

Generic open-source frameworks to integrate and control infrared sensors : from laboratory standard experiments to real field measurements

18th International Conference on Quantitative InfraRed Thermography
30 June 2026 – Session “Image & Data Processing I”

Thibaud Toullier¹ Jean Dumoulin¹
thibaud.toullier@univ-eiffel.fr jean.dumoulin@univ-eiffel.fr

¹ Université Gustave Eiffel, Inria, COSYS-SII, I4S Team, F-44344 Bouguenais, France



BRIGHTER has received funding from the Chips Joint Undertaking (JU) under grant agreement No 101096985. The JU receives support from the European Union's Horizon Europe research and innovation program and France, Belgium, Portugal, Spain, Turkey.



BRIGHTER

Inria



Université
Gustave Eiffel

I4S

Contents

- Introduction / Context
- Objectives
- IRTools Architecture
- IRLab – Application for `irtools`
- Conclusions & Perspectives

Different cameras and brands

- Various communications interfaces and protocols
 - generic protocol **at best** (GenICam[®]),
 - vendor specific SDK,
 - proprietary software

Needs efforts to

- Integrate custom algorithms
- Make reproducible laboratory experiments / pipeline

⚠ | But

Most cameras have same functionalities



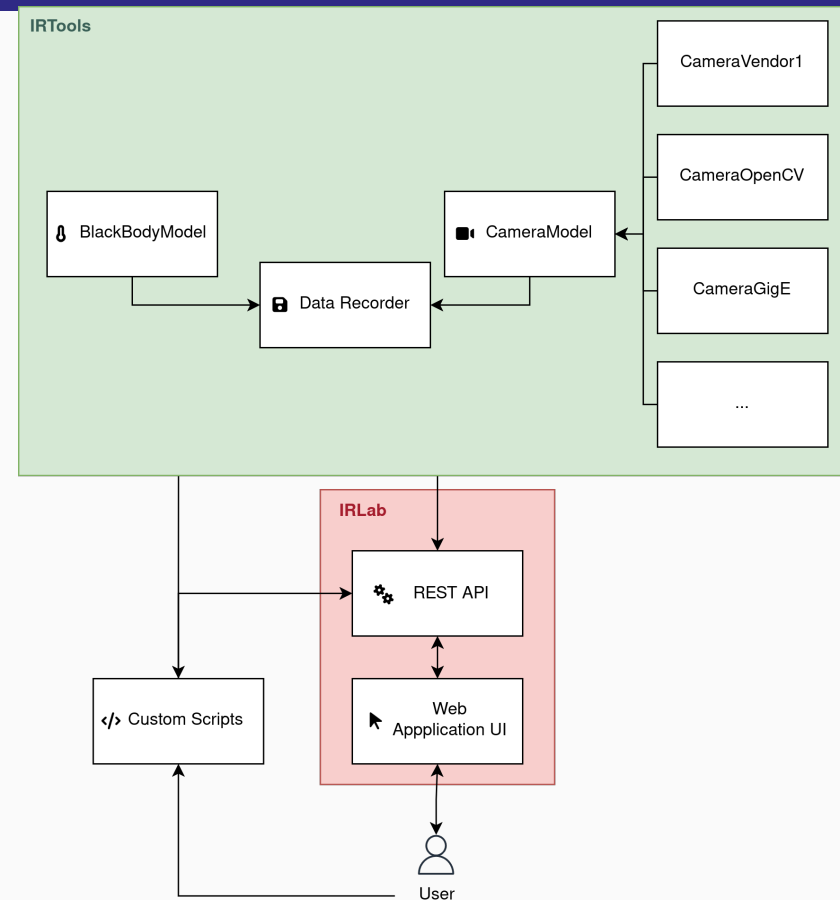
Figure 1: Different models & different SDKs

Past tools

- **IRLaw¹**
 - focused on laboratory experiments (NDT, ...)
 - extended to outdoor applications
 - RTSP streaming capabilities
- **Cloud2IR²**
 - asynchronous system
 - focused on using standards (OGC, GenICam®, HDF5, ...)

IRTools & IRLab

- Learn from previous tools
- **Generic Python API + REST API**
- **Web-based User Interface (UI)**
- **New standards**
 - WebRTC / RTSP
 - HDF5 / Zarr
 - **Easy Deployment**



¹[1] J. Dumoulin and R. Averty, "Development of an infrared system coupled with a weather station for real time atmospheric corrections using GPU computing: Application to bridge monitoring," in *Proc of 11th International Conference on Quantitative InfraRed Thermography, Naples Italy*, 2012.

²[2] A. Crinière, J. Dumoulin, and L. Mevel, "Management of Local Multi-Sensors Applied to SHM and Long-Term Infrared Monitoring: Cloud2IR Implementation," *Quantitative InfraRed Thermography Journal*, vol. 0, no. 0, pp. 1–19, Oct. 2018, doi: 10.1080/17686733.2018.1519752.

IRTools is a **Python module** made to handle specifically — *but not only* — infrared cameras. It provides a **generic API**, simply based on an abstract camera class.

One module to rule them all!

Camera Features (inspired from GenICam® and OpenCV)

- Automatic cameras discovery,
- Asynchronous,
- Recording,
- Parameters read & write,
- Streaming,
- Autofocus
- ...

i | Info

Once the camera vendor has an implementation, all the features are made available!

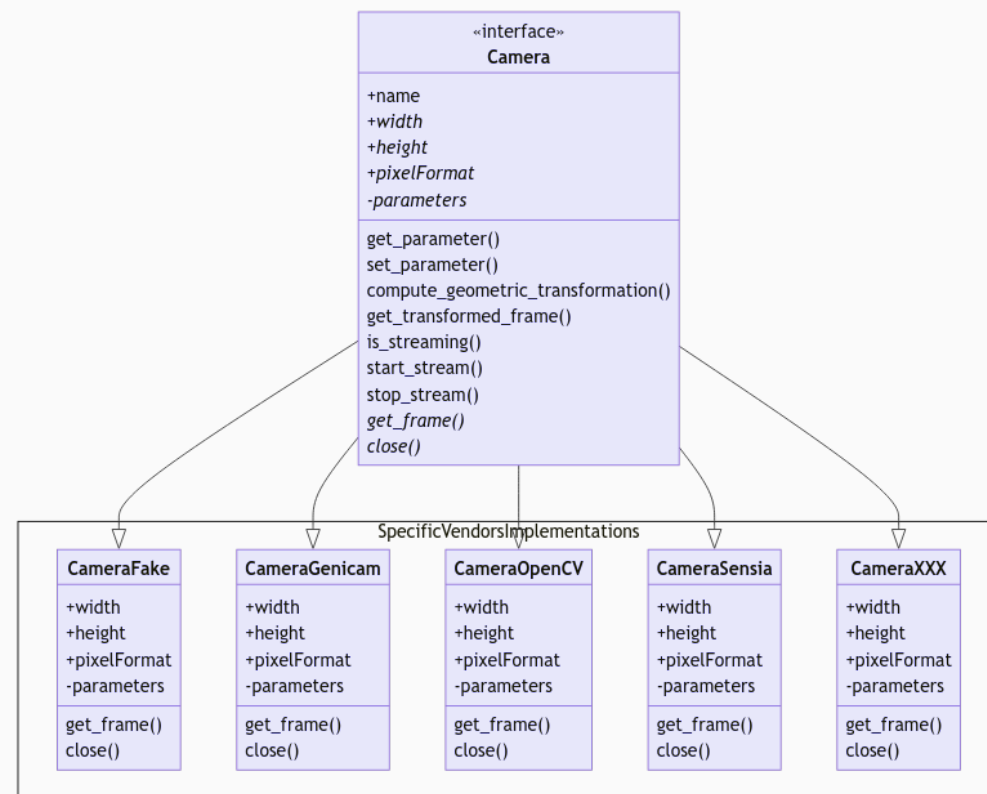


Figure 3: Principle of the architecture of a generic camera API

Why Python?

- many libraries available
- widely used among scientists
- easy to wrap existing code, even for C/C++ SDKs ctypes.

Thanks to `irtools` Python module we can:

- create codes that are compatible with different cameras
- avoid to switch between softwares
- integrate more easily new cameras with already existing code base.
- Implement reproducible experiments

Python code example

```
1 import asyncio
2 from irtools.camera import discover
3
4 cameras = asyncio.run(await discover())
5
6 for camera in cameras:
7     print(camera) # show available cameras
8
9 camera = cameras[1].get()
```

python

} Automatic discovery of available cameras

} Show detected cameras

} Use camera

Result

```
1 Aravis/Fake : @25.0FPS (512 x 512) - format: 17301505 -
2 OpenCV : @27.0FPS (382.0 x 290.0) - format: 16.0 - port: 1
3 OpenCV : @30.0FPS (640.0 x 480.0) - format: 16.0 - port: 2
4 Sensia SN#XXXXXXXX : (382 x 288) - - port: USB/1
```

bash

Derive an object from `irtools.camera.Camera`

```
1  class DemoCamera(Camera):
2      _counter: int = 0
3
4      def __init__(self):
5          super().__init__(name="Demo")
6
7      def get_frame(self):
8          self._counter += 1
9          return (
10             np.random.rand(500,500),
11             {"counter": self._counter},
12         )
13
14     def close(self):
15         pass
```

python

Give a name

Implement `get_frame` abstract method

Implement `close` abstract method

[optional] Implement discovery method(s)

```
1 @discovery_method
2 async def discover() -> CameraSummary:
3     pass
```

python

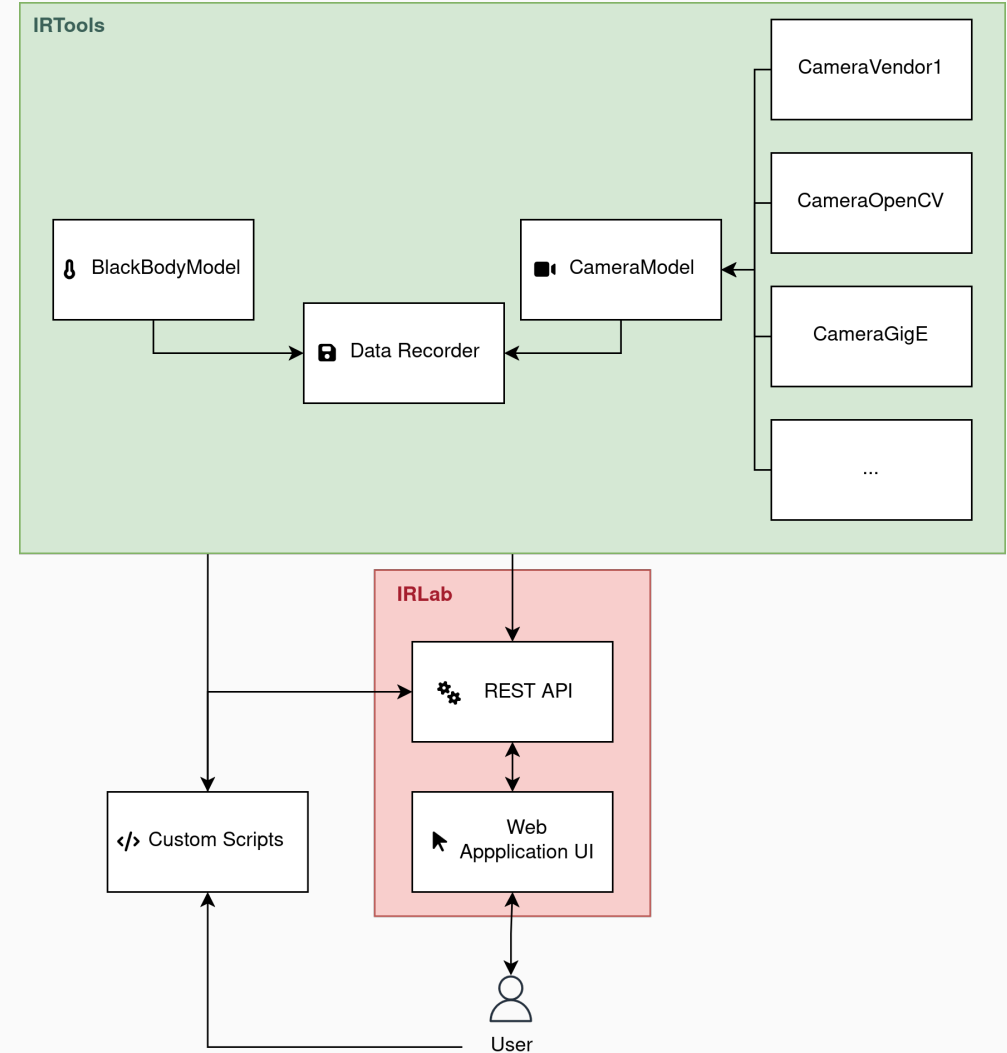
i | Info

The discovery methods are automatically registered by `irtools` and enable the camera to be listed among others.

Currently implemented protocols / libraries / SDK

- OpenCV
- GenICam® (through Aravis library) ⇒ compliant with Teledyne FLIR
- EvoCortex SDK (Optris)
- [! WIP] Sensia SDK
- [! WIP] Xeneth SDK (Xenics)
- ...

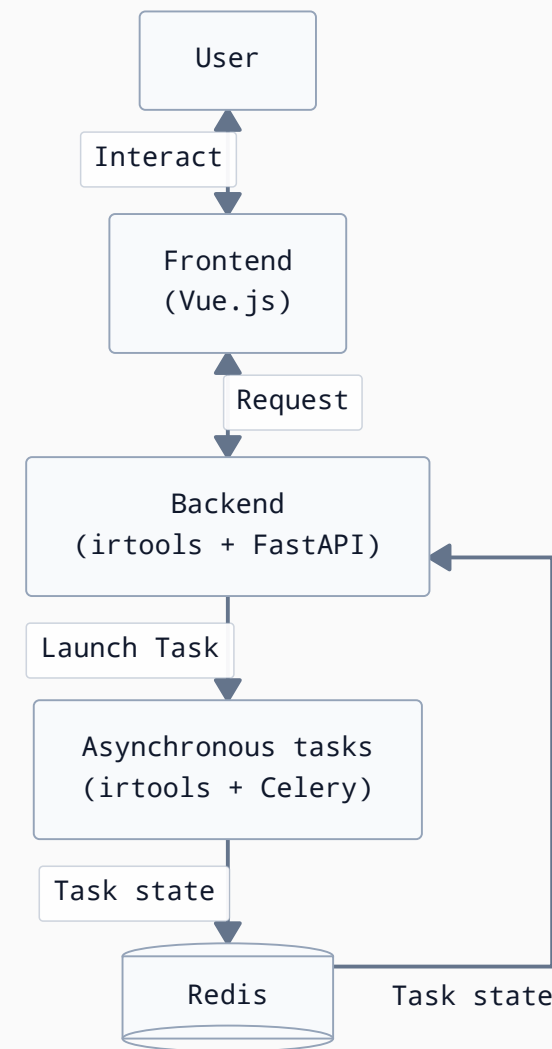
- IRTools provides the core functionalities
- IRLab will operate and manage this functionalities



IRLab

A **modern** server to communicate with cameras through a **REST API** and a **user interface** using the IRTools python API;

- Compatible with all platforms (Web interface)
- REST API to communicate with any camera (OpenAPI compatible)
- Remote camera control
- Camera Streaming (WebRTC)
- Images / Sequences recording (various images formats, hdf5, ...)
- Custom procedures
- ...

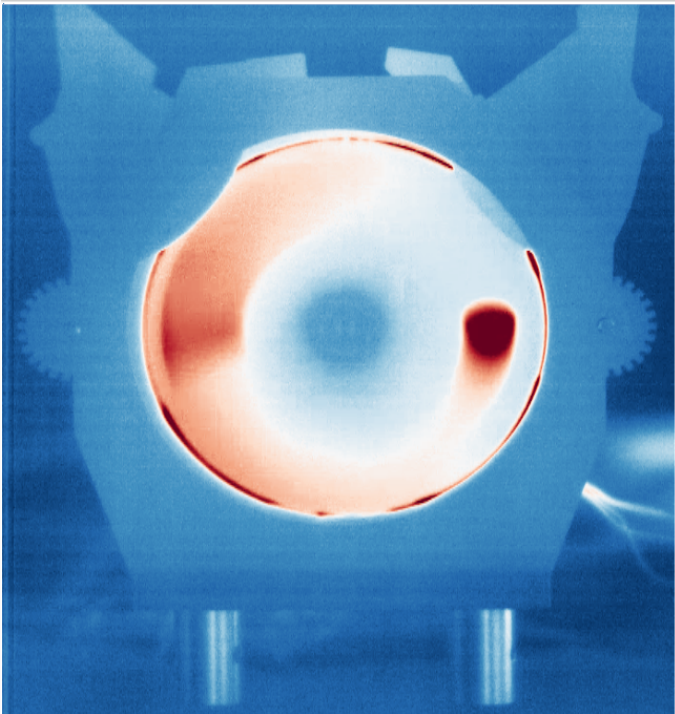


IRTools - Editor

File Edit Processing Help

Camera
0_Xenics-Ceres-1280-GigE

Colormap
Rdgy



CAMERAS AREAS OF INTEREST TASKS PROCESSING

Available cameras

Id	Status	Vendor	Image dimensions	Serial Number	Ports	FPS	Pixel Format	Actions
0_Xenics-Ceres-1280-GigE	 Connected	Xenics-Ceres-1280-GigE	696 x 872	17894	-	10	17825799	  
1_OpenCV	 Not connected	OpenCV	640 x 480	-	0	30	16.0	
2_OpenCV	 Not connected	OpenCV	576 x 360	-	1	15	0.0	
3_OpenCV	 Not connected	OpenCV	576 x 360	-	2	15	16.0	
4_OpenCV	 Not connected	OpenCV	640 x 480	-	3	30	0.0	

Graphics

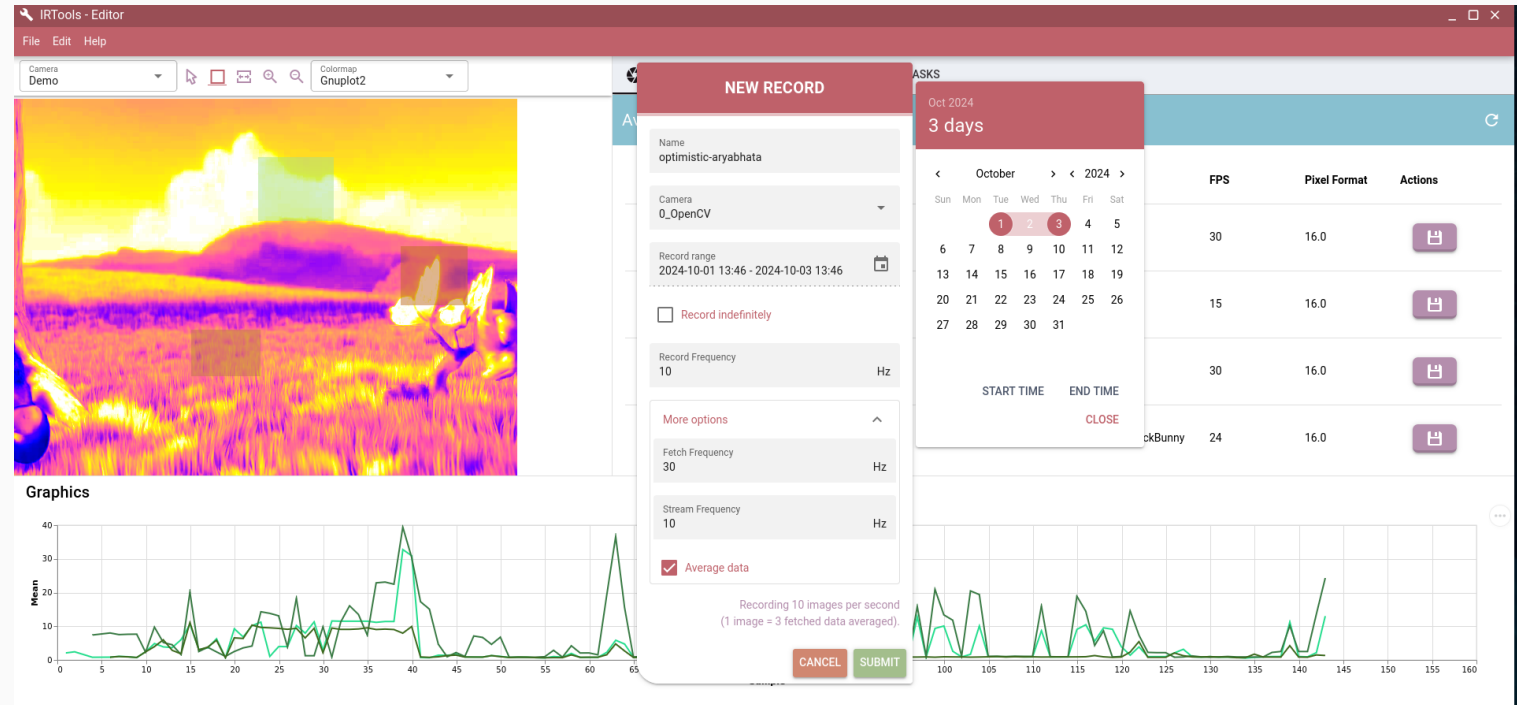
Mean

Sample



Not only useful for laboratory but also for real site implementations

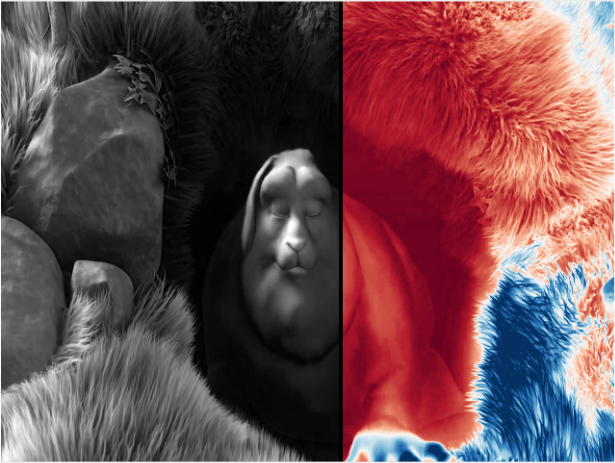
- Scheduled recording
- Acquisition frequency
- Recording frequency
- Compliant with instrumentation system supervisor¹
- HDF5 data recording including metadata



¹ [3] T. Toullier, A. Bouché, and J. Dumoulin, “Design and study of an instrumentation and software for permanent monitoring of a cable-stayed bridge,” in *Proceedings*, in Proceedings. Cartagena de Indias, Colombia, Dec. 2023, pp. 1–8. [Online]. Available: <https://inria.hal.science/hal-04303006>

Same image, two colormaps instantaneous results

- Useful to quickly test impact of colormaps for visual analysis

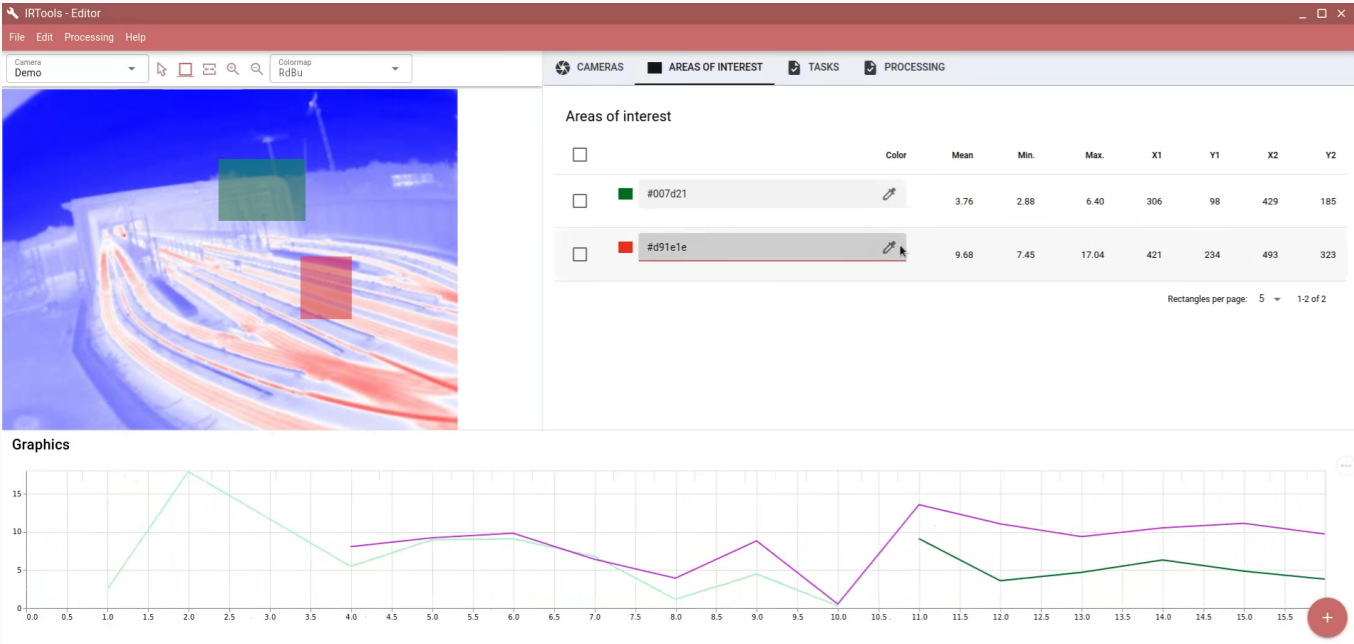


The screenshot shows the IRTools - Editor interface. On the left, there is a camera selection dropdown set to 'Demo' and a colormap selection dropdown set to 'Rdbu'. The main view displays a grayscale image of a Big Buck Bunny character and its corresponding heatmap visualization. The heatmap uses a red-to-blue color scale to represent intensity values.

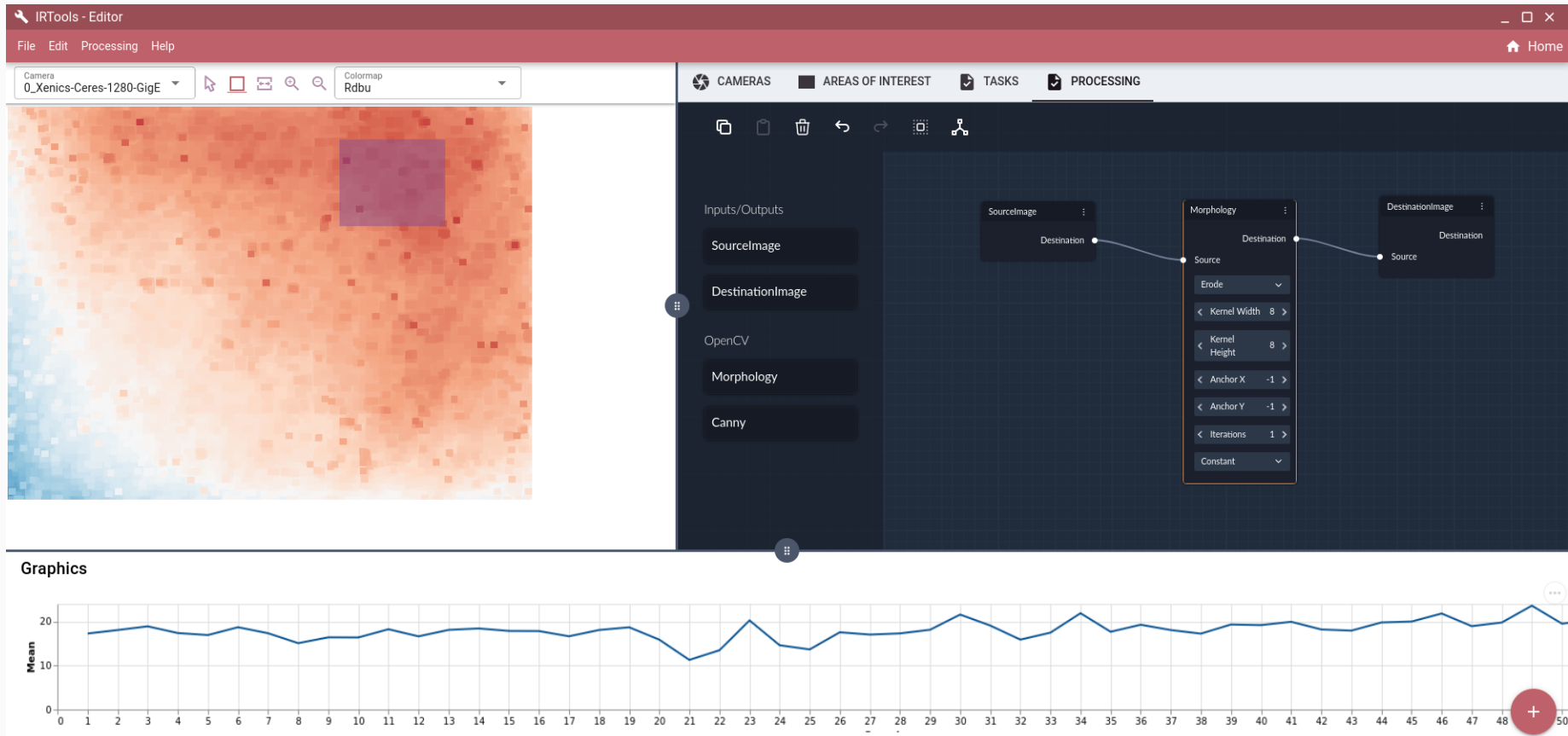
Below the image, there is a 'Graphics' section with a plot area. The y-axis is labeled 'Mean' and the x-axis is labeled 'Sample'. A red '+' button is located at the bottom right of the plot area.

On the right side of the interface, there is a 'CAMERAS' tab with a table of available cameras. The table has columns for Id, Status, Vendor, Image dimensions, Serial Number, Ports, FPS, Pixel Format, and Actions.

Id	Status	Vendor	Image dimensions	Serial Number	Ports	FPS	Pixel Format	Actions
0_OpenCV	Not connected	OpenCV	640 x 480	-	0	30	16.0	[Icon]
1_OpenCV	Not connected	OpenCV	576 x 360	-	2	15	16.0	[Icon]
2_OpenCV	Not connected	OpenCV	640 x 480	-	4	30	16.0	[Icon]
Demo	Connected	Blender	1280 x 720	-	BigBuckBunny	24	16.0	[Icon]



CAMERAS AREAS OF INTEREST TASKS PROCESSING			
Tasks			
Id	Status	Date Done	Actions
nostalgic-meitner	RECORDING		<button>VIEW RESULT</button> <button></button> <button></button>



- View and act on GenICam® settings
 - Customize calibration
 - Modify windowing
 - ...

The screenshot displays the IRTTools - Editor window, which is used for configuring GenICam camera settings. The interface is divided into several sections:

- Parameters Panel (Left):** A tree view showing the hierarchy of camera parameters. The 'AcquisitionControl' section is expanded, showing parameters like Acquisition mode, AcquisitionStart, AcquisitionStop, TriggerSelector, Trigger mode, TriggerSoftware, Trigger source, Trigger activation, ExposureTimeAbs, GainRaw, GainAuto, SensorHeight, SensorWidth, OffsetX, OffsetY, Width, Height, BinningHorizontal, and BinningVertical.
- Parameter List (Middle):** A table listing the parameters with their current values and units. For example, 'Acquisition mode' is set to 'Continuous', 'ExposureTimeAbs' is 10000 microseconds, and 'SensorHeight' is 2048 pixels.
- Control Panel (Right):** A set of buttons and dropdown menus for controlling the camera. It includes buttons for 'ACQUISITIONSTART', 'ACQUISITIONSTOP', 'TRIGGERSELECTOR', and 'TRIGGERSOFTWARE', as well as dropdowns for 'Acquisition mode', 'TriggerSelector', 'Trigger mode', 'Trigger source', and 'Trigger activation'.
- Description Panel (Far Right):** A list of descriptive text for each parameter, such as 'Start acquisition.', 'Stop acquisition.', 'Generates an internal trigger. TriggerSource must be set to Software.', 'Exposure duration, in microseconds.', 'Raw gain.', 'Automatic gain mode.', 'Full height of image sensor.', 'X offset of image, in pixels.', 'Y offset of image, in pixels.', 'Width of image, in pixels.', 'Height of image, in pixels.', 'Binning in horizontal direction.', and 'Binning in vertical direction.'

OpenAPI Compatibility

- Specification for describing, producing, consuming and visualizing web services.
- An OpenAPI Description (OAD) is a formal description of an API
 - allows to generate code, documentation, test cases, and more.

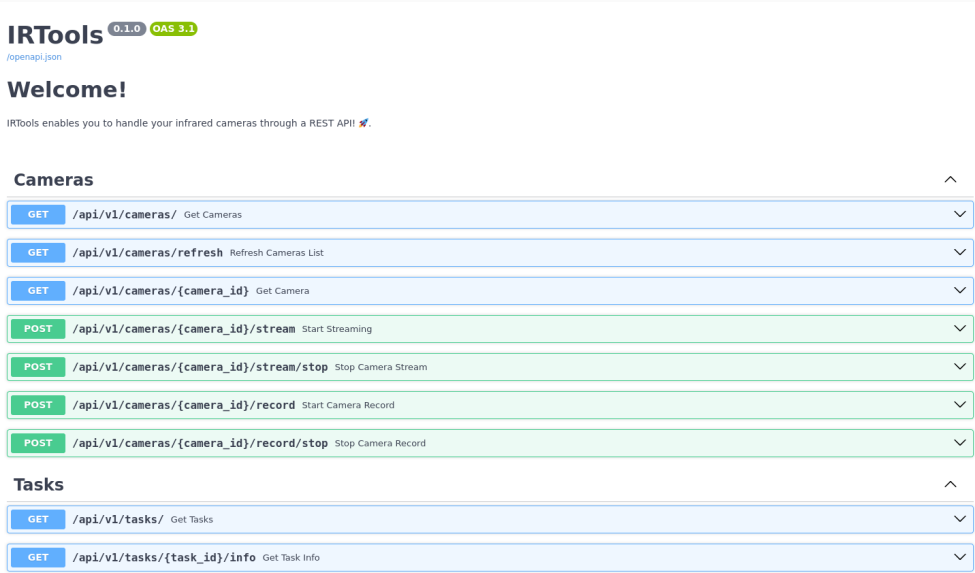


Figure 13: Swagger UI

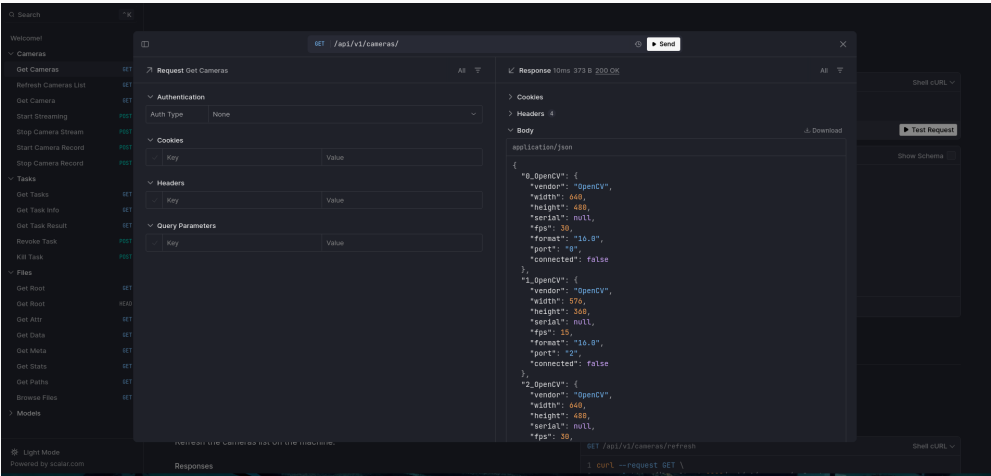


Figure 14: Redoc UI

 IRTools



IRTools Getting Started Developers Reference

Reference

- Camera
- Camera class
- CameraAravis class
- CameraFake class
- CameraGenicam class
- CameraOpenCV class
- CameraOptris class

Enumerations

- CameraParameter
- PixelFormat
- RecordFormat
- StreamMethod

Camera class

Here's the reference information for the `Camera` class, that regroups all the capabilities of any camera.

It is useful to see all the available methods of any camera, whatever the vendor use and also for users who want to implement their own camera vendor.

 **Warning**

This `Camera` is an **abstract class**, meaning that it **cannot be instantiated** on its own. A derived class (e.g. `CameraOpenCV`) must implement all the required methods and can then be instantiated.

You should not do this:

Python

```
1 from irtools.camera.camera import Camera
2
3 camera = Camera("Hello")
4 camera.height # +
```

class `irtools.camera.camera.Camera`

Bases: `BaseModel`

An abstract class to represent a Camera

Table of contents

- class** `Camera`
- attr** `name`
- attr** `logFolder`
- meth** `get_parameter`
- meth** `set_parameter`
- meth** `compute_geometric_tra...`
- meth** `get_transformed_frame`
- meth** `is_streaming`
- meth** `stop_stream`
- meth** `start_stream`
- meth** `pixelFormat`
- meth** `width`
- meth** `height`
- meth** `parameters`
- meth** `get_frame`
- meth** `close`

Conclusions

- 🚀 First working version IRTools & IRLab
- Compatible with multiple cameras (OpenCV, GenICam® (Teledyne FLIR), EvoCortex SDK (Optris), Sensia SDK, Xeneth SDK (Xenics))
- Modern UI
- Data recording including metadata (HDF5)
- WebRTC streaming
- Live OpenCV processing
- Remote control

Perspectives

- implement correction tools
 - temperature estimation algorithms
 - spatial correction
- add further utilities
 - autofocus
 - automatic cropping
- integrate laboratory hardware and test bench
 - target test bench : see this morning presentation (23)
 - black body
 - oven
- toward an **open version** next year

Generic open-source frameworks to integrate and control infrared sensors : from laboratory standard experiments to real field measurements

18th International Conference on Quantitative InfraRed Thermography
30 June 2026 – Session “Image & Data Processing I”



BRIGHTER has received funding from the Chips Joint Undertaking (JU) under grant agreement No 101096985. The JU receives support from the European Union's Horizon Europe research and innovation program and France, Belgium, Portugal, Spain, Turkey.



BRIGHTER



Université
Gustave Eiffel



Thank you!

Thibaud Toullier
thibaud.toullier@univ-eiffel.fr

Jean Dumoulin
jean.dumoulin@univ-eiffel.fr

Bibliography

- J. Dumoulin and R. Averty, “Development of an infrared system coupled with a weather station for real time atmospheric corrections using GPU computing: Application to bridge monitoring,” in *Proc of 11th International Conference on Quantitative InfraRed Thermography, Naples Italy, 2012*.
- A. Crinière, J. Dumoulin, and L. Mevel, “Management of Local Multi-Sensors Applied to SHM and Long-Term Infrared Monitoring: Cloud2IR Implementation,” *Quantitative InfraRed Thermography Journal*, vol. 0, no. 0, pp. 1–19, Oct. 2018, doi: 10.1080/17686733.2018.1519752.
- T. Toullier, A. Bouché, and J. Dumoulin, “Design and study of an instrumentation and software for permanent monitoring of a cable-stayed bridge,” in *Proceedings*, in *Proceedings. Cartagena de Indias, Colombia, Dec. 2023*, pp. 1–8. [Online]. Available: <https://inria.hal.science/hal-04303006>