idendified. Usually as a result of a reset.

TxDrops

The number of messages that the microcontroller had to delete instead of sending them to the RPi. This only happens if the microcontroller wants to send too many messages and the communication would be overloaded. In those situations the microcontroller starts to drop the oldest messages instead of transmitting them.

Mctl AutoCtl

The guiding state of the microcontroller. When TRUE it shows that the microcontroller has enough information to automatically control the telescope itself.

RemCtl

When TRUE it shows that the microcontroller has enough information for the RPi to issue trajectories to it.

ClkSyn

Indicates that the RPi and microcontroller have synchronised their clocks. The microcontroller cannot start following targets until it has the correct time.

Restarts Forced

The number of times that the RPi has forced the microcontroller to reset. These are not always the result of errors, they can be deliberate restarts just to clear out previous trajectories and targets that the microcontroller was following.

Remote

The number of times that the microcontroller has reported a reset.

Alive

How long has the microcontroller been alive since the last reset.

Azimuth Conf

Indicates that the configuration of the azimuth motor is complete on the microcontroller. The RPi passes details about the motor, and its initial position at startup.

Reported angle

The last reported angle of the azimuth motor.

OnTarget

Does the microcontroller consider that the azimuth is on target?

Dynamic trajectory

How many trajectory elements does the microcontroller have for the azimuth motor.

Trajectory expires

When do the trajectory elements expire for the azimuth motor.

Altitude Conf

Indicates that the configuration of the altitude motor is complete on the microcontroller. The RPi passes details about the motor, and its initial position at startup.

Reported angle

The last reported angle of the altitude motor.

OnTarget

Does the microcontroller consider that the altitude is on target?

Dynamic trajectory

How many trajectory elements does the microcontroller have for the altitude motor.

Trajectory expires

When do the trajectory elements expire for the altitude motor.

Traj.flushes

If the communication with the microcontroller fails for some unexpected reason, the microcontroller may continue to follow the planned trajectory even if the main program has stopped working. If the microcontroller does not get any communication from the main program for more than 120 seconds then it will flush the trajectory for safety. This will stop the camera moving. When communication resumes the trajectory will be reloaded and movement will resume. Each time the trajectory is flushed, this flush count increments.

Clk diff

An estimate of how closely the RPi and Microcontroller clocks are synchronised, generally the synchronisation improves the longer the observation is running. A difference of one or two seconds is quite acceptable.

Connection

The microcontroller can be powered via the GPIO ports or via the USB port. There are risks with conflicting power supplies to the microcontroller if BOTH connections exist at the same time. Pi-lomar will try to identify this situation and warn here. Normally you expect to see just “GPIO” reported here. If a USB connection is detected then pi-lomar will handle power and resets differently to try to reduce power conflicts.

Receive from the microcontroller

```
Receive from microcontroller
motor status 20230103194610 altitude y 20230103200818 6 14405 54.0187 y y 0.1 272
session status 20230103194617 y y y 8347
comms status 20230103194617 0 11931 232943 0
motor status 20230103194620 azimuth y 20230103200733 8 36000 135.0 y y 0.1 304
motor status 20230103194620 altitude y 20230103200818 6 14410 54.0375 y y 0.1 288
motor status 20230103194630 azimuth y 20230103200733 8 36015 135.056 y y 0.1 288
motor status 20230103194630 altitude y 20230103200818 6 14414 54.0525 y y 0.1 304
controller log :20230103194634:trajectory.Clean: Expired ( azimuth 20230103194333 13
4.045 20230103194633 135.079 )
session status 20230103194637 y y y 8367
comms status 20230103194637 0 11931 233503 0
motor status 20230103194640 azimuth y 20230103200733 7 36030 135.112 y y 0.1 288
motor status 20230103194640 altitude y 20230103200818 6 14419 54.0712 y y 0.1 304
```

The actual messages received from the microcontroller.

Transmit to the microcontroller

```
Transmit to microcontroller
55.790088800108684 [130]
trajectory 20230103194141 azimuth 20230103200133 140.4376901310085 20230103200433 14
1.54822722009882 [131]
trajectory 20230103194421 altitude 20230103200418 55.790088800108684 20230103200818
56.148593912145266 [132]
trajectory 20230103194440 azimuth 20230103200433 141.54822722009882 20230103200733 1
42.67177613736771 [133]
```

The actual messages sent to the microcontroller

Camera events

```
Camera events
18:32:19 LatestTrackingImage_20230620183218.jpg
18:33:47 TargetTrackingImage_20230620183347.jpg
18:33:56 TrackingAnalysis_20230620183356.jpg
18:34:45 light_20230620183356_00.jpg
18:35:37 light_20230620183447_00.jpg
18:36:14 preview_20230620183540.jpg
18:37:04 light_20230620183615_00.jpg
```

Major events performed by the camera handler. Including each image generated.

Drift Tracking

```
Drift tracking
19:37:01 Updating drift calculation for tracking.
19:37:01 Drift result: -16,9 px; -7,-4 steps.
19:37:01 Not tuning azimuth, drift is too small.
19:37:01 Not tuning altitude, drift is too small.
19:42:15 Begin tracking image capture.
19:42:53 Updating drift calculation for tracking.
19:42:53 Drift result: 2,0 px; 0,0 steps.
19:42:53 Not tuning azimuth, drift is too small.
19:42:53 Not tuning altitude, drift is too small.
```

Messages from the optical drift tracking process that Pi-lomar uses.

Receive from camera

```
Receive from camera
, ObsEnd=2023-01-03 19:43:52.731259+00:00, ObsTime
=0:00:58.527393, RunThread=True
TimeStamp=2023-01-03 19:44:52.426657+00:00, PhotoC
ount=83, ObsStart=2023-01-03 19:43:53.053847+00:00
, ObsEnd=2023-01-03 19:44:52.410417+00:00, ObsTime
=0:00:59.356570, RunThread=True
TimeStamp=2023-01-03 19:45:51.851990+00:00, PhotoC
ount=84, ObsStart=2023-01-03 19:44:52.810079+00:00
, ObsEnd=2023-01-03 19:45:51.845921+00:00, ObsTime
=0:00:59.035842, RunThread=True
```

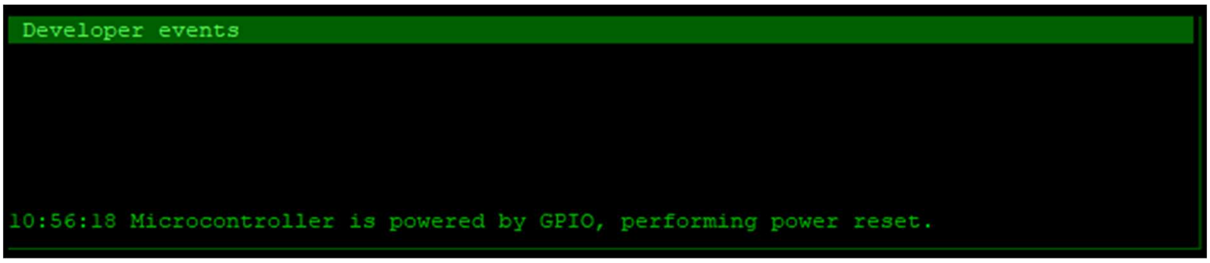
Messages received by the main loop from the camera handler thread. Mainly this is reporting the progress of images captured.

Transmit to camera

```
Transmit to camera
TimeStamp=2023-01-03 17:52:54.469700+00:00, ReadyT
oObserve=False
Stop=True
TimeStamp=2023-01-03 18:04:12.032061+00:00, PhotoC
ount=0
TimeStamp=2023-01-03 18:04:34.404431+00:00, ReadyT
oObserve=False, BatchSize=1
TimeStamp=2023-01-03 18:04:40.593089+00:00, ReadyT
oObserve=True, BatchSize=1
```

Messages sent from the main loop to the camera handler thread. Mainly this is telling the camera thread when to start/stop image capture.

Developer events



For development purposes there is a window which shows development messages. Normally you won't see much appear here.

Practical information

File structures

If a USB memory stick is attached, the images are stored there, otherwise they are stored on the Raspberry Pi's SD card.

All Pi-lomar related files are stored under /home/pi/pilomar/

Image and object data files are stored under /home/pi/pilomar/data

Each target is given its own folder in the /data folder.

For example 10second exposures of Mars will be stored under
/home/pi/pilomar/data/campaign_Mars_e10s/
The _e10s represents the exposure time.
If you capture exposures of Mars over several different sessions, all images will be stored under this folder if the exposure time is the same. The target + exposure time is a unique 'campaign'. You can add to a campaign over several nights if you repeat the same settings.

Each individual observation run is then stored in a subfolder.
/home/pi/pilomar/data/campaign_mars_e10s/session_yyyymmddhhmmss
This is the UTC timestamp when the observation begins. These are observation 'sessions'. They are all stored within the same campaign (Target + exposure time).

Within each individual session the different images types are separated into folders.

- /home/pi/pilomar/data/campaign_mars_e10s/session_yyyymmddhhmmss/bias
  The JPG and DNG bias images are stored here.
- /home/pi/pilomar/data/campaign_mars_e10s/session_yyyymmddhhmmss/dark
  The JPG and DNG dark images are stored here.
- /home/pi/pilomar/data/campaign_mars_e10s/session_yyyymmddhhmmss/flat
  The JPG and DNG flat images are stored here.
- /home/pi/pilomar/data/campaign_mars_e10s/session_yyyymmddhhmmss/light
  The JPG and DNG light images are stored here. These are the main observation images.
- /home/pi/pilomar/data/campaign_mars_e10s/session_yyyymmddhhmmss/preview
  The JPG preview files and summary AVI animation are stored here.
- /home/pi/pilomar/data/campaign_mars_e10s/session_yyyymmddhhmmss/tracking
  The JPG tracking performance images are stored here.

If you want to use a different structure you can edit the CreateFolderList() function in pilomar.py to generate different directories as you require. Some astrophotography image stackers expect specific folder structures when you process the images.

Parameters

Pi-lomar stores many parameters in a json file in the /data directory.

The parameter file is generated automatically when you first run pilomar. It will be populated with default settings. These defaults should be good for operation with the standard 16mm lens. But you can adjust many behaviours if needed.

The parameter list evolves with new versions of the software. If you update the pi-lomar software version your previous parameter settings are retained. You can reset the parameters to their defaults simply by deleting the parameter file and restarting the software. If newer software versions add or remove parameters, they will be automatically added/removed from the parameter file too.

The parameter filename is /home/pi/pilomar/data/pilomar_params.json

You can edit the parameter file with most text editors, if you change any parameters it is strongly recommended to completely restart the pilomar software AND the microcontroller. Configurations are shared across the two devices and only a clean restart guarantees that parameter changes are correctly passed everywhere correctly.

StepperDriverData

Python dictionary of recognised stepper driver boards and the microstepping options available. Used to define the mode signals that the microcontroller uses to trigger the correct type of movement.

ImagePrivacy

Default ‘high’: The pilomarfits camera handler can store metadata in the headers of .jpeg and .fits files. You can decide how anonymous to make the metadata.

- ‘high’: Observer’s lat and long are set to 0. Owner name is anonymised.
- ‘med’: Observer’s lat and long are rounded to nearest whole degree. Owner name is anonymised.
- ‘low’: Observer’s lat and long are rounded to nearest 1/10th of a degree. Owner name is included.

Owner

Default ‘telescope owner’. You can optionally give your name or a user-id that can be included in .jpeg & fits image metadata tags. (Only applies to images captured with the pilomarfits camera handler.

BoardType

Default None. Reference only parameter. Somewhere to store the motorcontroller PCB board type. It is available if logic has to support different capabilities and behaviours.

BatchSize

Default 100. Specifies how many LIGHT images are captured in each observation session. The session ends when this number of images have been taken, or if some other reason occurs to terminate the session.

UARTOverride

Default None. This forces a specific UART port to be used for communication between the RPi and the microcontroller. If left as None value, the software will choose the port based upon the standard pi-lomar design and the RPi model in use. If you set this parameter to a string such as ‘/dev/ttyAMA0’ then that port is opened for UART communication regardless of hardware. Use this if you have a special motorcontrol circuit design for example.

ControlBatchSize

Default 20. Specifies how many BIAS, FLAT, DARK images to capture. You do not need to capture as many control images as you do for LIGHT images. You do not improve the image quality much if you increase this.

ColorScheme

Default ‘green’. ‘white’,’red’,’green’,’blue’. Different color schemes for the observation dashboard. Choose a color scheme that works well for you at night.

CameraEnabled

Default True. Turn on/off the camera. If the software cannot detect the camera at startup this will automatically switch to False, disabling the camera. You must manually return it to True when you have connected the camera successfully.

When False, pilomar generates simulated images instead. Use this for development.

DisableCleanup

Default True. When True, pilomar will disable the on-chip cleanup process on the camera sensor. This cleanup can degrade image quality for astrophotography work, although generally you will not see much difference.

NOTE: This only applies when raspistill camera utility is in use, eg when running pilomar on Raspbian Buster O/S. If you are running libcamera under Bookworm or later this parameter is not used. You can control cleanup by editing the camera command templates in the parameter file instead. Add or remove the option ‘--denoise off’ in each command line instead.

BacklashEnabled

Default False. If you have problems with ‘backlash’ in the telescope motors and gears (ie ‘slack’ in the movement) you can set this to True. Pi-lomar sends slightly different movement instructions to the telescope to try to compensate for the backlash. This is not generally required because the tracking system will correct for any significant drive differences.

To handle backlash, the telescope recognises when the motors change direction. In these cases the motor is sent slightly BEYOND the target position and then driven BACK to the real target position in an attempt to absorb the slack in the motors/gears. This makes it easier for the drift calculation to then adjust for any final errors caused by the backlash.

Fault sensitive

Default. False. If TRUE the microcontroller listens to the FAULT signal from the DRV8825 driver chips. If a fault signal is received then the observation is aborted. During development it is sometimes useful to disable this. The DRV8825 will report a fault if it overheats or there is a power problem.

MctlLedStatus

Default True. When set to False, pilomar instructs the motorcontrol board to turn off any LED status lights. When set to True the motorcontrol board will indicate activity by flashing various colours through the RGB LED.

ObservationResetsMctl

Default False. Set to True to force the microcontroller to reset at the start of every observation.

MctlResetPin

Default 4. The Raspberry Pi GPIO pin that performs the microcontroller reset.

UartRxQueueLimit

Default 50. Messages received from the UART channel between the RPi and Microcontroller are queued until they can be processed. You can specify how many messages are allowed in the queue. If the queue is larger than this, the older messages get dropped.

MinAzimuthAngle
Default 0. Minimum angle that the azimuth motor is allowed to travel to. 0-360. 0 = Due North, 90 = Due East, 180 = Due South, 270 = Due West

MaxAzimuthAngle
Default 360. Maximum angle that the azimuth motor is allowed to travel to 0-360. 0-360. 0 = Due North, 90 = Due East, 180 = Due South, 270 = Due West

AzimuthDriver
Default 'drv8825'. This is the motordriver board that is driving the Azimuth motor. You can define different types of board in the StepperDriverData dictionary (above). This value tells the program which entry in the StepperDriverData dictionary to use to configure the motor.

AzimuthGearRatio
Default 240. This is the gear ratio of the drive system for the Azimuth motor. 240 full turns of the motor are required to turn the Azimuth 1 full rotation.

AzimuthMotorStepsPerRev
Default 400. This is the number of FULL steps the stepper motor must take to complete 1 full turn. Do not include 'microstepping' in this value.

AzimuthMicrostepRatio
Default 1. One way to use microstepping is to set this value to increase the steps taken by the stepper motor to complete 1 full turn. You lose power and precision when using microstepping.

1 = Full steps only. (Recommended)

2 = ½ steps taken.

This feature may change in the future and depends upon the capabilities of your motorcontroller circuit. The DRV8825 driver in the pilomar circuit supports 1,2,4,8,16 and 32.

AzimuthRestAngle
Default 0. When the motor is homed, this is the angle that it returns to.

AzimuthBacklashAngle
Default 0.0. If there is slack in the drive system, this adds an extra amount of movement to the motor when it changes direction to try to absorb gear slackness. Not recommended.

AzimuthOrientation
Default -1. If you have a different gear or drive system this allows you to flip the entire direction of the motors when moving the telescope. In the Pi-lomar design, -1 is the correct orientation. If you change this to 1 the movement will be reversed.

AzimuthLimitAngle
Feature under development. Linked to the 'OptimiseMoves' parameter.

MinAltitudeAngle
Default 0. Minimum altitude that the camera is allowed to move to. 0-90. 0 =-Horizontal, pointing at the horizon. 90 = straight up pointing to the zenith.

MaxAltitudeAngle
Default 90. Maximum altitude that the camera is allowed to move to. 0-90. 0 =-Horizontal, pointing at the horizon. 90 = straight up pointing to the zenith.

AltitudeDriver
Default ‘drv8825’. This is the motordriver board that is driving the Altitude motor. You can define different types of board in the StepperDriverData dictionary (above). This value tells the program which entry in the StepperDriverData dictionary to use to configure the motor.

AltitudeGearRatio
Default 240. This is the gear ratio of the drive system for the Azimuth motor. 240 full turns of the motor are required to turn the Azimuth 1 full rotation.

AltitudeMotorStepsPerRev
Default 400. This is the number of FULL steps the stepper motor must take to complete 1 full turn. Do not include ‘microstepping’ in this value.

AltitudeMicrostepRatio
Default 1. One way to use microstepping is to set this value to increase the steps taken by the stepper motor to complete 1 full turn. You lose power and precision when using microstepping.

1 = Full steps only. (Recommended)

2 = ½ steps taken.

This feature may change in the future and depends upon the capabilities of your motorcontroller circuit. The DRV8825 driver in the pilomar circuit supports 1,2,4,8,16 and 32.

AltitudeRestAngle
Default 0. When the motor is homed, this is the angle that it returns to.

AltitudeBacklashAngle
Default 0.0. If there is slack in the drive system, this adds an extra amount of movement to the motor when it changes direction to try to absorb gear slackness. Not recommended.

AltitudeOrientation
Default -1. If you have a different gear or drive system this allows you to flip the entire direction of the motors when moving the telescope. In the Pi-lomar design, -1 is the correct orientation. If you change this to 1 the movement will be reversed.

AltitudeLimitAngle
Feature under development. Linked to the ‘OptimiseMoves’ parameter.

MotorStatusDelay
Default 10. The number of seconds between each motor status message from the microcontroller to the RPi. Shorter delays mean more messages, longer delays mean the RPi software is slower to respond to telescope movements.

OptimiseMoves
Default False. Feature under development.

When False the Azimuth movement will not move clockwise or anticlockwise more than 360 degrees. It will reverse back to the required position instead. This protects the cables from twisting when many continuous moves are made in the same direction.

When True the Azimuth movement is allowed to continue turning clockwise or anticlockwise This does not protect the cables from twisting if the telescope rotates multiple times in the same direction.

MctlCommsTimeout
You can tell the microcontroller how long to wait for messages from the RPi. After this time period it will assume that communication is broken. This will cause any loaded trajectories to be flushed for safety to stop the motors.

SDPath
Default '/'. This is the file path used by the storage monitor to check there is enough memory on the system SD card for the system to operate.

USBPath
Default '/media/pi' This is the file path used by the storage monitor to check there is enough memory available on any attached USB drive. The storage monitor searches this path for connected USB drives.

UseUSBStorage
When True, pilomar will scan for connected USB memory sticks at startup. If found pilomar will attempt to mount the memory stick and will store captured images on the memory stick instead of the operating system SD card. This preserves the life of the SD card and also appears to generally improve performance if you are using a USB3.0 memory stick via the Raspberry Pi's USB3 port.

TuneOn32Bit
Default: False. The software will run on 32bit Raspberry Pi but will not perform well. This flag can be used to disable some features in the software to maintain some reasonable performance on 32bit O/S builds. It is highly recommended to use a 64bit O/S if you can.

LocalStarsMagnitude
Pi-lomar calculates a list of Hipparcos stars to use during the observation, this is a subset of the complete Hipparcos catalog for performance reasons. LocalStarsMagnitude says the dimmest magnitude of stars that are included in the subset. See also TargetMinMagnitude parameter which is related to this in the tracking mechanism.

ConstellationStarsMagnitude
A selection parameter for selecting stars from the Hipparcos catalog. The preview calculations can mark major constellation points. This is the magnitude of the dimmest stars to select from the Hipparcos catalog.

CameraSaveJpg
When True pilomar saves images as JPG files. Note JPG files have some compression in them, so the image quality will not be as good as the raw files. This is set automatically from the camera menu if you choose which image types you want to capture.

CameraSaveDng
When True pilomar saves images as DNG files. These are 'raw' dumps of the camera sensor data. Astro stacking software generally produces higher quality images with these raw files. There is no compression with these files. This is set automatically from the camera menu if you choose which image types you want to capture.

UseTracking
When True the drift calculation is active. The telescope will periodically compare the live image against a simulated image to check that the telescope is on target. If any differences are found Pilomar can issue correction commands to the motors.

TrackingTargetGrayscale
Default False. The drift tracking function calculates an expected view of the sky.

When False: The image is created in colour and diffuse objects (galaxies and nebulae) are included.

When True: The image is created in grayscale, no diffuse objects (galaxies or nebulae) are included.

LatestTrackingFilter
Default None. In urban settings there can be light pollution and other issues in the images that make the stars less clear. When capturing images for the drift tracking calculation you can define simple OpenCV based filters to apply to the image. This identifies which (if any) filter script should be applied to the images.

You can define your own filters via the parameter file. They perform things like haze reduction and enhancing stars in the images. These are discussed later in the document.

TrackingMatchThreshold
Default 5: The drift tracker will only recommend tuning the motor positions if sufficient stars have been matched between TARGET and LATEST images. This is the threshold value.

TrackingInterval
How many seconds between each tracking check? This tells pilomar how often to perform the drift calculation. This checks that the telescope is on target and sends adjustments to the motors if required.

TrackingExposureSeconds
When taking a tracking photograph for the drift calculation you need to take a consistent exposure time so that we get a predictable number of stars captured. This is the exposure time.

TrackingMapSpan
Default 1.0. You can widen the search area that the drift tracking will check with this parameter.

The default 1.0 means that an area equivalent to the camera image field of view is checked. This will detect small differences in position and is the most efficient.

A value of 2.0 means that an area TWICE the width and TWICE the height of the camera image field of view is checked. This will detect larger position errors, but is slower.

TrackingZoneMatches
Default 3. If you are using a high TrackingMapSpan value the drift tracking will search multiple regions of the target area looking for a match. This tells the drift tracking when to stop searching. Default 3 means that after 3 zones have been matched the search can stop. This is to prevent too

many unnecessary searches being performed once the drift has been established with high enough confidence. If only 1 or 2 zones match, the drift tracking will still try to autocorrect the telescope's position.

TrackingZoneShift
Default 0.33. This is the proportion by which the different tracking areas are shifted relative to each other. 0.33 means that the next row/column of checks will be offset from the previous areas by 33% of the camera's field of view.

Higher values will mean fewer areas are checked (=faster), and there is an increased risk of failing to find a match.

Lower values will mean more areas are checked (=slower), but there is more chance of identifying a match for the drift tracking calculation.

TrackingDebug
Development feature. Additional debug messages recorded during the tracking calculations.

ShowPGCEntries
Default False. The latest copy of the 'ngc.json' catalog of galaxies includes 'ngc','ic' and 'pgc' catalogs. The pgc catalog is very large and very slow to process. It will generate more realistic star fields, but will take several minutes to process the list. The PGC catalog is still being refined for this project as it currently lacks apparent magnitude information. Setting this to True will include pgc entries in TargetTrackingImage and PreviewImage files.

GeneratePreview
Default True. When True the software periodically takes a live image from the observation and paints an overlay on top of it creating a Preview image. The preview image has various scales, markings and labels of how objects should be arranged in the image. The object labels to no precisely match with the actual items in the image, but it gives an indication of what you should be seeing in the images.

InitialGoTo
Default True. When True, pilomar begins each observation session by pointing the camera at the target BEFORE downloading the complete trajectory.

When False, pilomar waits for the trajectory to be downloaded before moving the telescope.

The overall time to begin an observation is the same, but using InitialGoTo makes sure that the motors are configured and working faster.

TargetInclusionRadius
Default 15. A selection parameter for selecting stars from the Hipparcos catalog. The drift and preview calculations generate a target image of the sky that it expects to see. This radius specifies how many degrees around the observation target it should include stars.

TargetMinMagnitude
Default 7.0. Target and preview images select stars of this magnitude or brighter. Use this to increase/decrease the number of stars in the target images to match those captured in the live target image. See also LocalStarsMagnitude parameter which sets an upper limit on this value.

UseLiveLocation

Default True. When calculating object locations for the PREVIEW and TRACKING images pi-lomar can choose the ‘time’ of the calculations using LIVE or LAST REPORTED camera positions. This switches between the two calculations. This option is likely to disappear in the future.

DebugMode

Default True. This turns on/off debug mode during observations. You can turn this on/off using the ‘d’ key during an observation too.

When True – the observation dashboard does not appear, pilomar just lists scrolling text as the observation proceeds. This makes it easier to see when errors occur.

When False – the observation dashboard is displayed instead. If error messages are generated the dashboard display may rapidly erase them making debugging more difficult.

KeyboardScanDelay

Default 2. The observation dashboard scans the keyboard regularly for keypresses by the user. This parameter tells how often the keyboard is scanned. Eg 3 means that the keyboard is scanned after every 3 display refreshes.

The more often you scan the keyboard the more likely the screen is the blink or flash during the refreshes. If you are sensitive to flashing images it is better to slow down the scanning by increasing this value.

LocalTZ

Default ‘UTC’. Feature under development. Define the local timezone, Pi-lomar expects to run with UTC clocks, you can define a local timezone here. Pi-lomar works internally with UTC timestamps, however you can switch some displays to show dates and times in your local timezone.

The value must be one of the recognised timezone values from the pytz package. There is an option on the Miscellaneous Tools menu that lets you set the timezone to an acceptable value.

You can also list the values directly in Python

```
>>> import pytz
>>> print(pytz.all_timezones)
```

DisplayTZ

Default ‘UTC’. Feature under development. This says which timezone dates and times should be shown in. Pi-lomar works internally in UTC timestamps, however you can switch some displays to show dates and times in your local timezone. This says which timezone is being used for displays.

It should be either “UTC” or the same as the LocalTZ parameter.

If will be error-checked and auto-corrected to one of those values when the program starts. Generally, you don’t need to set this. The value can be changed from the menu and during observations.

You can change the value from the *Miscellaneous tools* menu. You can also toggle between UTC and LocalTZ by pressing the ‘U’ key during an observation.

HomeLat

Default None.

Eg: "52.4224 N"
This is the latitude of the telescope. Its home location. You have to provide a value for this when you first run Pilomar. You can edit it directly in the parameter file, or use the *Change Home Location* option on the Miscellaneous Tools menu..

HomeLon

Default None

Eg: "0.4532 W"
This is the longitude of the telescope. Its home location. You have to provide a value for this when you first run Pilomar. You can edit it directly in the parameter file, or use the *Change Home Locatio*n option on the Miscellaneous Tools menu.

LocationList

Dictionary of predefined Home Locations for the telescope. You can add to this list if you have additional locations that you frequently use. You can select from this list through the software without having to edit the parameter file directly.

MarkupInterval

Default 300. How many seconds between generating each 'preview' image. See the preview image description for more detail. This is a diagnostic image generated periodically to explain what is being photographed.

MarkupShowNames

Default True. When True, the preview images will include the names of objects if known. (Replaces earlier MarkupShowLabels parameter)

MarkupShowCoordinates

Default True. When True, the preview images will include the Ra/Dec and Alt/Az position of objects. (Replaces earlier MarkupShowLabels parameter)

MarkupStarLabelLimit

Default 100. This the maximum number of stars that will be marked and labelled in a preview image. If the number is too high the image becomes too cluttered.

MarkupAvoidCollisions

Default False. When True some of the labels in the preview images are suppressed if they overwrite other labels. This is to keep busy images uncluttered.

MarkAllStars

Development feature to mark all stars in preview images even if they are below selection thresholds.

LensList

A dictionary of predefined lenses. You can select from the list when configuring the telescope. You can add to this list if you have additional lenses.

LensLength

Defualt 16. Focal length of the lens. The telescope is primarily designed for the 16mm lens.

LensHorizontalFov

Default 21.8. Field of view (horizontal) of the lens. Default is for 16mm lens.

LensVerticalFov
Default 16.4. Field of view (vertical) of the lens. Default is for 16mm lens.

SensorType
Default 'imx477'. Sensor code ie IMX447

IRFilter
Default True. Documents that Infra Red filter is in place or not.
True: Infra red filter is fitted.
False: Infra red filter is removed.
For future developments, currently documentation only.

PollutionFilter
Default False. Documents that a Pollution Filter is in place or not.
True: Pollution filter is fitted to the lens.
False: Pollution filter is not fitted to the lens.

SolarFilter
Default False. Documents that a solar filter is in place or not.
True: Solar filter installed.
False: Solar filter not installed.
For future developments, currently documentation only.

TrajectoryWindow
Default 1200. Pi-lomar sends a trajectory of the target to the motorcontroller board. This value tells how many seconds into the future the trajectory must cover. The observation will not begin until the motorcontroller has received enough trajectory information to cover this period.

UseDynamicTrajectoryPeriods
Default True. Set to False to force Pi-lomar to cut the trajectory into equal time periods.

Set to True to allow Pi-lomar to optimise the trajectory chunks. This usually reduces the amount of trajectory data that needs to be synchronised.

AltitudeSlewMicrostepRatio
Default 1: GOTO and HOME moves are large, you can specify the stepper motor's microstepping ratio for large moves here. It is faster to perform large moves with larger motor steps. Values 1,2,4,8,16,32 for DRV8825.

AuroraCameraAltitude
Default 5degrees. This is the altitude that the camera points to when capturing Aurora.

AzimuthSlewMicrostepRatio
Default 1: GOTO and HOME moves are large, you can specify the stepper motor's microstepping ratio for large moves here. It is faster to perform large moves with larger motor steps. Values 1,2,4,8,16,32 for DRV8825.

CameraCommands
A dictionary of commands for capturing different types of images. There are commands for each image type and each camera driver that pilomar can use.

CameraDriver
Which camera driver is pilomar using to capture images? raspistill, libcamera-still, pilomarfits.py? This is set automatically from the camera menu if you choose which image types you want to capture.

CameraSaveFits
Default False: When TRUE and CameraDriver = pilomarfits.py the program generates a basic FITS format file from the camera's bayer raw data. This is set automatically from the camera menu if you choose which image types you want to capture.

FakeAurora
Default True: If the camera is disabled, pilomar will generate simulated Aurora images. Set this to True if you want it to generate a simple aurora image. This is a development feature.

FakeField
Default False: If the camera is disabled, pilomar will generate simulated images. Set this to True if you want it to generate artificial electrical field noise in the image. This is a development feature.

FakeMeteorPercent
If FakeMeteor parameter is True, this is the percentage of images which will have artificial meteor trails added.

FakeNoise
Default False: If the camera is disabled, pilomar will generate simulated images. Set this to True if you want it to generate artificial image noise. This is a development feature.

FakePollution
Default False: If the camera is disabled, pilomar will generate simulated images. Set this to True if you want it to generate artificial light pollution in the image. This is a development feature.

FakeStars
Default True: If the camera is disabled, pilomar will generate simulated images. Set this to True if you want it to generate simulated stars in the image.

FastFlush
Default False: When TRUE all entries written to the log files are immediately flushed to 'disc'. This captures messages immediately in the log file, but hits the disc/SDCard heavily. Only use this during development/debugging activities.

FastImageCapture
Default False: Normally pi-lomar performs all its image processing as soon as an image is captured from the camera, before proceeding to capture the next image. In some situations if you want to capture images as quickly as possible set this to TRUE. When True the image is captured with as little processing as possible. You must then process the images separately at the end of the observation run.

FilterScripts
A dictionary of image filters available to pi-lomar to clean/enhance images. This is mainly used by the tracking mechanism to enhance stars for alignment calculations.

GenerateKeogram
Default False: When True a Keogram is generated during the observation. This is a new feature in development, mainly for aurora monitoring. Keograms are automatically created if an aurora target is chosen.

HorizonAltitude
Default 0degrees. This is the minimum altitude that targets will be observed. When a target falls below this altitude the observation stops. This is separate to the MinAltitudeAngle parameter. The telescope may be able to physically move lower for other purposes, but observations will not continue below this altitude.

MinimumDriftCorrection
Default 50: This is the smallest tracking error (in pixels) that pi-lomar will react to. Any tracking error less than this is ignored.

SessionHistoryLimit
Default 30: Pi-lomar remembers recent observation targets so that you can repeat them easily. This is the number of targets that are retained.

SlewEnabled
Default False: When TRUE pi-lomar can use LARGER (=faster) steps in the stepper motors when performing large moves such as HOME and GOTO moves. The telescope will still use the finer microstepping rate when performing actual observations.

SpeedList
Dictionary of stepper motor speed combinations. You can select from this list for common settings of FastTime, SlowTime and TimeDelta parameters.

FastTime
Default 0.001 seconds: Fasted time of the ‘ON’ pulse sent to the DRV8825 driver to move the motor one step. This is the pulse length when a move has accelerated to full speed.

On RP2040 microcontrollers the minimum practical value seems to be about 0.001 seconds. This is relatively noisy.

On RP2350 microcontrollers the minimum value can be lowered to 0.0005 seconds. This is faster, quieter and smoother operation.

SlowTime
Default 0.05 seconds: Slowest time of the ‘ON’ pulse sent to the DRV8825 driver to move the motor one step. This is the initial pulse length when starting a move.

TimeDelta
Default 0.003 seconds: The acceleration rate of the ‘ON’ pulse sent to the DRV8825 driver to move the motor. The pulse length starts at the SlowTime value, after each step the TimeDelta is subtracted. The following pulse will be faster. This repeats until the FastTime pulse length is reached.

ObservationStopPin
If a STOP BUTTON is wired up to the RPI GPIO header. This is the GPIO pin to use. This can be used to send a simple ‘STOP OBSERVATION’ signal to the software. (This is not an emergency stop, it is only read during the observation process. To urgently stop movement in all circumstances it is best to kill the power to the motors.)

LatestTrackingFilter

Name of the software filter selected for enhancing tracking images.

MenuTitleFG

Default textcolor.BLACK. Color code (0-255) for menu titles displayed by pilomar. (Foreground)

MenuTitleBG

Default textcolor.GRAY. Color code (0-255) for menu titles displayed by pilomar. (Background)

MenuSubtitleFG

Default textcolor.BLACK. Color code (0-255) for menu sub titles displayed by pilomar. (Foreground)

MenuSubtitleBG

Default textcolor.GRAY30. Color code (0-255) for menu sub titles displayed by pilomar. (Background)

TitleFG

Default textcolor.WHITE. Color code (0-255) for Titles displayed by pilomar. (Foreground)

TitleBG

Default textcolor.GRAY. Color code (0-255) for Titles displayed by pilomar. (Background)

TextFG

Default textcolor.WHITE. Color code (0-255) for text displayed by pilomar. (Foreground)

TextBG

Default textcolor.BLACK. Color code (0-255) for text displayed by pilomar. (Background)

TextGood

Default textcolor.LIGHTGREEN. Color code (0-255) for values representing 'good' values. (Foreground)

TextPoor

Default textcolor.YELLOW. Color code (0-255) for values representing 'poor' values. (Foreground)

TextBad

Default textcolor.ORANGERED1. Color code (0-255) for values representing 'bad' values. (Foreground)

BorderFG

Default textcolor.DARKGREEN. Color code (0-255) for window borders. (Foreground)

BorderBG

Default textcolor.BLACK. Color code (0-255) for window borders. (Background)

ScanForMeteors

After taking photographs Pi-lomar can scan them immediately for meteor trails in the images. These images will be listed separately at the end of the observation so that you can analyse them more closely.

If a trail is spotted in an image, it is listed in the camera events window of the dashboard.

```
Camera events
09:44:23 light_20230614094347_00.jpg
09:45:02 light_20230614094425_00.jpg
09:45:04 Trail in image (Meteor/plane/satellite).
09:45:42 light_20230614094505_00.jpg
09:46:20 light_20230614094544_00.jpg
09:46:51 preview_20230614094623.jpg
09:47:30 light_20230614094652_00.jpg
```

MinSatelliteAltitude

The minimum angle that a satellite must rise above in order to be suggested as an observable satellite pass. Very low angles will be lost in the haze and other objects around the horizon.

SuggestionMagnitude

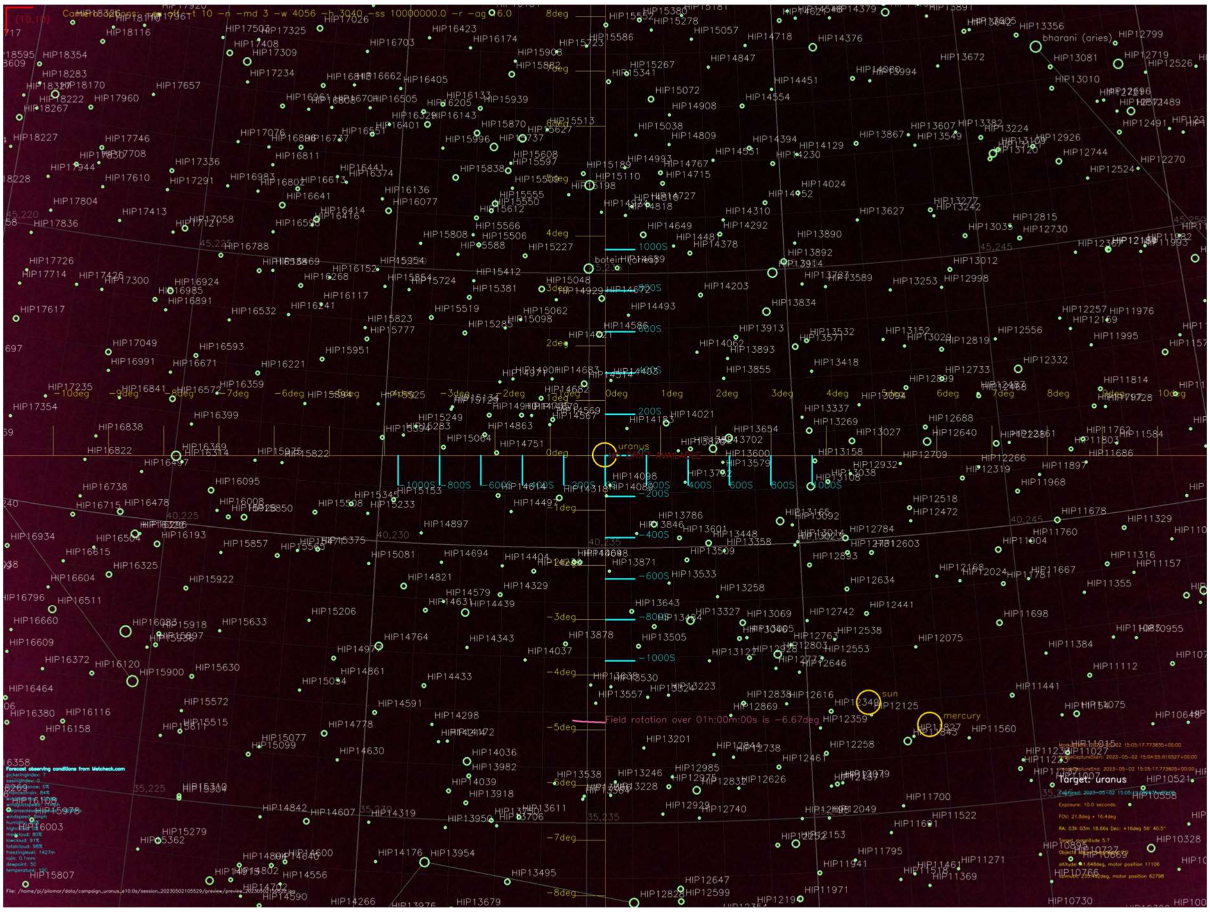
Default 11: When suggesting possible observation targets this is the minimum magnitude of targets that will be listed.

SuggestionPixels

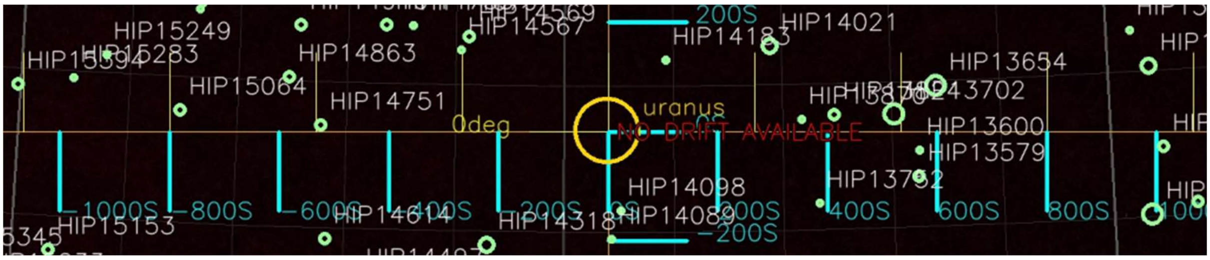
Default 100: When suggesting possible observation targets this is the minimum image size of targets that will be listed.

Preview image

The telescope generates a preview image periodically, this is a copy of the latest 'light' image captured but with additional information superimposed.



Central scale



The centre of the image will always show the target being observed. In this case Uranus.

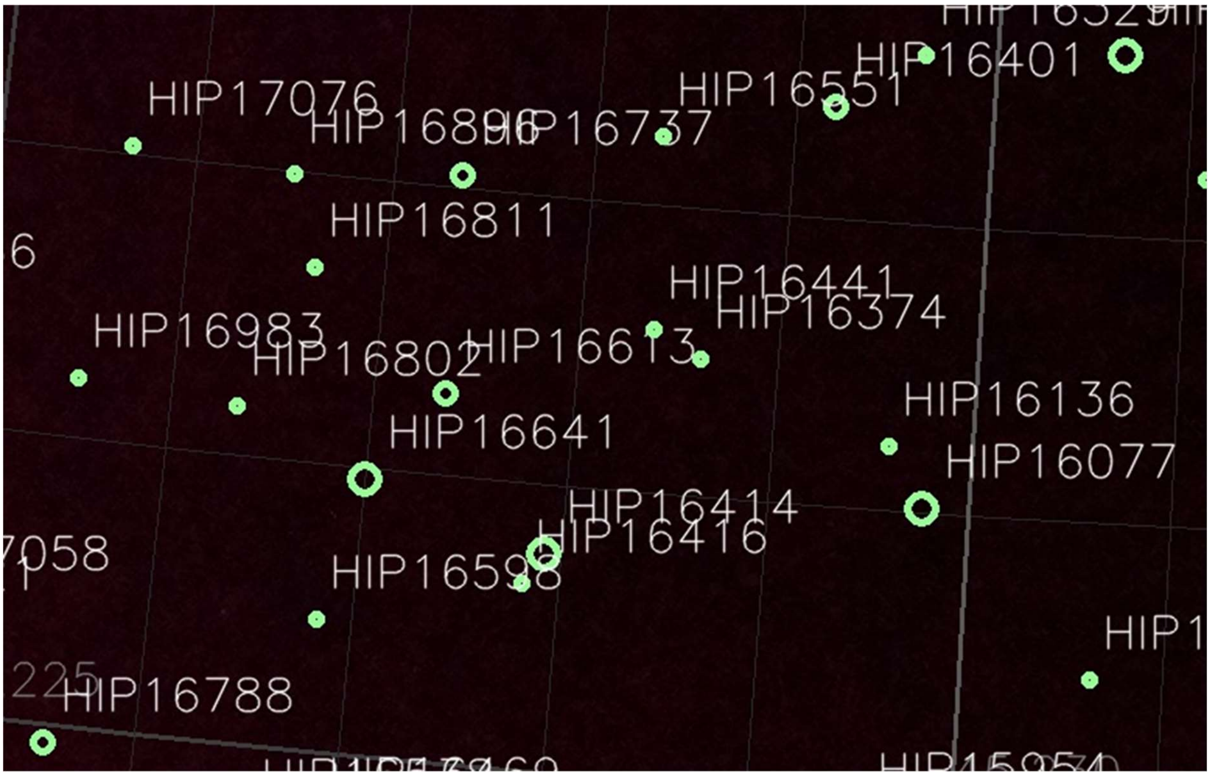
The YELLOW horizontal and vertical scale is a measure of the field of view in degrees from the centre of the image. Useful for measuring the angles and distances between objects.

The BLUE horizontal and vertical scale is a measure of the field of view in steps from the centre of the image. The steps are the number of steps the steppermotor and gearbox have to make to cover that distance. This is useful for finetuning the position of the telescope.

Red central drift figure

If a drift calculation has been performed recently, this shows how far off target the tracking algorithm thinks the telescope is. It lists the horizontal and vertical step count that was measured. It’s a good indication that the tracking is working.

Hipparcos references

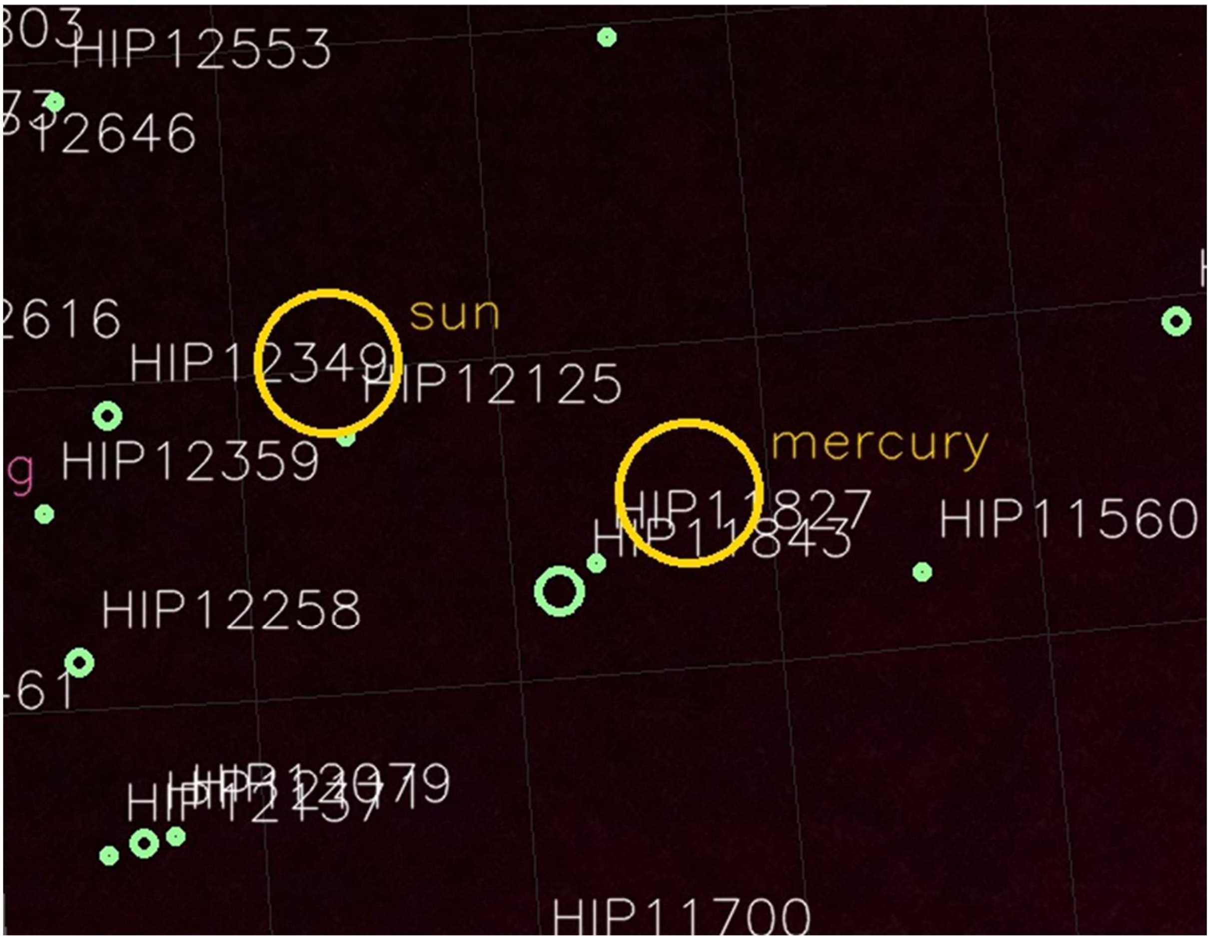


The green circles are the expected positions of stars. The HIPnnnn label is the Hipparcos catalog reference for the star. Due to lens distortion and alignment differences these will not always match perfectly with the actual stars photographed, but you should be able to recognise patterns and identify major stars in the actual photographs using this.

If there are too many or too few stars being labelled in the image you can change the MarkupStarLabelLimit parameter to change the number of stars selected. You can also try activating the MarkupAvoidCollisions parameter which will further reduce the label count.

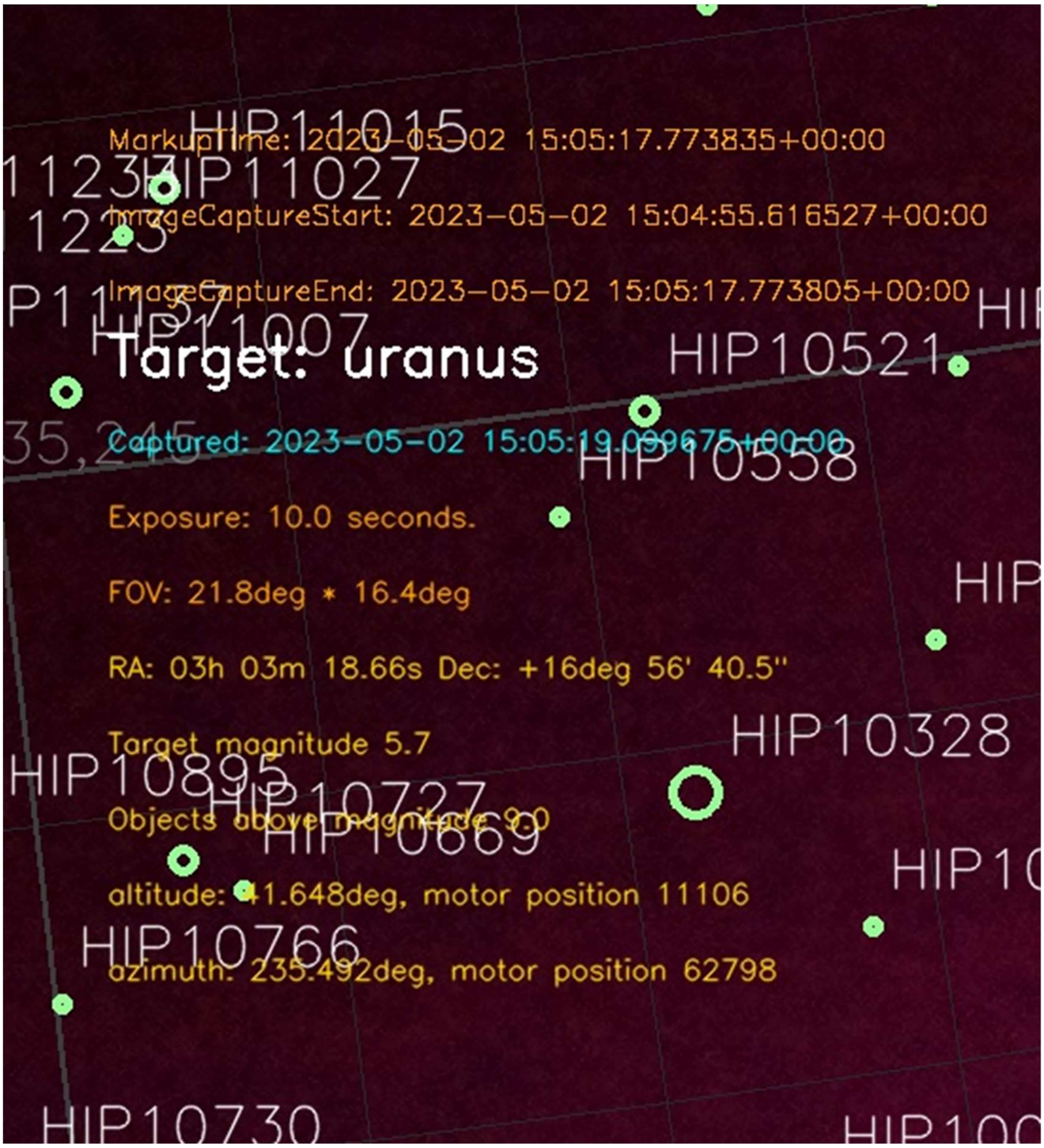
NOTE: Messier and New General Catalog (NGC) objects are also labelled in the images.

Solar system objects



All solar system objects are shown as yellow circles and labelled accordingly.

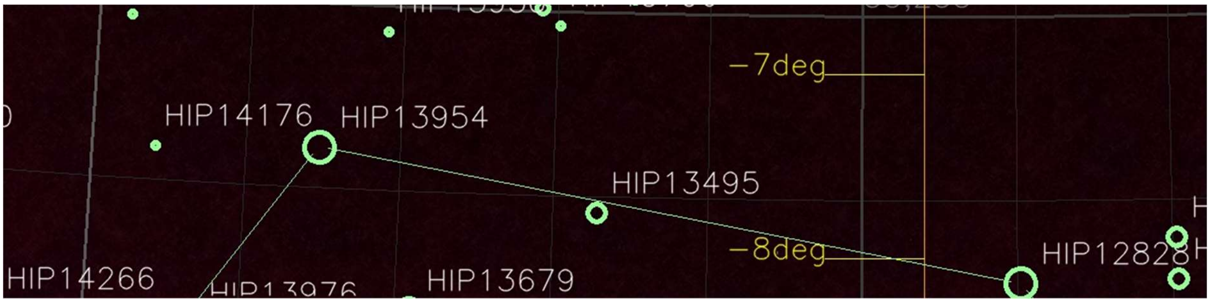
Target information



The bottom right of the image contains key target information for the image, including the object, location, times, motor positions, exposure time and field of view.

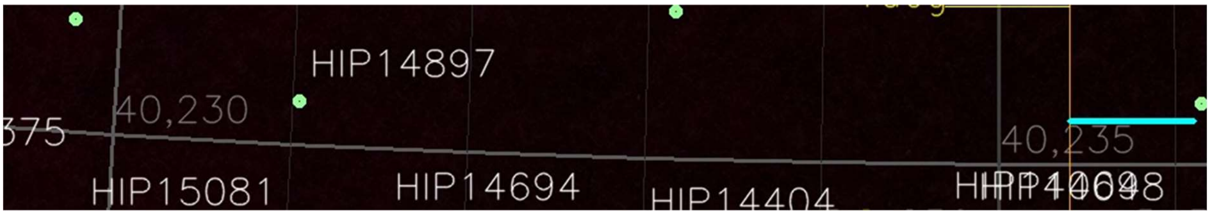
The preview and tracking mechanisms use the same rules for choosing entries in the Hipparcos (star) catalog. The entry ‘Objects above magnitude xxxx’ tells how they been selected. If the preview (and therefore tracking) is showing too many or too few stars then you can adjust this in the parameters file.

Constellation lines



Major constellation lines are marked between stars. This is to help identifying objects in the image.

Altitude/Azimuth lines

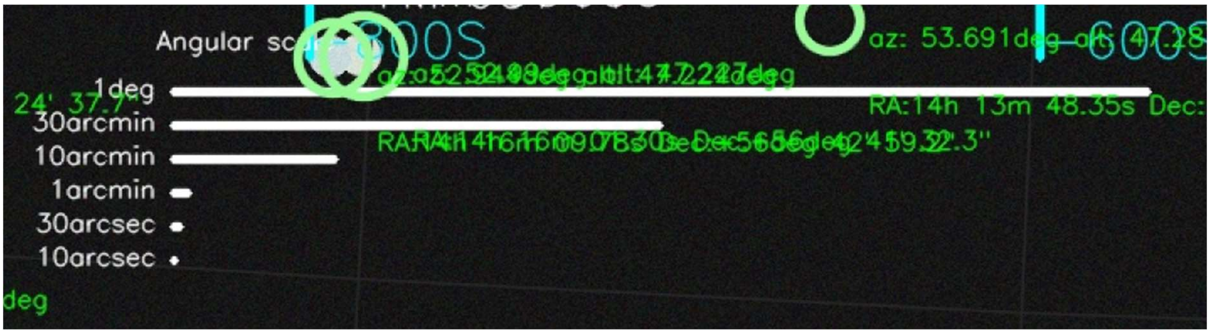


The altitude and azimuth lines help to identify where the telescope is pointing when the image is taken. The azimuth lines will converge towards the nadir and diverge towards the horizon.

Camera settings

The yellow text in the top left of the image shows the camera settings used to capture the light image.

Angular scale



A simple angular scale is also printed on the image. Showing some useful relative angular sizes if you are measuring positions on the images.

Limitations with the preview image

The preview image is based upon an actual ‘light’ image captured by the telescope, however there are some limitations to consider. It was developed to help with tuning and aligning the telescope but is sometimes very interesting itself. For example if you want to understand objects in one of the ‘light’ images that the telescope collects it is useful to look at a suitable preview image to see what should be visible.

- The image is a single 'jpg' from the camera. It is not processed, cleaned, enhanced, stacked etc. So an individual frame may look quite poor.
- All the markings are added based upon expected positions of objects. Pi-lomar does not actually identify individual items directly from the image. You should expect some differences between the locations in the original image and the markings added by the software. If the telescope is misaligned, the differences can be even larger.
- The real image will have some distortion due to the characteristics of the lens, Pi-lomar does not take that into account, so the labels may disagree more as you approach the edges of the image.

Taking all the above items into account the preview image is still useful to help you identify individual items in the image. Even if slightly misaligned you will still recognise many items and be able to match the labels to the objects in the image.

Preview animation

When the observation completes, the preview images are combined into a simple .avi animation. You can play this back to see how the telescope has followed the target throughout the observation as a simple movie. For example, this demonstrates how the field of view rotates during an observation, and for fast moving targets shows how they travel against the star background.

Utilities

pilomarfits

The traditional camera utilities which come with the operating system will export .DNG format files, but not .FITS files.

src/pilomarfits.py is a utility to generate .FITS format image files using the raw data directly from the camera sensor. This uses the astropy and picamera2 modules to generate the files in the correct format.

What can pilomarfits do?

Pilomarfits is intended to generate just the .FITS format image files. These contain the raw bayer matrix data from the camera. The data needs to be 'debayered' by your astro photography software to return the image to full colour.

For utility purposes pilomarfits can also generate a debayered and normalised copy of the image as a .JPG file too. This will be easily viewed in most image viewers, but it not intended to be a high quality image. The image is debayered using OpenCV utilities then the resulting data normalised to fit within the 8bit colour depth of the .JPG format. Normalizing the image increases the chance of a visible image regardless of exposure settings.

How to use pilomarfits

Pilomarfits presents a similar command line operation to libcamera-still. Where possible it copies the same command line options syntax. It does not support ALL the features of libcamera-still, it ONLY supports the options needed to generate the .FITS and .JPG files.

You can use pilomarfits from the command line similarly to the way you would call libcamera-still or libcamera-jpg. To generate .jpg and .dng files with libcamera still…

```
% libcamera-still –output myimage.jpg –shutter 5000000 --raw
```

It is a python script so you run it via python. To generate .jpg and .fits files with pilomarfits…

```
% python3 pilomarfits.py -output myimage.jpg -shutter 5000000 --raw
```

Common issues

The jpg images are generated from the bayer matrix data, the resulting image may not be as high quality as jpg images generated from other utilities, but it should be useable. IF you want high quality .jpg images you should not try to generate .fits files at the same time. You can then use libcamera-still which has more image processing logic in it.

Pilomarfits will only work if your Raspberry Pi computer has picamera2 / libcamera support. Older operating systems which use the old raspistill command cannot run pilomarfits.

If .jpg images are completely white or completely black it is usually an exposure issue. Either the exposure time is too low or too high. You can also get the same problem if the analog gain is set too high. A very high value of analog gain will overexpose the entire .jpg image, when it is normalised it will produce a BLACK image file.

Regular maintenance

Currently the following things should be checked regularly

Cable condition

Check that cables are not excessively twisted or trapped in the mechanism. This is especially important if you are using the telescope with the ‘OptimiseMoves’ parameter enabled.

Gear engagement

You may need to periodically check that the gearbox assemblies are still engaged firmly with the gear rings. These may loosen after prolonged use, or the plastic body may even wear or soften over time and shift.

Camera focus

Over time the focus can need finetuning, if you notice stars are losing their crispness in images then it is a good idea to repeat the focussing operations from the original build instructions.

Operating system updates

It’s a good idea to periodically update the operating system with the latest patches. However beware that some general updates can change the behaviour of the Raspberry Pi computer. You may find that some features of the software stop working due to these changes.

It is critical that you back up your SD card image before applying updates.

If you find there are problems with the telescope after performing an O/S update there are some things you can try.

1) Check the current ‘parameter’ list to see if there are any parameters which allow pi-lomar to adapt to the new O/S.
2) Check for an update of the pi-lomar repository on Git-Hub. It may have already been updated to solve any new problems with recent O/S updates.
3) In severe cases you may want to build a fresh installation of the pi-lomar project on a new SD card. Keep your previous image as a backup or separate SD card in case of persistent problems.
4) If you cannot solve the problem you can create an issue in the Git-Hub project to see if anyone else has any ideas.

5) If you find a new problem and solve it, please also share that on the Git-Hub project, you may be the first person to discover the problem, but you may not be the last!

General assembly

If you have used 'nyloc' nuts in the assembly you should not have to tighten any of the bolts in the structure very often at all. If you have used regular plain nuts in the assembly then they may loosen over time due to vibration and the flexibility of the plastic.

Updating Pi-lomar

Updating the Pi-lomar package

There may be updates to the software available from the GitHub site. I try to make these updates compatible with earlier versions, but I cannot guarantee this will always be the case.

Generally, if you want to update your Pi-lomar installation you should consider the following issues.

- ALWAYS make a backup of the SD card before performing any update. It is the easiest way to reverse a change if something goes wrong.
- ALWAYS take a copy of the pilomar_params.json file from the /data directory on the RPi. These are your personal settings, including your home location and your preferred motor operation. Each new version of the package will try to adapt existing parameter files to the new software, so you do not need to start configuring your system from scratch each time.
- Pi-lomar's GitHub project page will offer advice on how to perform the update. Generally you have two options.
  - (1) Perform a completely fresh build on a clean SD card. You can use your backup copy of pilomar_params.json on the new installation.
  - (2) Simply download the latest GitHub .zip file from the website and overwrite the existing /home/pi/pilomar contents. You can then re-apply your backup of pilomar_params.json before starting the new program version.
- New releases generally change the code on the RPi AND the Tiny microcontroller. Be sure to load the latest code.py sourcecode onto the microcontroller before restarting the project.
- It is generally safe to re-run the build script for your operating system. (such as /scripts/buildpilomar_bookworm64). It should normally only apply NEW requirements to your build.
- If you get into problems, try with a clean installation on a fresh SD card, this may include the latest version of the O/S for the RPi that you are using.

Troubleshooting

Operating system

Which operating system should I use?

Pi-lomar currently runs on Raspberry Pi 3B, 4B and 5 models. It should also run on the Compute Module 4.

On RPi3B you need to run the legacy 32bit BUSTER operating system. You need the specific version listed on the project web page. The performance on a RPi3B and 32bit O/S is significantly lower than the new models and O/S.

On RPi4B, Compute Module 4 and RPi5 you can run the latest 64bit BOOKWORM operating system. You can currently use the latest available version. It is strongly recommended to use this 64bit latest version. The performance is much better.

It may not always be possible to support the RPi3B as the software is developed. Currently it is recommended to use a RPi4B as the optimum balance of cost vs power.

Why is it such an old copy of the O/S for the Raspberry Pi 3B?

Pi-lomar depends upon several packages in order to work fully. Each time a new O/S is released there is sometimes a delay before all the packages are fully available for the new O/S, so you may not be able to install them. Also, the new versions of the packages may have been updated. They in turn may depend upon other packages which are not yet ready.

So when a new O/S comes out, you usually find that some elements that Pi-lomar depends upon are missing or broken.

For this reason, I've stuck to a legacy copy of the O/S where I know that all the packages are available and work.

Where do I find the old legacy copies of the O/S?

The Raspberry Pi O/S installer only goes back 1 version of the O/S, to get older versions you need to pull the archived images off the Raspberry Pi website elsewhere. At the time of writing, the required O/S images are archived here…

https://downloads.raspberrypi.org/raspios_armhf/images/

You need to use a disc imager to copy the O/S image onto your SD card.

Can I use the 64bit O/S image?

Yes, but only on the RPi4B and RPi5 and Compute Module 4. It is strongly recommended to use the latest 64bit image.

Software

How do I get support for the programs?

Pi-lomar is just a freely published project, not a commercial product. There is no guaranteed functionality, quality or support, but you have full access to the source codes involved. Pi-lomar is maintained and distributed through https://github.com/Short-bus/pilomar. You can raise an issue there if you are struggling with something, but there's no guarantee someone can provide a solution, although you can at least share the problem! You are free to fork copies of the software, but please retain and respect all the licences. You can also submit pull requests if you have contributions to add to the project.

Communication between RPi and microcontroller is failing.

Possible reasons are :-

UART wiring is wrong.

TX on RPi should connect to RX on microcontroller. RX on RPi should connect to TX on microcontroller. If they are incorrectly linked then the RPi will receive garbage when it listens to the microcontroller. The RPi will complain that it cannot decode the data received for example.

```
2023-12-11 02:34:02.041178+00:00 0.000299 uart.Read: uart.read(1) failed. Ignored. 'utf-8' codec can't decode byte 0xa
4 in position 0: invalid start byte
```

Microcontroller is not working.

Check the lights on the microcontroller. It should flash 'blue' periodically to show that it is trying to communicate. No lights means microcontroller isn't working or that the MctlLedStatus parameter is False. Any other color may indicate an error state.

If there is no indication on the LED at all then check that CircuitPython is successfully installed.

Microcontroller has gone into read-only mode.

Sometimes microcontrollers go into a safe mode where they set their memory as read-only. When this happens you will not be able to update the software such as code.py anymore.

Check on the website for your microcontroller to see what action should be taken in this case. For the Tiny2040 and Tiny2350 there are a couple of solutions recommended on the CircuitPython website. Both solutions require that you erase everything on the microcontroller and reinstall from scratch.

If you have to use one of the flash_nuke.uf2 files to completely reset the microcontroller it usually works first time, however I have found occasions where you have to do it multiple times before it takes effect.

Problems installing CircuitPython or code.py on the microcontroller.

Sometimes you have to be persistent with this step. You can get multiple errors at various stages, which may disappear if you repeat the task, or you may need to reset the microcontroller, or even reset the RPi.

If you have repeated failures to install the uf2 file or code.py program things can get in a mess. The microcontroller can switch to 'read only' mode. You will not be able to download the code.py program anymore on to it. Check on the CircuitPython website for instructions on how to reset the file system on the microcontroller (See above).

In recent versions of CircuitPython I have noticed that the code.py program needs to have the correct permissions otherwise it will not execute properly on the microcontroller for some reason. Make sure that the code.py program has EXECUTE permission for all users. It may have NO execute permission by default. This seems to have cured a few installation problems with CircuitPython 9.2

If the Thonny editor issues errors, make sure that the editor is in the correct mode, it should say it is editing CircuitPython on a Tinyxxxx microcontroller at the bottom of the editor window.

Hardware

Which RaspberryPi computers can I use?

The telescope was originally designed to run on a Raspberry Pi 4B with 2GB of memory.

Tested and succeeded:

RPi5 8GB 64bit O/S (recommended)

RPi 4B 2GB 32bit O/S (slower)

RPi 4B 2GB 64bit O/S (recommended)

RPi 3B 1GB32bit O/S (Very slow)

Tested and failed:

RPi 4B 2GB 64bit O/S (Cannot install all packages)

RPi Pico W 1.1 (Cannot install all packages)

Other RPi computers

RPi 4B with 4 or 8GB memory should be no problem.

RPi 4B with 1 or 2GB memory should work, but it may run more slowly due to memory requirements.

Compute Module 4 (2GB memory) has been used by at least one build.

RPi5 with 2GB and 4GB should work, but has not been tested.

The RPis running 64bit O/S often seem to run at higher clock speeds. The CPU heats up, I strongly recommend having a cooling solution in place!

If running on an older RPi such as a 3B you may want to reduce functionality so that it requires fewer resources from the CPU. Tracking and Preview handling are particularly heavy and can be disabled or made less frequent through the parameter file.

Which microcontrollers can I use?

Any microcontroller which supports :-

- UART communication with RPi.
- CircuitPython/MicroPython.

Note that the pins used for sending/receiving signals across the circuit will probably have to be changed, but they are all configurable within the source code.

The code.py program is written in CircuitPython, it could be adapted to MicroPython with a few changes to accommodate minor method and 'include' differences.

Pi-lomar was designed for an 8MB RP2040 based microcontroller. The Pimoroni Tiny2040 was the most reliable board at the time of design. The Pimoroni 4MB Tiny2350 will also work with Pi-lomar because it is a drop in replacement for the original Tiny2040 board.

The circuitpython/tiny2040/code.py program provided in the GitHub package is designed for the Tiny2040. Its functionality is limited to the capacity and capabilities of the Tiny2040. If you want to add more capabilities to the microcontroller software I suggest you use the Tiny2350 instead because it has a faster CPU and more RAM available at runtime. The circuitpython/tiny2350/code.py program has a few enhancements specifically for the Tiny2350.

If you are building a new project I recommend using the Tiny2350 because it can use more up-to-date CircuitPython builds. The Tiny2040 software will only run on CircuitPython 7.2 and 8.2 which you need to retrieve from the CircuitPython website.

How do I debug the microcontroller?

WARNING! BEFORE PROCEEDING! When the motorcontrol circuitboard is connected to the RPi it provides power to the microcontroller through the GPIO cable. If you connect the two devices with a USB cable at the same time you will be providing two conflicting power sources to the microcontroller. These two sources can be slightly different voltages. You may damage the microcontroller or RPi! You should use only 1 method to connect the two devices at a time.

If the code.py program triggers a runtime error due to a programming error you may see the microcontroller LED start to flash RED every couple of seconds. This is a clue that you may need to debug the code further.

The microcontroller provides some debug information via the SHELL window in Thonny, but this is not normally exposed when the telescope is running. To see this information you must view this serial stream by using a USB cable to connect the microcontroller to the RPi and running Thonny on the RPi to view it. This is the same way that you would update the software on the microcontroller. (See the power warning above!) When you connect or disconnect a USB cable to the PCB make sure that the pi-lomar software is stopped before making the change. Then restart the software so that it can recognise the cable change and behave accordingly.

If the microcontroller is failing or rebooting, the messages on the USB serial feed may be the only way for you to understand what is happening because the microcontroller software may not be able to transmit any log messages before terminating/rebooting.

If you want to run the microcontroller without the GPIO header attached you will need to add jumper cables to reconnect the GND, TX and RX pins between the RPi and the motorcontrol circuitboard so that UART communication still works.

Pi-lomar software on the RPi will TRY to recognise a conflict if it thinks that the GPIO cable AND the USB cable are connected at the same time. It will try to restrict the power to just the USB cable, but there are no guarantees that this will work!

Microcontroller communication problems.

If everything is set up as per the instructions the communication between the RPi and the microcontroller should work straight away. However O/S updates or alternative setups may cause communication problems.

The communication is performed using a serial UART connection between the RPi and the microcontroller. It does not use the USB or debug-serial ports on either device, it must be set up using wires between the correct IO pins on each device.

Common problems with the UART communication are :-

- Double check that you have configured the serial ports correctly when you built the RPi. Check in raspi-config (terminal) or Raspberry Pi Configuration (graphical). In the Interfaces section you must have SERIAL PORT enabled, but SERIAL CONSOLE disabled.
- The RX->TX and TX->RX wires are the wrong way around. It is easy to connect RX to RX and TX to TX because the names are the same, but the wires must cross!
- A different operating system, or some local changes in your configuration on the RPi may cause the serial ports to be assigned differently. The software knows the default serial port assignments for the supported RPis and the recommended operating system. You can see what has been chosen, and what alternative UART connections may be possible by checking the About Microcontroller option on the Microcontroller Tools menu. You can assign different UART ports via the UARTOverride parameter in the parameter file.
- You may have a power problem to the microcontroller. If the microcontroller is not ON it cannot communicate.
- On newer modes (RPi5 for example) there is a separate dedicated UART port. The software does not expect to use this, it will still try to communicate via GPIO pins on the GPIO header.

If you are having problems you can use the Monitor Communication tool on the Microcontroller Tools menu. This will show the flow of messages SENT and RECEIVED by the Raspberry Pi. It can be useful to see if traffic is flowing on both directions sensibly. You can also send test signals from here to prove that the microcontroller is receiving the RPi's messages.

If you continue to have problems you may also want to connect the microcontroller via its USB port to the Raspberry Pi and then use the circuitpython IDE (eg Thonny) to monitor messages directly from the circuitpython program. You need to open the 'serial' window in Thonny to see that. As always: If you connect the USB to the RPi it is CRITICAL that you stop down the pilomar software first.

- Stop the pilomar program entirely. It will turn off the microcontroller.
- Connect the USB cable between the RPi and microcontroller.
- When the RPi desktop shows the microcontroller has connected it is safe to restart the program. The program will recognise the microcontroller and will know that it is powered via USB.
- If you don't do this, the microcontroller will receive power from the GPIO header **AND** the USB cable – they might be different voltages and you could damage something!

Which steppermotor drivers can I use?

Pi-lomar was developed using the DRV8825 stepper motor driver. There are other similar driver boards available, they are all likely to work with some minor wiring differences. Any motor driver that supports the stepper motors in use, and accepts DIRECTION and STEP instructions can probably be used with Pi-lomar, other control pins can all be left to their defaults or hardwired high/low as required.

The circuit schematic published with Pi-lomar supports 3 pin microstepping signals, but the software does not need to use those by default. You can permanently wire your microcontrollers to FULL-STEP or any chosen microstepping configuration if you need to. You can configure additional steppermotor drivers in the parameter file.

If you build a telescope using stronger stepper motors which require more than 2A per coil the DRV8825 driver will need to be replaced with something bigger.

Can I use microstepping?

Yes, pi-lomar supports microstepping, in fact it is recommended now for smoother operation. The 'DRV8825' board is pre-configured in the parameter file, you CAN change this if your circuit is different.

By default pi-lomar moves in FULL steps. The DRV8825 supports 2, 4, 8, 16 and 32 microsteps. The chosen value is stored in AzimuthMicrostepRatio and AltitudeMicrostepRatio parameters.

You can change this value from the default 1 to any of 1, 2, 4, 8, 16 or 32.

1 means the motor takes FULL steps only.

2 means the motor takes 1/2 steps.

4 means 1/4 steps.

Etc.

You can select from common combinations of microstepping values from the Motor Tools menu. This is safer than trying to edit the related parameters directly in the parameter file. If you change the microstepping or the parameter file you MUST restart the pi-lomar program and also reset the microcontroller. The easiest way is to power cycle the system.

Beware: Using microstepping reduces the power available from the motor and also slows down the motion of the telescope. With FULL steps, the telescope takes about 1 second to move 1 degree. At 32 microsteps it would take about 30seconds per degree of movement.

The stepper motors are not moving

There could be multiple reasons for this.

- The wiring or connections are faulty.
- The DRV8825 is faulty.
  Try swapping the two driver chips around, does the problem move switch to the other motor?
- The current limits on the DRV8825 need adjusting.
  Start with a low current limit and gradually increase it until you hear the motors engage.
- There is a power problem (is the 12V supply connected and on?)
  Often you will hear a very faint ‘hum’ from the motors if they are powered on, and eventually they will warm up a little. If they are silent and cold it suggests that there is no power to the motors at all. Either the 12V power is missing or the DRV8825s are not ON and ENABLED (you need the software running to enable the drivers).

How to test signals to the DRV8825

Remove the DRV8825s from their sockets.

You can now measure voltages and signals to each DRV8825 pin on the board. Check that each output pin can go high and low. Check the 12v and GND connections too. Don’t set the ‘fault’ input pin.

Using the Monitor Communication utility on the Microcontroller Tools menu:

Press ‘c’ to enter a manual command.

Enter a 'pin' command...

You can set all the pins to match what you want. You can even send a pulse to the ‘step’ pin which should move the motor a tiny amount.

```
pin {date} {name} state/on/off [duration [repeat]]
```

[repeat] Optional: Defines how many times to repeat the command.

[duration] Optional: Defines how long the signal stays at set value before reverting.

state/on/off: Turn ON or OFF the pin, or return its current state.

{name} The pin name (must be set when GPIOpin instance created).

{date} Standard message timestamp string. Can be any valid 14 digit timestamp.

Examples

```
pin 20240307013045 dir on
```

Would turn the 'dir' pin on and leave it on.

```
pin 20240307013045 dir on 0.5
```

Would turn the 'dir' pin on for 0.5 seconds then turn it off.

```
pin 20240307013045 dir on 0.25 10
```

Would turn the 'dir' pin on for 0.25 seconds then off for .25 seconds. It would do this 10 times.

```
pin 20240307013045 dir status
```

Would return a comment message to the RPi with the current state of the 'dir' pin.

Output pin names :-

'azstep' GP29 Azimuth Step signal (azumuth DRV8825 only)

```
pin 20240307013045 azstep on
pin 20240307013045 azstep off
pin 20240307013045 azstep on 0.25 10 <- Will send 10 STEP signals,
clear slow pulses.
```

'altstep' GP28 Altitude Step signal (Altitude DRV8825 only)

```
pin 20240307013045 altstep on
pin 20240307013045 altstep off
pin 20240307013045 altstep on 0.25 10 <- Will send 10 STEP signals,
clear slow pulses.
```

'dir' GP27 Common Direction signal (Both DRV8825 DIR pins)

```
pin 20240307013045 dir on
pin 20240307013045 dir off
```

'mode0' GP3 Common Mode 0 signal (Both DRV8825 MODE0 pins)

```
pin 20240307013045 mode0 on
pin 20240307013045 mode0 off
```

'mode1' GP4 Common Mode 1 signal (Both DRV8825 MODE1 pins)

```
pin 20240307013045 mode1 on
pin 20240307013045 mode1 off
```

'mode2' GP5 Common Mode 2 signal (Both DRV8825 MODE2 pins)

```
pin 20240307013045 mode2 on
pin 20240307013045 mode2 off
```

'enable' GP2 Common Enable signal (Both DRV8825 ENABLE pins)

```
pin 20240307013045 enable on
pin 20240307013045 enable off
```

Input pin names :-

'azfault' GP6 Azimuth Fault (Azimuth DRV8825 fault pin only)

```
pin 20240307013045 azfault status
```

'altfault' GP7 Altitude Fault (Altitude DRV8825 fault pin only)

```
pin 20240307013045 altfault status
```

You can set the pins in various configurations with the above commands, so you can match the signals up with the DRV8825 datasheet and should be able to make the motors move any way you want if the DRV8825s are back in the sockets.

The stepper motors are not running smoothly

You want the stepper motors to run as smoothly as possible. Here are some thoughts on how to improve the movement.

Adjust the STEP signal

The motorcontroller sends pulses called 'step signals' to the driver boards. The signal is a brief OFF/ON/OFF pulse that causes the motor to move 1 step of rotation. If the motor is trying to move but not always succeeding you may need to adjust the length of the 'ON' pulse. You can do this in the parameter file by adjusting the following values.

- FastTime = 0.001 # The shortest length of the 'ON' pulse when moving at full speed.
  Larger numbers are longer pulses which may be more reliable for the motor.
  Too small a number may mean the pulse is not long enough for the motor to move.
  The program's processing time will also restrict how fast it can trigger a pulse.
  RP2040 microcontrollers can handle about 0.001 as the fastest speed.
  RP2350 microcontrollers can handle about 0.0005 as the fastest speed, this is quieter and smoother.
- SlowTime = 0.05 # The longest length of the 'ON' pulse when starting to move.
  Larger numbers are longer pulses which may be more reliable for the motor.
  Too small a value may mean the pulse is not long enough for the motor to move.
- TimeDelta = 0.003 # The acceleration rate.
  Larger numbers mean the motor accelerates from 'SlowTime' rate to 'FastTime' rate.

Note: In earlier versions of the pilomar software these parameters were hard coded into the src/pilomar.py program. I suggest updating your pilomar project to a more recent version to make this easier.

Adjust the microstepping settings.

When moving with full steps the signal to the motors is very abrupt. By increasing microstepping you send a smoother signal to the motors. This slows the motion but makes it smoother.

Adjust the current to the motors

Each stepper motor will have a current rating. If the motor is not moving cleanly, or not holding its position well the simplest thing is to adjust the current limiter on the DRV8825 chip (check online for instructions). The higher the current the more cleanly the motor will move, and the stronger its hold. But avoid excess current to the motor and/or DRV8825 as you may damage them. The DRV8825 can handle maximum 2A per coil (4A per motor) even with heatsinks to help it keep cool!

Adjust the friction within the gears

The gear mechanisms are very adjustable, if you have too little friction then the gears may slip, if you have too much friction then there may be too much resistance in the system. Try relaxing the pressure where the gears contact each other so that the gears are still fully meshed, but have enough space to move cleanly.

Remove one motor from the board

By having only one motor connected you can see if any instability is a power problem. The motors can draw power even if not moving. If your power supply or circuit is not delivering enough power the motors may not move or hold position well. Having a single motor attached effectively doubles the power available. If the single motor moves more precisely then you may need a stronger power supply.

Can you turn the status lights off on the electronics?

There is a RGB status light on the Pimoroni Tiny microcontrollers and the two status lights on the Raspberry Pi 4B.

You can disable the Tiny's status light by editing MctlLedStatus in the parameter file. TRUE leaves the LED on, FALSE leaves the LED off.

The two status lights on some Raspberry Pis can be disabled by changing some configuration files. Check online for the instructions for your particular model of the RPi. The instructions vary by model.

Can I have GPIO and USB connections to the microcontroller at the same time?

If you want to update the software on the microcontroller you can remove the GPIO header and make just a USB connection. Then update/run the software via Thonny.

But you will not have any UART communication with the RPi. So the pilomar.py and code.py do not get a chance to communicate. This makes testing difficult.

If you want the UART communication running too, you can do the following…

1) Remove the GPIO header.
2) Use 3 jumper cables to make selected connections between the GPIO socket and the ribbon cable.
3) The GPIO header on the circuitboard marks the GND, RX, TX pins.
4) Use 3 jumper cables to connect these pins into their appropriate sockets on the ribbon cable.

That will connect JUST the UART communication, allowing you to test the entire suite without risking power issues etc.

If you add or remove the USB connection to the microcontroller you MUST stop the pilomar program first. It will only recognise changes when it is restarted. If you add a USB connection while the program is running – even just sitting on the main menu – it will not recognise the change and may allow damaging power conflicts to arise. You may damage the microcontroller and/or the RPi.

What if an item is missing from a target catalog?

The catalog of potential targets come from various sources. Some lists such as the Messier and NGC catalog are physical lists. If you notice that an observable target is not in the catalog you have options.

1) If you know the Right Ascension and Declination of the target you can select it as a RADEC target instead.

2) If the target is observable via the pi-lomar system you are welcome to suggest an addition via the Github repository. You can raise issues or add the information to the discussion area there. Please provide the RA/DEC coordinates of the target as a minimum.

How do I make or suggest improvements to the programs?

Pi-lomar is maintained and distributed via github.
https://github.com/Short-bus/pilomar
All suggestions and pull requests will be gratefully received and considered.

If your improvements don't fit with the core project you are welcome to create your own version, but please respect all the appropriate licensing terms and give credit to the other people who have contributed to the original software and the included component parts.

The program is failing to download target data files.

When you first run the software it will require an internet connection because it may download some catalogs from the internet. These catalogs are saved and do not generally need to be downloaded again.

The datafiles are often maintained by voluntary sources and sometimes the names of the files can change, or the format of the files can change. If a file is nolonger available it is worth searching online to see if anyone else has the same issue. There has been at least one case where the Hipparcos catalog changed compression method which caused problems with the underlying packages. The package impacted was modified to handle the situation!

The dates and times are wrong

Pi-lomar displays all dates and times in UTC. Your local timezone may be different from this. If the UTC time is wrong then check the configuration of the Raspberry Pi timezone.

How do I check/change the parameters

There is an option on the menus to edit the parameter file. You can edit it manually with your own editor, or use the menu option to open the file in the nano editor. When you use the menu option, Pi-lomar will save a backup of the previous version. After editing the parameter file, pilomar will insist that you restart the software for safety. Some parameter changes need to propogate through the system, so a restart is safest.

Logging

Pi-lomar writes two separate log files into the log directory. The files are uniquely timestamped. After 48 hours the log files are deleted to preserve disc space. The timestamp represents the UTC startup time of the program.

- log/pilomar_yyyymmddhhmmss.log
  Contains program log from the main process of pilomar.
- log/pilomar_camera_yyyymmddhhmmss.log
  Contains program log from the camera process of pilomar.

If an error message is reported to the log file it is retained in memory and redisplayed when the observation ends, so that you can also see the error summary on the screen. This may not report all errors, it depends what the problem is!

The parameter file is corrupted how do I recover it?

If you want to reset the parameter file, just delete (or rename) the current file. When Pi-lomar starts it will generate a new default parameter file. You will have to re-enter your home co-ordinates in the new file, the program will tell you what to do.

The file is in the data directory.

/home/pi/pilomar/data/pilomar_params.json

The software terminates saying that another copy is already running.

If you have a network failure or somehow lose the link to the session running on the Raspberry Pi, the copy of pi-lomar running on the RPi may still be operating.

If you open a new session and try to start pi-lomar again you will have 2 copies running at the same time. They will conflict over some resources. The second copy will terminate and warn you that pilomar is already running.

Restarting the RPi will clear the old copy, or you can find and kill the pilomar process from the command line. The error message will include a list of all the pilomar processes running on the RPi.

If the kill command cannot clear the old session you may need to reboot the RPi.

Even if you have lost the terminal connection to the original session that will stop moving the telescope automatically after a timeout period. But when you restart pi-lomar make sure that the telescope positioning is accurate, you may need to use the TUNE options to correct the motors. In general any error is minor and the tracking system will automatically correct it after a few minutes.

The software gets killed by the operating system.

```
./pilomar: line 1:  2419 Killed                    python3 pilomar.py
pi@pilomar:~/pilomar/src $
```

The most common reason for the operating system to 'kill' the program is because of memory constraints. You will probably notice slow performance in this situation before the operating system intercedes.

This is most likely to happen on smaller, older Raspberry Pis with the 32bit operating system.

The Raspberry Pi 3B with 1GB of memory can run the software in its basic configuration, but if you activate some of the more advanced image handling features you may exceed the memory available.

If you are getting these symptoms check the parameters for your instance and see if you can disable some of the less critical features.

This situation has been observed when running the Raspberry Pi without a camera installed, or when it is switched to simulating images instead of actually taking photographs. Try switching off some of the advanced simulation options such as 'FakeNoise','FakeField','FakePollution' and 'FakeAurora'.

The software reports microcontroller resets

If this is a rare occurrence then it is probably not worth worrying about. The software is generally good at recovering from a microcontroller reset. It may take a minute or more for the software to recover fully because it will need to download a new configuration and trajectory to the microcontroller. But it generally recovers nicely.

If this is occurring multiple times, then check for power supply issues or try swapping for a new microcontroller. I have seen frequent issues with Raspberry Pi Pico and Adafruit Feather RP2040 boards where they reset randomly even when running completely independently of the telescope. This is why the current design uses the Pimoroni Tiny microcontroller, this has proven extremely reliable so far. The Tiny microcontrollers have more storage compared to the 2MB for the Pico, that may also have something to do with the increased stability!

USB storage is not found

The Raspberry Pi should be configured in raspi-config to boot to the desktop, even if you are running headlessly. If the desktop is running then the USB memory will be automatically mounted.

Desktop IS running

Check that the USB memory is formatted with a suitable file system. Not all options work in Linux.

Pi-lomar expects USB memory to be mounted under /media/pi. If you have it mounting to some other path you need to provide that new path in the parameter file by changing the USBPath parameter.

Desktop is NOT running.

If you do not have the desktop running, Pi-lomar will try to auto-mount USB memory sticks if found, but this is less reliable.

The following error sometimes appears when using USB storage without the desktop running.

discmonitor.FindUSB: Check {devname} as {path} not found in df listing.

This means that Pi-lomar found a USB storage device and thinks that it is already mounted. But when it performs a final check the device is not found. This is because Pi-lomar did not think it was necessary to issue a mount command.

There may already be a folder matching {path} on the SD card. Pi-lomar may mistake this for a mounted USB memory stick. Try to remove the folder named in {path} then restart the program.

I recommend labelling the USB storage as “USBMEMORY” when you format the memory stick. That will help pi-lomar to recognise the extra storage space.

Observation issues

The network drops during an observation

If you are running the telescope headlessly, accessing it over the network there is a risk that the network connection drops for some reason.

- The telescope may lose its connection.
- Your workstation may lose its connection.
- The router may restart.

If you are using a VNC connection to the RPi remote desktop there is little risk of problems. The session will continue on the telescope and you can rejoin it safely when the network recovers.

If you are using a puTTY remote connection to a terminal session there are some potential problems. You will get a ‘zombie’ session running with no user access.

- The RPi will continue running for some time before it notices that the session is lost. It will then abruptly abort the session.

- The pi-lomar software may not have the chance to save the state of the telescope correctly when it dies.
- You will not be able to reconnect to that session even if the network recovers.
- If you start a new session, the old zombie session may still be exclusively accessing the system and assets.
- Even if the session dies, the motorcontrol board may continue to follow its trajectory plan until it is reset. The motorcontrol board may run for 20 minutes or more if it has a fresh trajectory loaded.

Actions to take when recovering a broken puTTY session.

- You can abort the previous zombie session using the 'kill' command. Pi-lomar will suggest this and list the processes if you try to start a fresh session while the earlier one is still running.
- When restarting the software it is wise to 'home' the telescope first. And check that it homes to the correct location before resuming. The telescope may move some considerable distance before you regain full control.

If you have an unreliable network, then it is wise to perform your observations using a VNC connection as this will keep everything running on the RPi even while the network is down, and you can reconnect safely to resume control when the network returns.

However, VNC connections sometimes provide smaller terminal windows – which means you may not see ALL the possible detail from the dashboard. You will always see the critical data though!

I generally use puTTY sessions when developing and testing, but VNC sessions when making actual observations.

The telescope aborts an observation

The software may terminate the observation for many reasons.

- The target is nolonger visible.
- The chosen number of images has been captured.
- The telescope has reached its movement limit.
- A fault has been reported by the motor control system.
- A critical resource is not available (disc space etc).
- Some software fault has occurred.
- You pressed the 'x' key to terminate the observation.

In 'debug' mode the Python error messages will be visible on the screen, however when displaying the observation dashboard the error messages may be overwritten by the display when it refreshes.

The log files generally hold enough clues to show where the error occurred. If you cannot identify the error – try to recreate it with the telescope running in 'Debug Mode' so that all error messages remain visible. You can activate Debug Mode from the menu or via the Parameter file.

What happens if the pi-lomar software itself crashes?

If the motorcontroller has a trajectory, the motorcontroller will continue moving until the trajectory is completed! But pi-lomar will not be running on the RPi, so the final position of the telescope will be lost.

If this happens, be prepared to home and tune the telescope position when you restart a fresh session.

The motorcontroller has some logic to try to reduce the impact. If it notices that the RPi is nolonger communicating it will abandon the current trajectory. If this happens then the position error will be much smaller. You may be able to resume the session with a new connection and pi-lomar's tracking solution will automatically correct for the error.

The telescope will not start an observation

The software may refuse to start an observation if the target is nolonger visible, or if there are other reasons (out of disc space etc). Normally the program will tell you what the problem is. If you need to dig deeper Pi-lomar generates a couple of log files on the /log folder.

See Troubleshooting/Software/Logging

Camera issues

The camera reports that it is hung and the telescope needs restarting

There are occasions where the camera board itself seems to hang. The cause is not known, but may be related to power problems to the camera board. If this happens, the camera never completes taking a photograph and gets stuck. Sometimes the camera starts responding again after a very long delay (20+ minutes!) The only solution at the moment is to shut down the software and power cycle the whole telescope.

The telescope is not saving the .JPG, .DNG or .FITS images during observations

This is a parameter setting. You can select the image types to be generated from the Camera Tools menu. Try different image combinations.
NOTE: You can only save FITS images when running on the Bookworm 64bit operating system. The option is not available on the older Buster 32bit O/S.

Can I use different lenses?

Yes, Pi-lomar has run successfully with the 16mm telephoto lens and also with the Arducam 50mm telephoto lens. You will have to modify some parameters in the parameter file if you change the lens, but with some trial and error you can configure the system to work with the new lens.

I recommend starting with the 16mm lens. This has a wider field of view so objects are relatively small in the image, but it is the easiest lens to learn with. The telescope features will work without needing precise setup. When you are confident using the telescope with the 16mm lens you could swap out for a longer lens.

50mm is about the longest focal length you can use. The resolution of the telescope motors is not fine enough to handle much stronger lenses. Above 50mm, the magnifying power is probably too great for the tracking system to work effectively.

The 16 and 50mm lenses are budget items, they do not have the fine optics of SLR lenses. You can get CS adaptors to mount larger SLR style lenses. You may need to design a new camera cradle if your SLR lens is quite large, but in theory you can operate the telescope with up to 50mm higher quality SLR lenses.

If you use some other focal length lens you will need to calculate the field of view for the camera. There is a utility on the Camera Tools Menu to help you do this.

NOTE: The Hi Quality Camera sensor is much smaller than a regular 35mm frame, so you get a magnifying effect. The 50mm lens mounted in the telescope has the same magnifying power as a 280mm lens on a traditional camera.

Which parameters are related to the lens?

TrackingExposureSeconds

The exposure time to be used for taking tracking images. Longer = More stars.

TargetInclusionRadius

This is a filter to the Hipparcos catalog, it defines how wide a field of stars to select for tracking calculations. As the focal length gets larger, this value can be reduced to improve performance.

TargetMinMagnitude

This is the minimum magnitude of stars selected for the tracking algorithm. As the focal length gets longer you need to increase the magnitude of stars selected. Short focal length lenses already capture a lot of stars, so you can use a lower magnitude cutoff.

LensLength

Tell the system which focal length you are using.

LensHorizontalFov

Tell the system the field of view (width) that the camera/lens combination is capturing.

LensVerticalFov

Tell the system thefield of view (height) that the camera/lens combination is capturing.

Tracking / targeting issues

What is the purpose of drift tracking?

If the telescope is set up perfectly level, facing the correct starting position and motors are running smoothly, then it should find and follow all targets automatically.

If any of the initial setup or running conditions are slightly wrong then the telescope may not point correctly to the target, or may fall behind as the target moves. The drift tracking mechanism is an attempt to recognise small position errors and send corrections to the motors.

The mechanism here is not as precise as some other solutions, it is not a star tracker, and is not performing true plate solving. However if working it should help the telescope to stay roughly on target during observations.

Individual images captured in an observation set do not have to be perfectly aligned with each other. The image stacking software which combines them into the final image will take care of small alignment differences easily.

How does Pi-lomar’s drift tracking work?

Pi-lomar uses its database of stars and planets to create an ‘target’ image that it expects to see through the camera. Think of the ‘target’ image as a map of the sky. If the telescope is perfectly set up and moving smoothly the images captured by the camera should be a very close match to the ‘target’ image map.

Pi-lomar compares the ‘target’ image map against the camera’s current view. If it recognises the same stars in each image it measures the difference in locations. If there is a large enough difference it assumes that the telescope is slightly off target and sends corrections to the motor controller.

Very small differences are ignored because the stacking software will deal with those alignment issues automatically.

If the telescope is very far from the target then the camera view may not have any recognisable stars in the image. The tracking calculation will fail to find a match in this case.

For differences of 2 – 3 degrees the telescope will detect the tracking error and send a correction to the motor. These corrections are 'Tune' commands.

If you are familiar with the concept of 'plate solving' then Pi-lomar's drift tracking calculation is a simplified approximation of this type of operation. It balances some precision and capability for acceptable performance on the small Raspberry Pi computer.

How critical is drift tracking?

The automatic drift tracking is not critical to the operation of the telescope, it is just an assistance measure. You can manually correct tracking errors during the operation from the observation sub menu, and if the telescope is set up and running perfectly it should always be pointing to the correct position.

If you have problems getting drift tracking working in your case you can still use the telescope, but pay more attention to the initial set up of the telescope and make sure that the motors and gears are running smoothly.

What targets does pi-lomar recognise?

Stars and planets are the main objects that are used for drift tracking calculations.

How do I initially set the telescope up at the start of an observation?

Have the telescope 'homed' when you start an observation.

Place the telescope on a flat surface with the camera pointing due south at the horizon.

Select your target and start the observation.

The telescope will move to point at the target.

You can visually check that the telescope is pointing at the target in the sky. If the azimuth is slightly wrong you can just move the entire observatory to align it more closely with the target. If the altitude is slightly wrong you can use the TUNE ALTITUDE option on the motor submenu.

The target tracking image has too many or too few stars.

The 'actual' and 'target' tracking images should have a similar number of stars in them. Ideally about 30 stars should be visible. If the 'target' tracking image has too few stars you can increase the TargetMinMagnitude parameter to select dimmer stars. If there are too many stars, decrease TargetMinMagnitude.

NOTE: The LocalStarsMagnitude parameter acts as an upper limit on the TargetMinMagnitude parameter. If you want to increase TargetMinMagnitude beyond the LocalStarsMagnitude value you must increase LocalStarsMagnitude too. LocalStarsMagnitude is a filter used to reduce the size of the Hipparcos catalog used for an observation to increase performance.

The actual tracking image has too many or too few stars.

The 'actual' and 'target' tracking images should have a similar number of stars in them. Ideally about 30 stars should be visible. If the 'actual' target image has too few stars you can increase the

TrackingExposureSeconds, it will take a longer photograph and capture dimmer stars. If there are too many stars then reduce the parameter instead.

The tracking function will try to enhance the image to make the stars clearer. If the image has very low contrast this enhancement can create a large number of false stars. In this case you can try disabling the enhancement by changing the PrepImagesForTracking parameter to False. There are several reasons for low contrast.

- Haze if photographing something low in the sky.
- Clouds or fog in the image.
- The sky is not dark enough.
- Light pollution or stray moonlight entering the image.
- TrackingExposureSeconds parameter is far too short. (Image is black)
- TrackingExposureSeconds parameter is far too long. (Image is burned out)

How do I know the telescope is on target?

The program periodically generates a ‘preview’ image in the preview folder.

These preview images take a recent live image from the camera and superimpose some scales and descriptions of the expected image.

By examining this image you can usually tell if the telescope is on target or not. If you are ‘close’ to the target then you will be able to see how to tune the telescope by reading the targeting scales in the image.

If the telescope is a long way off the image then the preview labels will not match at all with the live image shown. In this case you may want to manually adjust the position of the telescope to more closely align with the current location of the target in the sky.

How do I correct the positioning of the telescope?

You can use the TUNE commands on the motor submenu.

During an observation press the ‘m’ key to get the sub-menu.

Then select TUNE AZIMUTH or TUNE ALTITUDE options.

You can then give a number of STEPS that the motor should move to adjust its position.

The PREVIEW image gives a good estimate of how many steps are needed to move the camera by the required amount.

This is the same mechanism that the software uses automatically to ‘auto track’ the target during an observation. It calculates the number of steps to move by comparing the live images with the theoretical image it expects to see.

If the telescope is not on a completely horizontal surface it may not find or track all objects accurately over time. The auto-tracking will correct for small differences, but if it is very far from level you may need to adjust the telescope feet or the platform it is sitting on to make it more horizontal.

Changing tracking behaviour

The drift tracking mechanism creates an ‘target’ image map the same size as the camera’s field of view. This is generally good enough for most installations. It gives a good balance of performance vs detection rates. You can tell the software to generate a larger ‘target’ image map. This will slow down the drift tracking calculation, but the software can search a larger map of the sky looking for a match. This should allow the telescope to recover from larger position errors which may be useful if you cannot set the telescope up very precisely at the start of the observation.

Processing the images

What is image stacking, how do I do it?

Image stacking is a technique which combines many separate images of a target and pulls out more detail than is obvious from a single image. It uses statistical techniques and correction algorithms to deduce more subtle information.

Pi-lomar does not perform image stacking, there are freely available image stackers out there with their own documentation and support networks. Look up ‘free astrophotography stacker’ online and see which tool suits your projects. I’ve had some success with Deep Sky Stacker, but there are others out there too.

Why does Pi-lomar disable the sensor’s “on-chip cleanup”?

The IMX447 sensor of the Hi Q camera has some built in logic to clean up the images that it captures. If you look up how camera sensors work you will see that there are a number of things done to produce the image you recognise as a photograph. Astrophotography stackers have their own routines tailored to astronomical photographs. The automatic image cleaning that the sensor performs can interfere with the best results that the stacking software can achieve.

Therefore Pi-lomar can send commands to the sensor to disable the cleanup option. This means that the stacking software gets a more pure raw feed of the data that the sensor captures.

You can turn this feature on and off from the parameter file.

For Raspbian Buster O/S with raspistill camera utility. Change the DisableCleanup parameter in the parameter file.

For Raspbian Bookworm O/S with libcamera utility. You can edit the camera command templates in the parameter file instead. Add or remove ‘--denoise off’ to each command template.

Example images

16mm Telephoto lens, Infrared filter removed. Orion

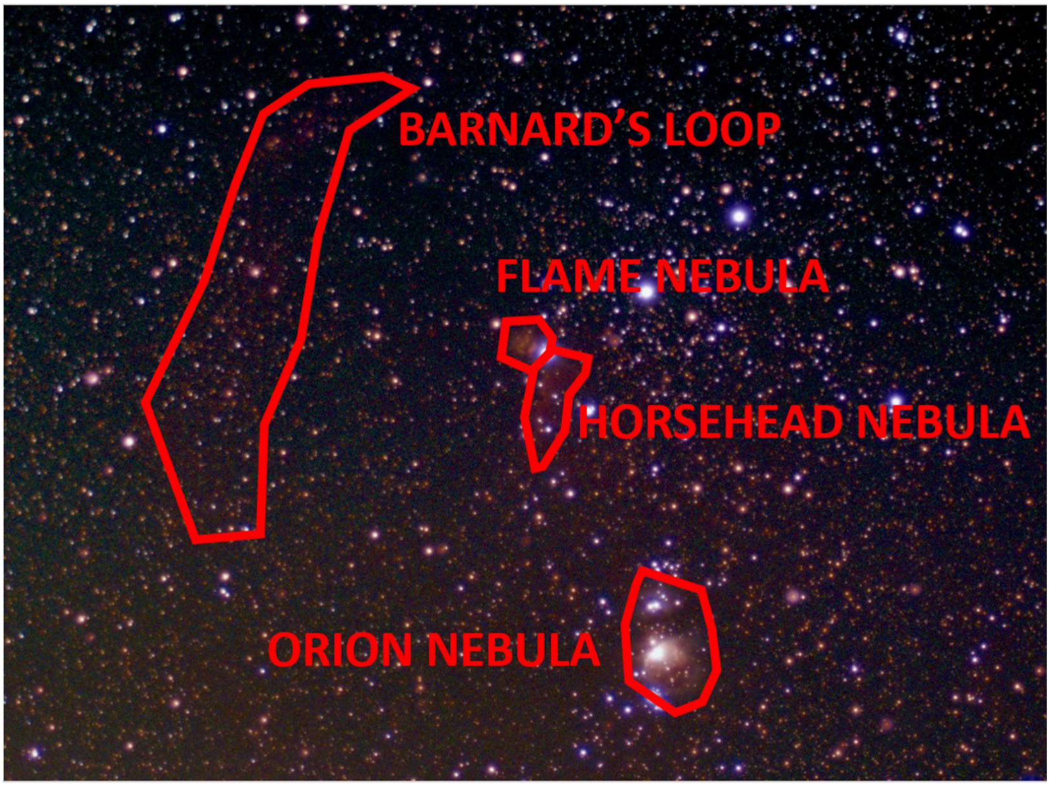
Conditions: Light pollution, mist/haze

Single frame

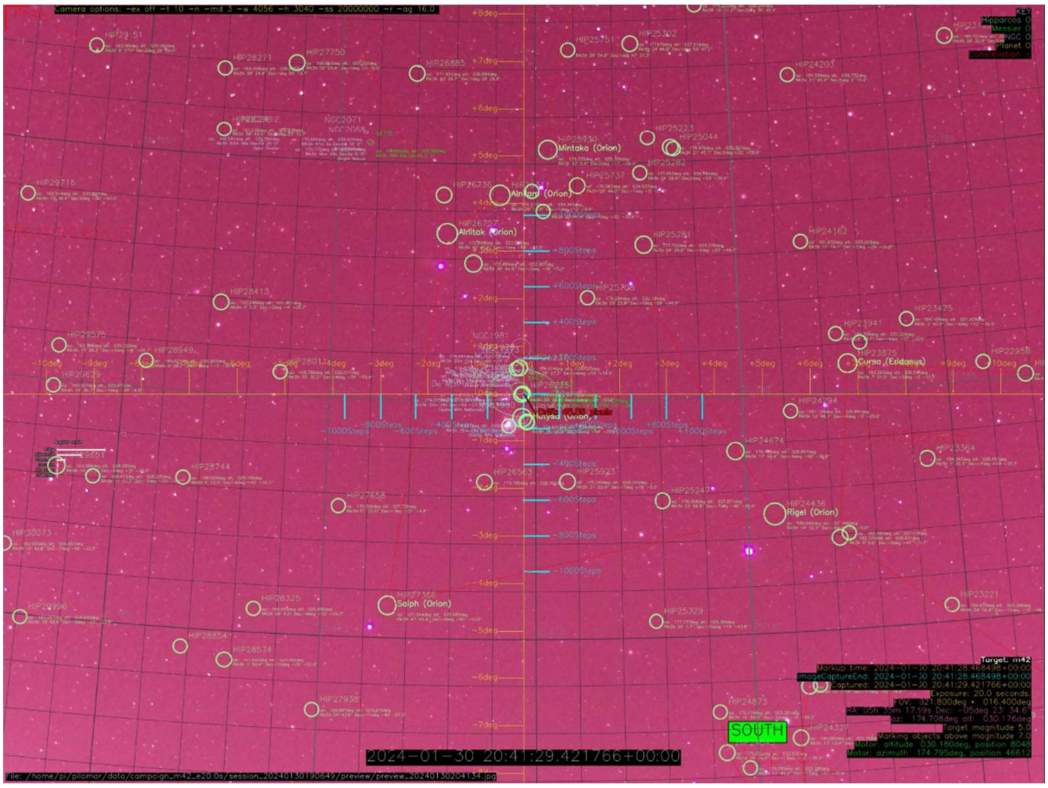
Stacked

GIMP cleaning

Manually annotated



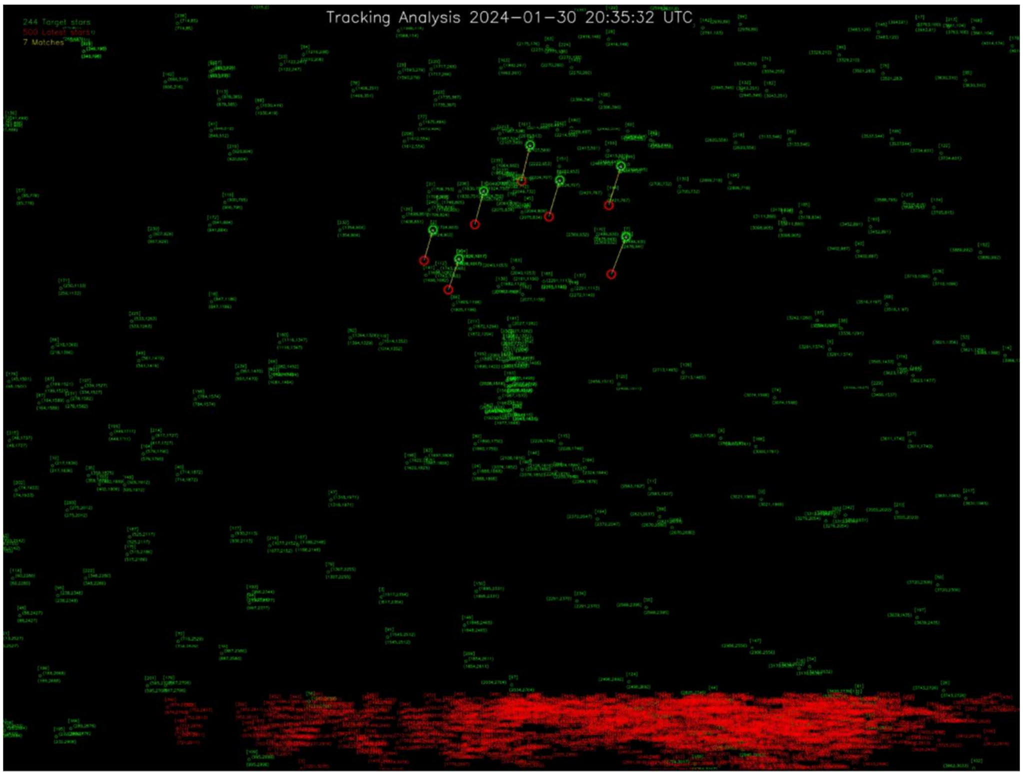
Preview image



As you can see on the preview image earlier, the label positions of objects are not 100% accurate, but should be close enough to help identify the image contents. This is also the precision of the drift-tracking, it is good enough to keep the camera on the target, but the telescope relies upon the alignment capabilities of the stacking software to align all the frames with precision.

The ‘light’ and ‘preview’ images show very strong ‘pink’ cast, this was taken with an IR sensitive sensor and the gains / white balance of the image is shifted as a result. But even this strong shift in colors is handled nicely by the stacking software.

Tracking analysis



The tracking analysis image gives some idea how well the drift tracking is able to understand the image. In this example the major stars in Orion's Belt have been identified and used for alignment. However you can see that the strong haze in the lower half of the images has caused a lot of false stars to be recognised. The drift-tracking algorithm in the AstroAlign package concentrates upon the brightest/strongest stars in the images, so has ignored the mass of false stars caused by the haze. In good observation conditions the false stars do not appear.

Notes

This observation was made in challenging conditions, there was considerable haze and light pollution obscuring the target which was quite low in the sky. However it was still possible to make an observation with interesting detail.

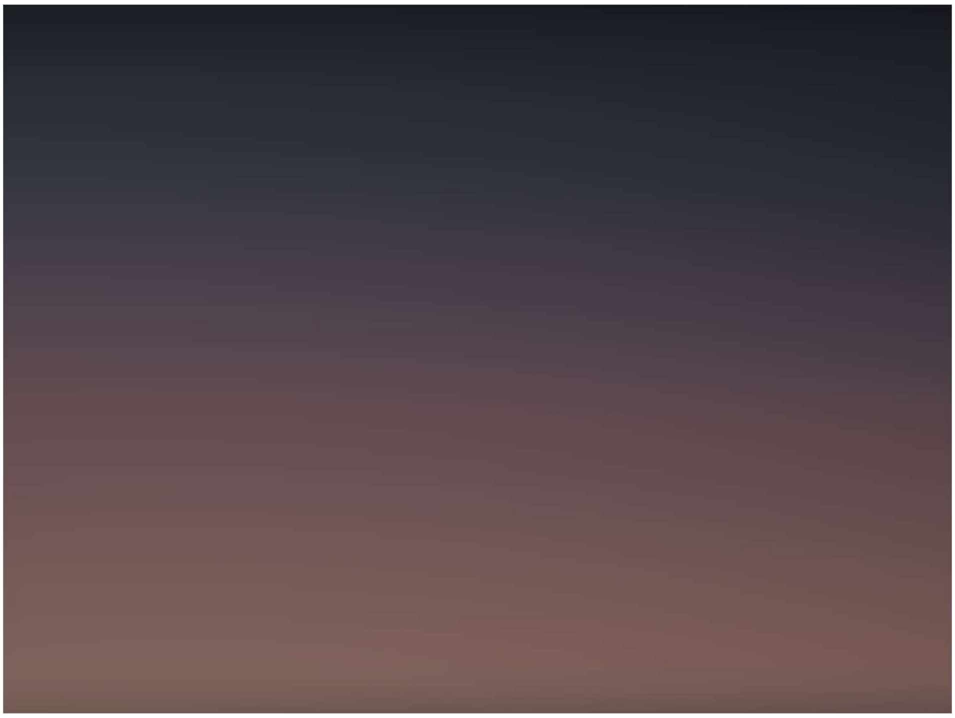
This image was captured using the Raspberry Pi Hi Quality sensor with the infrared cut-off filter removed. It used the 16mm telephoto lens which captures about a 20 degree wide field of view.

The image is the result of 141 20second images being stacked with DeepSkyStacker 5.1.

If you leave the infrared cut-off filter in place on the sensor you can get good images of M42 Orion Nebula with some nice colour, the flame nebula will appear but can be weaker. In good observing conditions you will also start to detect the color surrounding the HorseHead nebula.

If you remove the infrared cut-off filter then the telescope is more sensitive to infrared wavelengths and some of the less obvious objects start to appear. The pink cast to the individual frames is due to the color balance being shifted by the additional infrared wavelengths. DeepSkyStacker seems to correct for this color shift quite nicely.

Haze filter calculated in The GIMP



The original stacked image still contained a lot of haze and light pollution in the lower half of the image, so it was further processed via GIMP to reduce the haze. This was done by creating a duplicate layer of the image and using the Gaussian Blur filter on very high settings to create a gradient image of the general background brightness of the image (see below). Then set GIMP to subtract this new layer from the original image.

Discussions

How does Pi-lomar's drift tracking work?

Pi-lomar generally works by dead-reckoning. It does not have position feedback, it relies upon the telescope being correctly positioned to star with and then calculates where it should point to and how it should move.

If all is well the telescope will keep on target very well for a long time. But sometimes there may be errors in its positioning.

1) The telescope may not be positioned correctly. It must point due South (in the Northern Hemisphere) and be on a horizontal surface.
2) The telescope gears or motors may be slipping for some reason. Perhaps mechanical wear, something is jamming them, they may need position adjustments or a power problem to the motors.

In general small differences of 1 or 2 degrees will not impact the final stacked images, the stacking software will easily correct for any tracking errors between the individual photographs.

However Pi-lomar employs a technique to keep the telescope roughly on target so that the image stack is useful. Larger telescopes use many different techniques to do this, but we have limited options on the simpler Pi-lomar structure.

Pi-lomar's tracking technique is to periodically take a standardised photograph of the sky. Capturing what it can see at the moment. It then generates an expected image (map) of the sky, using calculated star positions to create a 'target'.

Pi-lomar compares the latest live image against the calculated target image (map) using the 'astroalign' Python package. The difference between these two images indicates if Pi-lomar needs to finetune the camera position. If the different is large enough Pi-lomar calculates the adjustments itself and passes them to the motors.

In this way Pi-lomar can roughly correct for any inaccuracies in setup or the mechanical state of the telescope. It will not be pixel perfect alignment, it does not need to be.

Tracking can go wrong for many reasons. It is optimised for the 16mm lens, if you use a different lens the field of view and the stars captured will be different. There are a few parameters available to tune the behaviour of the tracking calculation. You can change the exposure time of the reference photograph and change the construction of the target image too.

If the drift tracker is recognising too many or too few stars you can make some adjustments in the software to improve things.

- Increase/decrease the photographic sensitivity by changing the 'TrackingExposureSeconds' time in the parameter file.
- Adjust the star detection rate by changing the min/max values in the pilomarimage.CountStars() method. Minval and maxval specify the min and max lower and upper sizes for stars being detected in an image. They specify the 'area' of the stars in pixels.
- Adjust the number of stars included in the calculated target image by changing the 'TargetMinMagnitude' value in the parameter file.

How do Pi-lomar's filter scripts work?

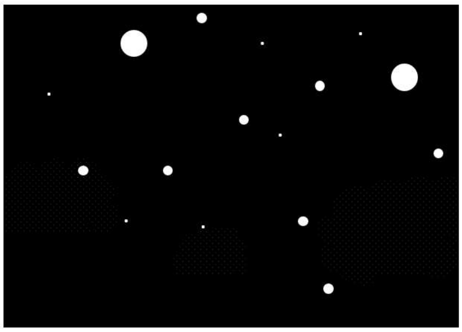
The images captured for actual stacking ('light' images) are not processed in any way by pi-lomar. The aim is to preserve as much detail as possible for the stacking software to work with.

However the images captured for the drift tracking mechanism CAN be cleaned and enhanced. A clean and crisp image of the stars in view will help the tracking mechanism identify the position more easily.

Several things can degrade the quality of the tracking images captured. The most common reasons are due to light pollution or haze in the sky.

Pi-lomar can use some simple 'opencv' routines to enhance the tracking images, making the stars more clear. It uses a simple 'scripting' mechanism which allows you to adjust the filter actions easily.

Without UrbanFilter



With UrbanFilter

There are two filters already defined when you build pi-lomar, you can add more if you want via the parameter file.

The two filters are :-

- EnhanceStars

  EnhanceStars converts the image to Grayscale, applies a threshold to remove clouds, uses a Gaussian Blur to enlarge the remaining stars, then a final adaptive threshold to make them more crisp.

- UrbanFilter

  UrbanFilter converts the image to Grayscale, calculates and subtracts the background light gradient from the image, uses a Gaussian Blur to enlarge the remaining stars, then a final adaptive threshold to make them more crisp.

Activating a filter.

The easiest way to do this is to use the "*Set Latest Tracking filter*" on the Development Tools menu. This lists all the recognised scripts from the parameter file and lets you choose the one you want. You can also test the action of a filter using the "*Test Latest Tracking filter*" option on the same menu.

Creating your own filters

It is very likely that your local observing conditions are different to the default settings, so you may want to make these filters behave differently. The default filter scripts in the parameter file provide examples of the different filters you can create. You are free to add more scripts to the dictionary or to adjust the values of the existing scripts.

See the FilterScripts parameter in the parameter file. This is a python dictionary that you can edit via most text editors. Your changes are saved in the parameter file and will be retained across future versions of the software.