flirpy Documentation

Josh Veitch-Michaelis

Sep 19, 2021

Getting started

Welcome to the documentation for Flirpy, a Python library for controlling thermal imaging cameras and processing thermal images.

CHAPTER 1

Overview

CHAPTER 2

Thermal cameras

Flirpy is designed primarily to control FLIR's camera cores - the Lepton, the Boson and the Tau 2. Each core has slightly different characteristics and use cases, so this overview will help you decide which one you need for your project and what Flirpy can and can't do with them.

2.1 Camera cores

Lepton

Boson

Tau

2.2 Resolution

The obvious difference between the various cores is their resolution. The Lepton has the lowest resolution, at either 80x60 or 160x120 pixels. The Boson and Tau are available in either 320x240 or 640x512 (a bit bigger than VGA).

2.3 Form factor

2.4 Interface

CHAPTER 3

Radiometry

Radiometry means that the images you capture from the camera contain absolute temperature values, rather than "counts". The value, brightness or intensity of an individual pixel is directly related to the temperature of the object being imaged. It is important to understand the processing pipeline that happens to produce these radiometric images.

Thermal imaging sensors can be crudely viewed as a 2D array of resistors - this is not entirely accurate, but it is sufficient to understand how thermal cameras work. When a material heats up or cools down, its resistance changes. The camera is able to measure changes in resistance over the entire sensor area by, for example, passing a reference voltage or current through each pixel. The resistance of each pixel is digitised by an analogue to digital converter (ADC) and stored as an image. This is what we might call a *raw* image.

3.1 Radiant flux

In order to convert raw counts from the ADC into temperature values, we use Planck's law for spectral radiance

$$I(\nu, T) = \frac{2h\nu^3}{c^2} \frac{1}{\left(e^{\frac{h\nu}{k_B T}} - 1\right)}$$

where:

- I is spectral irradiance (often B is used here, but for whatever reason, FLIR's calibration coefficients use B for something else.)
- T is object Temperature (Kelvin)
- ν is frequency
- c is the speed of light
- k_B is the Boltzmann constant
- h is Planck's constant
- e is Euler's constant

We can make some substitutions:

$$R = \frac{2h\nu^3}{c^2}$$

$$B = \frac{h\nu}{k_B}$$

to give:

$$S = \frac{R}{\left(e^{\frac{B}{T_K}} - 1\right)}$$

This formula is described in the *FLIR Tau 2 advanced radiometry application note* (section 8) and in the *FLIR Lepton with Radiometry Quickstart Guide*:

$$S = W(T) = \frac{R}{\left(\exp\frac{B}{T_K} - F\right)} + O$$

and rearranging for T:

$$T_K = \frac{B}{\ln\left(\frac{R}{S-O} + F\right)}$$

where S is the counts from the ADC and O is some offset. Here we've assumed that the response from the detector is linear with respect to object radiance. This actually requires another calibration which ensures that the signal is corrected for the temperature of the detector. Here we introduce F as a free calibration parameter with a typical value of 0.5 - 2, rather than 1 in the original derivation.

This formula is used for the radiometric Lepton and Tau2 operating in TLinear mode and the coefficients can be read from the camera internal memory directly. In practice you don't need to do this and the cameras will return pre-calibrated images if you've enabled radiometry internally (TLinear mode on). Some FLIR cameras (such as the Duo Pro R) use a slightly different parameterisation:

$$T_K = \frac{B}{\ln\left(\frac{R_1}{R_2(S-O)} + F\right)}$$

These coefficients are then are stored in the camera, or in image metadata. Here is a snippet from flirpy, reading from image metadata (Duo Pro R):

As you can see above, the conversion actually takes into account atmospheric conditions, object distance and so on. These tend to be fairly small effects, but they are important for precision radiometry. The most significant factor is usually emissivity. For a perfect blackbody, the emissivity is 1. Most objects are not blackbodies, however, and to accurately measure temperature on a particular object you need to dial in the correct emissivity (quite often you will find that IR "gun" thermometers have presets for common industrial materials like concrete). Similarly if you are integrating your camera into another system, for example with a window or other filters then you need to re-calibrate the system to take into account the additional optics.

3.2 TLinear mode

FLIR describes TLinear mode as follows:

> *In normal mode with TLinear disabled, the Tau camera outputs digital data linear in radiometric flux. In TLinear mode, the Tau camera outputs digital data linear in scene temperature.*

The raw image alone is not terribly helpful, because as you might expect, the sensor itself can heat up and cool due to ambient conditions or heat generated internally by the electronics in the camera. The first issue this causes is noise, you can also buy cooled thermal imaging cameras, but they tend to be a lot more expensive.

In summary there are two corrections. The first is a linear correction to adjust for changes in the focal plane array (FPA) temperature. The second is a fit to the Planck equation to calculate the object temperature.

Out of the box, currently only the Lepton 3.5 and the Tau 2 are available with radiometry.

3.3 Radiometric chain

What determines how much signal falls on our detector?

Consider a camera pointed at an object. The object has some emissivity, ϵ and it has a radiated flux proportional to its temperature, $\epsilon W(T_{\text{object}})$. Of course if we had a perfect blackbody, we could assume that $\epsilon = 1$. Since the object is not a blackbody, it also reflects radiation ($r = 1 - \epsilon$).

Suppose that there is also some background radiation, such as the Sun, or a hot lamp. The material will absorb some of that heat, and it will reflect the rest: $rW(T_{\text{background}})$.

Now we have two components: radiation emitted by the surface, and background radiation reflected by the surface.

This radiation then passes through the atmosphere, which attenuates some of it - it has a transmittance τ_{atm}. And of course since the atmosphere is also made up of matter at some temperature, T_{atm}, it will also emit radiation (and that radiation will also be partially attenuated). We now can compute the flux at the detector, S:

$$S = \tau_{\text{atm}}(\epsilon W(T_{\text{object}}) + (1 - \epsilon)W(T_{\text{background}})) + (1 - \tau_{\text{atm}})W(T_{\text{atm}})$$

If there is also a window in the way, then we need to correct for that as well. In this case, the window also has a transmittance τ_{win} and a temperature T_{win}. To add to the confusion, there will also be radiation emitted from the detector, with T_{det} that reflects off the window (which has reflectivity r_{win}). Some of *that* radiation will pass through the window and out into the scene, never to be seen again:

$$\begin{aligned} S = \tau_{\text{win}}\,(\tau_{\text{atm}}[\epsilon W(T_{\text{object}}) + (1 - \epsilon)W(T_{\text{background}})] + (1 - \tau_{\text{atm}})W(T_{\text{atm}})) + \\ r_{\text{win}}W(T_{\text{det}}) + (1 - \tau_{\text{win}} - r_{\text{win}})W(T_{\text{win}}) \end{aligned}$$

Fortunately, we can approximate atmospheric parameters to a reasonable degree, which is a function of ambient temperature and humidity, as well as the distance to the object. For most use cases you don't need to worry too much about this, especially if your object is fairly close. This is all calculated automatically by flirpy, but bear in mind that most drone cameras such as the Duo store fixed parameters for the scene (temperature, humidity, etc). For the most accurate results, you should monitor ambient conditions throughout your data capture process and correct each frame individually with the conditions at the time it was captured.

How much difference does all this make? In practice not very much for most people. In the equations above, your object is probably bright enough that atmospheric transmission is negligible compared to your uncertainty on the object's emissivity. Before putting a lot of effort into worrying about temperature and humidity, consider what measurement error is acceptable to you. Typically thermal imaging cameras are capable of highlighting very small temperature differences *within a scene*. This is a subtle, but important point. If your comparison objects are in the same image, then you can effectively ignore the atmosphere as it will attenuate/emit the same amount throughout the image.

CHAPTER 4

Further reading

For a more exhaustive overview, you are recommended to have a look at these books:

- Infrared Thermal Imaging, Vollmer and Mollman
- Infrared Thermography: Errors and Uncertainties, Minkina and Dudzik's

Vollmer and Mollman, in particular, is comprehensive at just shy of 800 pages. It covers a lot of common use-cases for thermal imaging such as building/infrastructure monitoring. It also has plenty of good illustrations and images.

CHAPTER 5

SEQ Files

Some cameras, like the FLIR Duo capture files in a format called *SEQ* or "sequence". This is a semi-proprietary format containing raw thermal images and metadata associated with them. That metadata includes things like the geolocation of the image, if available, and the calibration coefficients used to convert the raw image to radiometric. SEQ files are simply a bunch of images stacked one after the other.

The reason that these files exist is that there is no universally adopted standard for saving raw video along with the necessary metadata. There *are* formats which could be used, like the Flexible Image Transport System (FITS) format used ubiquitously in astronomy, but that doesn't seem to have caught on anywhere else.

There is limited documentation online about the format of an SEQ file, but through a combination of FLIR's documentation and enterprising developers it has been reverse engineered to the point where we can write software to split an SEQ file into its constituent images. This is useful for a few purposes including

- You want to do some sort of analysis on individual frames, for example as a dataset for machine learning
- You want to create a movie

Flirpy offers a simple utility to split out SEQ files into constituent files, as well as synchronising with videos captured with the RGB camera (as on the Duo Pro). Assuming you have a folder containing all the infrared and visible data, i.e. SEQ files and MOVs, just run:

This will create a folder called split subfolders for each data type.

- Preview images are 8-bit versions of the radiometric image (similar to "AGC" mode on the camera). This is useful for easy inspection of data.
- Radiometric images are 16-bit TIFFs which can be converted to physical units (multiply pixel value by 0.04)
- Raw images are FLIR FFF files, also stored here is the metadata for each frame in a text file
- RGB images are stored as JPEGs and are time-synchronised to match the IR frames.

The program will first split each SEQ file into a temporary folder, before merging these folders into one containing the entire "video". Merging is optional, but you will probably find it convenient to have all the images, in sequence, in one place. You can disable splitting certain file types by using the appropriate flag (e.g. *–no_export_tiff*).

Note that JPEG/TIFFs are preferred because they can be geotagged. Flirpy will automatically copy geolocation information from the base SEQ file to the preview, radiometric and RGB images ready for use with photogrammetry

software like Pix4D.

Full documentation can be seen by calling *split_seqs -h*:

If you just want to split the data without worrying about where it goes, make a single folder with all the files from the flight (e.g. .SEQ, .MOV), *cd* to that folder and just run *split_seqs* which will create a folder called *split* with the processed files.

5.1 Thermal and RGB Synchronisation

One of the most frustrating "features" of the Duo Pro R is that the infrared and RGB cameras are not synchronised in video mode. This is an odd design choice by FLIR, since both cameras can presumably be hardware co-triggered. As a result, the IR sequence has an approximate frame rate of 30fps, while the RGB stream has a frame rate of 29.97fps which is standard. This means that you can't simply match up frame numbers. If you are doing a survey flight, you are **strongly** recommended to use multiple capture mode (1 photo a second) which will at least give you synced pairs of images a the cost of frame rate.

CHAPTER 6

Indices and tables

- genindex
- modindex
- search