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

by T. Toullier*, J. Dumoulin*

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

Abstract

Infrared cameras communication can depend on generic protocols such as GenICam®, but most of the time it relies on specific vendor Software Development Kit (SDK) and proprietary softwares. Such variety makes it diffcult to develop custom algorithms and reproducible pipelines through various camera models and brands, particularly in laboratory conditions. To overcome those difficulties, a modern framework has been developed in Python to interact with most infrared cameras on the market using either generic protocols or specific SDK if available. It provides a generic Application Programming Interface (API) and thus enables a new unified and homogeneous way to interact with infrared cameras.

1. Introduction

Infrared cameras handling can be complicated due to the various number of brands and way to communicate with them. If cameras communication can depend on generic protocols such as GenICam®[1], it often depends on specific vendor Software Development Kit (SDK) and proprietary softwares. As a consequence, custom algorithms integration and reproducible pipelines through various camera models, particularly in laboratory conditions, are made difficult. Switching between proprietary softwares is not convenient and can be tedious when managing multiple cameras at the same time.

To overcome those difficulties, a modern framework called “IRLab” has been developed in Python. It provides a generic Application Programming Interface (API) to interact with most infrared cameras on the market using either generic protocols, or specific SDK when available. The main advantage of the framework is to provide a common interface, getting rid of vendor specific code for users, enabling code sharing between cameras models and making it easier to integrate new cameras with already existing code base.

Thanks to its open-source specificity and its design, the software enables the easy integration of new camera models. Once the basic camera functionalities are integrated, the API automatically provides parameters settings, recording management, and streaming capabilities (through webRTC).

Based on this API, a web-based interface has been developed that automatically scan and discover known cameras on the machine. It provides a user-friendly way to plan video or image recording, interact with different cameras through the same and unified interface. It also enables the remote control of cameras, if installed as a server.

In this paper, the software architecture and main components will be presented first. Then, the main features, particularly the ability to stream and record data will be introduced. Finally, some application use-cases will be highlighted and conclusion and perspectives given.

2. IRLab Software Architecture

2.1 Components of IRLab

IRLab is based on the generic definition of what an infrared camera is, mainly inspired by the GenICam® [1] and OpenCV [2] specifications. To work within the framework, a new camera model must derive from this abstract class and it must implement some basic functionalities, such as the `get_frame` method to retrieve a frame. Once those few methods are implemented, IRLab framework automatically provides advanced functionalities such as WebRTC streaming, camera parameters management and visualization, calibration routines, *etc.* The overall IRLab framework is made of a modern GUI called IRLab by extent and a Python API called IRTools.

2.2 IRTools

The developed Python API enables an agnostic brand interaction with infrared cameras. An example of such interaction is given on Figure 1 where all available and compatible cameras on the machine are automatically retrieved. Having agnostic brand API enables reproducible developments and experiments automation accross different hardware. IRTools provides functions for recording, streaming, and setting parameters out of the box for different protocols and cameras.

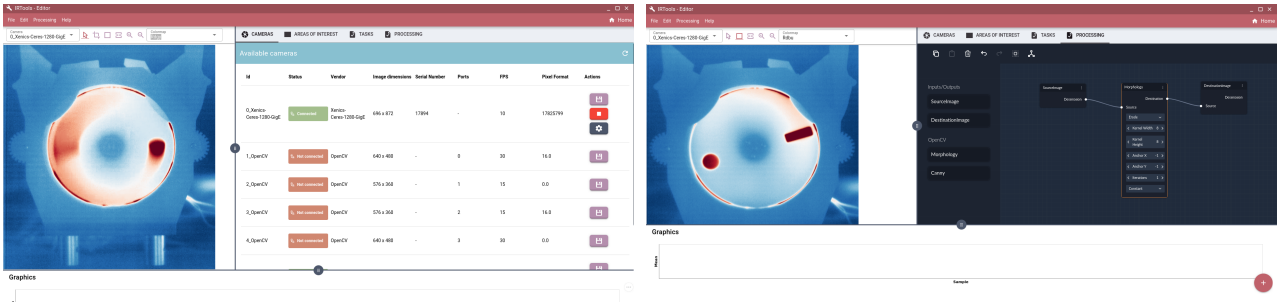
```python
import asyncio
from irtools.camera import discover
# This function will try to discover any available camera based on the vendors provided by IRTools
cameras = asyncio.run(await discover())
for camera in cameras:
  print(camera) # IRTools fetches some basic informations about the available cameras. You can connect to them easily later on.
```

```
Aravis/Fake : @25.0FPS (512 x 512) - format: 17301505 -
OpenCV : @27.0FPS (382.0 x 290.0) - format: 16.0 - port: 1
OpenCV : @30.0FPS (640.0 x 480.0) - format: 16.0 - port: 2
OpenCV : @15.0FPS (576.0 x 360.0) - format: 16.0 - port: 4
Sensia SN#XXXXXXXX : (382 x 288) -  - port: USB/1
```

Fig. 1. *Listing the availables cameras on the machine using the brand agnostic function* `discover` *in Python (left). The output of the Python script (left) is presented on the right.*

2.3 IRLab webserver

IRLab webserver has been built on top of the previously presented API, providing a modern, feature-rich web user interface and thus compatible with all modern platforms. It provides a REST (Representational State Transfer) API to interact easily and remotely to cameras. The backend has been built using the FastAPI framework[3] and Celery[4] for the management of asynchronous tasks. A Redis[5] broker is used for handling the state of the tasks and the frontend is built using VueJS[6] framework (see Figure 2b). The overall framework is documentated and a Swagger[7] interface to test the REST API is provided. Futhermore, a pipeline editor, connected to a WebAssembly compiled version of OpenCV enables live image processing (see Figure 2b).



(a) *Streaming view of a moving target.* **(b)** *Live OpenCV processing pipeline editor.*

Fig. 2. *Screenshots of the developed web interface*

2.4 Tested cameras

Cameras that have already been integrated and tested with the framework are listed in Table 1. Please note that the framework is also compatible with various visible cameras, thanks to the OpenCV IRTools module developed.

Camera Brand	Camera Model	Interface	Communication / Framework
Teledyne FLIR	A700	Ethernet	GenICam
Optris	Xi400	USB	Custom SDK
Xenics	Prototype	Ethernet	GenICam
Sensia	Prototype	USB	Custom SDK
PORT Connect	Full HD Webcam 900078	USB	OpenCV

Table 1. *Cameras models tested with the framework. When the model is referred as* `prototype`*, it is a prototype issued from the Brighter European project[8]*

3. Conclusions and Perspectives

This work introduced IRLab, an open-source framework designed to simplify the integration and management of infrared cameras from different manufacturers. By providing a unified API and a web-based user interface, the framework abstracts vendor-specific implementations and offers a consistent environment for camera configuration, data acquisition, recording, and real-time streaming. This approach improves software portability, facilitates reproducible experimental workflows, and reduces the effort required to integrate new devices into existing systems.

The modular architecture of IRLab demonstrates that heterogeneous infrared imaging systems can be operated through a common software layer while maintaining access to advanced camera functionalities. The integration of WebRTC-based streaming and remote control capabilities further extends its applicability to distributed laboratory environments, collaborative experiments, and remote monitoring scenarios.

Several directions can be considered for future developments. First, support for additional camera models and communication protocols could further increase the interoperability of the framework. Second, the integration of advanced data-processing pipelines, including real-time image analysis, calibration methods and additional laboratory hardware control compatibility, would enable more sophisticated applications directly within the acquisition workflow. Third, enhanced synchronization mechanisms between multiple cameras and external measurement devices could facilitate complex experimental setups.

Acknowledgments

The authors acknowledge the BRIGHTER HORIZON project. 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.

References

[1] GenICam – EMVA. Available at https://www.emva.org/standards-technology/genicam/ (June 2026), version 5.0.

[2] OpenCV modules — OpenCV Tutorials. Available at https://docs.opencv.org/5.0/ (June 2026), version 5.0.

[3] FastAPI - FastAPI. Available at https://fastapi.tiangolo.com/ (June 2026), version 5.0.

[4] Celery - Distributed Task Queue — Celery 5.6.3 documentation. Available at https://docs.celeryq.dev/en/stable/ (June 2026), version 5.0.

[5] Redis. Redis - Real-time data for agents & apps. Available at https://redis.io/, (June 2026).

[6] Vue.js. Available at https://vuejs.org/, (June 2026).

[7] API Documentation & Design Tools for Teams | Swagger. Available at https://swagger.io/, (June 2026).

[8] Chips JU. BRIGHTER: Breakthrough in micro-bolometer imaging. Available at https://project-brighter.eu/, (June 2026).